

Data and Analytics Skills for Your Career Security

Keeping it simple, only the skills you're likely to need
Richard G. Lamb

For those of us who are role holders in enterprise functioning, the personal purpose of acquiring practical working skills in data and analytics is to be able to better do what we already do and find new ways to do better yet. It follows that if you are a role holder who brings and incorporates data and analytics methods in your thoughts and tasks, your career outlook will be more secure and exciting. The book is written to be your gateway to the skills and to be the templates with which you will install the methods in your operational roles.

We all know that the field of data and analytics is huge and intimidating. It is a long slog to becoming comfortable. During the author's own long slog until arriving at the book, something exciting bubbled to the surface. There is a big difference between what we need to know and everything there is to know. We need to know what is possible as insight for decisions and functioning, we need to know how to get to the insight and, finally, we need to be able to interpret the insight. Just as the book does, we can leave the rest to the data scientists.

About the Author: In 2003, Richard Lamb, while struggling to get at the history captured in the databases of operational systems, found the skills to extract datasets of related history and join them in a super table of variables to make possible what was being envisioned for operational effectiveness. In 2014, Richard realized that, with statistical analytics and free enabling powerful pc-level software, an enterprise could ask and answer questions of operational effectiveness that are otherwise not possible. His activism to bring the epiphanies into the careers of role holders in the mainstream of operations has arrived at this book to explain data and analytics through the demonstration of methods.

Richard is a Registered Professional Engineer and Certified Public Accountant. He has previously authored two books: Availability Engineering and Management for Manufacturing Plant Performance, and Maintenance Reinvented for Business Performance. He has a BSCE, BBA and MBA from the University of Houston and a graduate-level Applied Statistics Certificate from the Texas A&M University.

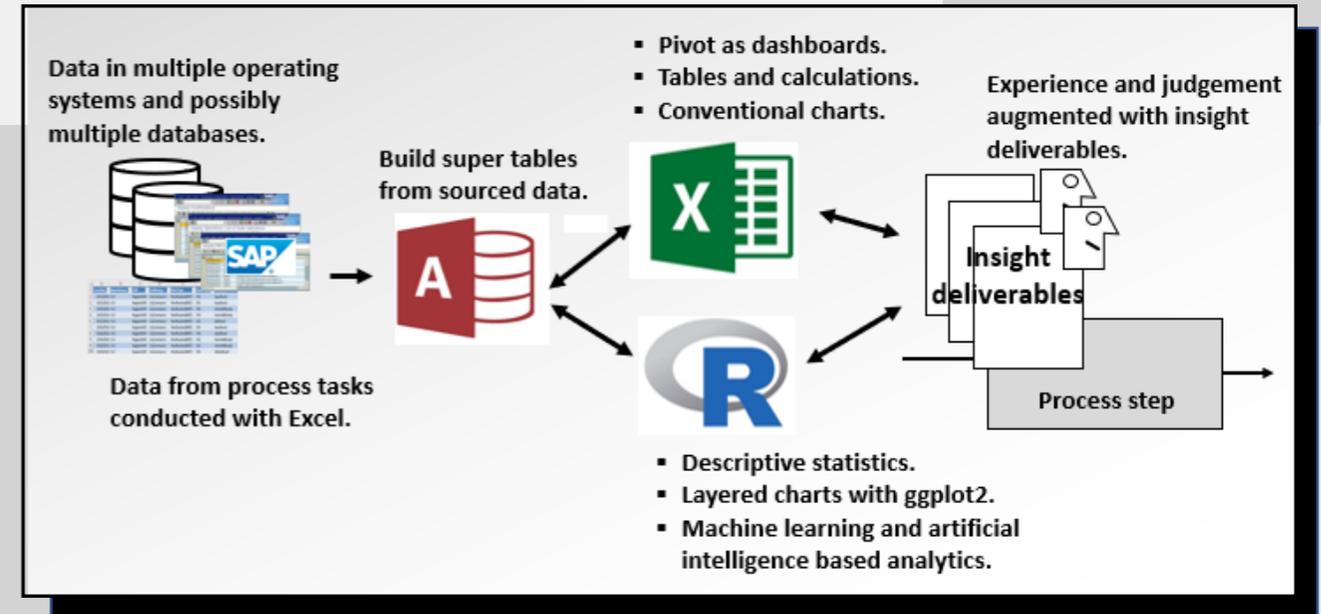
<https://analytics4strategy.com/data-and-code>



Lamb
Data and Analytics Skills
For Your Career Security

Data and Analytics Skills for Your Career Security

Keeping it simple. . . only the skills you're likely to use



Richard G. Lamb

ISBN 978-1-7343947-0-2

© 2021 Richard G. Lamb
Analytics4Strategy

All rights reserved. Except as permitted under the United States Copyright Act of 1976, no part of this book may be reproduced or distributed in any form or by any means or stored in a database or retrieval system without the prior written permission of the author.

Information contained in this work has been obtained from sources believed to be reliable. Neither the author nor publisher guarantee the accuracy or completeness of any information published herein and neither the author nor publisher shall be responsible for any errors, omissions, or damages arising out of the use of this information. The work is published with the understanding the author and publisher are supplying information, but not attempting to render professional legal, accounting, engineering or any other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Trademarks: Microsoft, Microsoft Office, Excel Access, Oracle, SAP, Tableau, Power BI, Maximo and Track are registered trademarks in the United States and other countries.

Contents

Preface	xiii
CHAPTER 1 Operational Availability is the Purpose.....	1
1.1. Attributes of Availability Performance.....	2
1.1.1. Availability as Probability.....	2
1.1.2. Constituent Probabilities	2
1.2. Availability as Subtypes	4
1.2.1. Maintenance Tasks and Activities.....	4
1.2.2. Subtypes of Availability.....	6
1.2.3. Restating the Equation.....	7
1.2.4. Interrelationship of Subtypes.....	9
1.3. Top factors of Operational Availability.....	9
1.4. Cost-Effectiveness in the Definition.....	11
1.5. Data-Driven as the Means.....	13
Bibliography	14
Chapter 2 Data, Analytics and Software to be Data Driven.....	15
2.1. Big Picture	15
2.1.1. What It Looks Like.....	15
2.2.2. Critical Mass and Grass Root.....	16
2.2.3. Essential Definitions.....	19
2.2. What and Why of Access and “R”.....	22
2.2.1. MS Access for Super Tables	22
2.2.2. “R” for the Analytic Core.....	26
2.2.3. Layered Charting.....	29
2.3. Insight Deliverables	31
2.3.1. System Reported and Recountive Insight	31
2.3.2. Know-Thy-Data Insight	32
2.3.3. Modeled Insight.....	33
Bibliography	41
Chapter 3 Super Tables from Operational Data	43
3.1. Perspective	44
3.2. Scenarios for Demonstration.....	45
3.3. Case 1: Foundation Super Table	47
3.3.1. Identify, Extract and Import or Link	47
3.3.2. Translation Tables	49

vi | Contents

3.3.3. Overview of the Query Process.....	50
3.3.4. Joins and Their Ramifications.....	53
3.3.5. Coding the Design Grid.....	56
3.3.6. Generate the Super Table	61
3.4. Case 2: Seek Outlier Work Orders.....	63
3.4.1. Measurement for Outliers.....	63
3.4.2. Activating Aggregation	65
3.4.3. Planning the Aggregation.....	66
3.4.4. Arriving at the Z-Score for Hours	67
3.4.5. Student’s T-Score as Alternative.....	70
3.4.6. Expression and Where Aggregations	71
3.5. Case 3: Seek Outliers with Lead Craft.....	72
3.5.1. Planning the Lead Craft Aggregation.....	72
3.5.2. Arriving at Z-Score with Craft Lead	72
3.6. Comparison of Findings.....	78
3.7. SQL in the Background	79
3.8. Administration of Tables	80
Bibliography	81
Chapter 4 The “R” Software in Action	83
4.1. Approach to Reach Critical Mass	83
4.2. “R” Installation and Session	84
4.3. Basics of “R” Demonstrated	87
4.3.1. Demonstration Case	88
4.3.2. Import Data to the Session	89
4.3.3. Know-Thy-Data Analysis.....	90
4.3.4. Correlation Analysis.....	95
4.3.5. Partial Correlation Analysis	100
4.3.6. Correlation Compared to Regression	103
Bibliography	108
Chapter 5 Layered Charting to Know Thy Data	109
5.1. Inspect for Missing Data.....	110
5.1.1. From Super Table to R.....	111
5.1.2. Find the Missing Data	113
5.2. Visual and Statistical Inspection.....	117
5.2.1. Load and Survey the Data	119
5.2.2. Test for Normal Distribution.....	122
5.2.3. Inspect Correlation Between Variables.....	131
5.2.4. Inspect Centrality and Spread.....	139

5.2.5. Inspect Categorical Variables.....	148
5.2.6. Inspect Variables Over Time.....	151
5.3. Save and Disseminate	157
Bibliography	158
Chapter 6 Unearth and Rectify Bad Data.....	159
6.1. Bad Data and Rectification	159
6.2. Strategies for Numeric Data.....	161
6.2.1. Spotting Outliers and Influencers.....	161
6.2.2. Models for Numeric Imputation.....	163
6.3. Strategies for Categorical Data	164
6.3.1. Finding and Rectifying Misclassifications	164
6.3.2. Models to Rectify Misclassifications	165
6.4. Subsets and Multilevel Models.....	166
6.5. The Models in Other Purposes.....	168
Bibliography	169
Chapter 7 Relate Operational Variables to Outcomes	171
7.1. The General Linear Model.....	172
7.1.1. Systematic Component.....	172
7.2.2. Linking Function and Solution Equation	174
7.2. Process of Machine Learning.....	177
7.2.1. Load, Inspect and Prepare Data.....	179
7.2.2. Dummy Variables.....	182
7.2.3. Select the Variables.....	185
7.2.4. Transforming Variables for Linearity	193
7.2.5. Test the Model for Generalization.....	200
7.3. Extreme Cases Cleansed by AI.....	207
7.3.1. Flag Cases that Change Parameters.....	208
7.3.2. Flag Outliers Cases to the Mainstream.....	211
7.3.3. Create Datasets for Imputation.....	213
7.4.4. AI to Impute Cleansed Data	215
7.4. Individual Predictions and Intervals	217
Bibliography	219
Chapter 8 Achieve Entirety of Data.....	221
8.1. Layered Structure.....	222
8.2. Current at Capture.....	225
8.3. Access Upon Demand.....	230
Bibliography	236

Chapter 9 Workload-Based Budget and Variance	237
9.1. Framework for Budget and Variance.....	238
9.1.1. Mission, Structure and Derivation.....	238
9.1.2. The Two Dimensions	240
9.1.3. Budget	242
9.1.4. Variance.....	243
9.2. From History to Budget to Variance.....	245
9.2.1. The Case.....	246
9.2.2. From History to Budget.....	249
9.2.3. Table of Actuals	253
9.2.4. Variance table.....	255
9.2.5. Variance Report and Analysis.....	258
9.3. Variance as Systemic or by Outliers.....	262
9.3.1. The Case.....	263
9.3.2. Year to Date Aggregation.....	263
9.3.3. Flag Outliers for Investigation	266
9.3.4. Pivot Report of Outlier Orders	268
9.3.5. Systemic Variance After Outliers.....	269
9.4. Maintenance Craft Capacity	270
9.4.1. The Case.....	271
9.4.2. Capacity Report and Analyses	272
9.4.3. Tables to Queries to Capacity Report.....	274
Bibliography	278
Chapter 10 Through the Lens of Time Series	279
10.1. Working with Time Series.....	281
10.1.1. Time Series as an R Object	281
10.1.2. Transforming Data to a TS object	284
10.1.3. Windows to Time Series	285
10.1.4. Intersection of Series	287
10.2. Analytics with Time Series.....	289
10.2.1. Notation, Models and Smoothing.....	290
10.2.2. Decomposition of the Series.....	291
10.2.3. Autocorrelation and Correlograms.....	295
10.2.4. Lead-Lag Variables	299
10.2.5. Seven-Day Cycles	302
10.3. Forecasting into the Future	305
10.3.1. Holt-Winters.....	305
10.3.2. ARIMA.....	311

10.3.3. Test as Deterministic or Stochastic	314
Bibliography	318
Chapter 11 Budget-Based Schedule and Craft Capacity	319
11.1. Scheduling Framed Top-Down.....	319
11.2. Budget-Based Schedule Cycle.....	321
11.2.1. Establish Baseline Schedule and Craft Capacity.....	321
11.2.2. Conduct Weekly Scheduling and Execution	324
11.2.3. Datasets, Measures and Reports Stage	327
11.3. Tables to the Schedule Cycle.....	329
11.3.1. System of Tables to Build and Maintain.....	329
11.3.2. Weekly Schedule as Table	330
11.3.3. Day Schedule as Table	331
11.3.4. Craft Hours Table.....	333
11.3.5. Craft Classification Translation Table	334
11.3.6. Super Table to Measurement.....	336
11.4. Measures and Methods	341
11.4.1. Confirm Sustainment.....	341
11.4.2. Planned and Scheduled Execution	343
11.4.3. Job Plan Accuracy and Variance.....	346
11.4.4. Measures as Time Series	349
Bibliography	351
Chapter 12 Elapsed Time Through Stages	353
12.1. The Insight We Need	354
12.1.1. Retention in the Pipeline	355
12.1.2. Exit Event from the Pipeline	357
12.2. How it Works.....	358
12.2.1. Life Data, Event and Window	358
12.2.2. Construct of Retention and Event	360
12.2.3. Equations to Retention-Event.....	362
12.3. Retention and Event Models.....	364
12.3.1. Data to the Analytic.....	364
12.3.2. Empirical Modeling.....	366
12.3.3. Parametric Modeling.....	368
12.3.4. Multiple Predictor Variables	370
12.4. Weibull Plot	373
Bibliography	375

Chapter 13 Prove There is a Difference	377
13.1. Effects Distinguished as Models.....	378
13.1.1. Single Factor, Two Conditions.....	379
13.1.2. Single Factor, Three or More Conditions.....	379
13.1.3. Covariant Variable in ANOVA.....	381
13.1.4. Two or More Factorial Variables.....	382
13.1.5. Measures are Repeated.....	383
13.1.6. Mixed Models.....	384
13.1.7. Robust Models.....	384
13.2. Test by Post Hoc and Contrasts.....	385
13.2.1. Post Hoc Adjusted Confidence Limits.....	385
13.2.2. Contrasts as Effect.....	386
13.3. Effects Analysis Cases.....	389
13.3.1. Two-Mean T-Test.....	389
13.3.2. One-Way ANOVA.....	394
13.3.3. Analysis of Covariance, ANCOVA.....	404
13.3.4. Factorial ANOVA.....	413
13.3.5. One-Way Repeated Measures.....	422
13.3.6. Repeated-Measures Factorial.....	427
13.3.7. Mixed Models.....	435
Bibliography.....	436
Chapter 14 Recover Lost Classifications	437
14.1. Generalized Procedure.....	438
14.2. Logistic Regression as Classifier.....	440
14.2.1. Concept of Logistic Regression.....	441
14.2.2. Case, Dataset and Preparation.....	443
14.2.3. Build and Interpret Model.....	445
14.2.4. Classification as Probability.....	449
14.2.5. Analysis of Residuals.....	451
14.2.6. Classify Cases.....	451
14.2.7. Test for Multicollinearity and Linearity.....	454
14.2.8. Logistic Regression as Predictive Insight.....	457
14.3. Naïve Bayes Probability as Classifier.....	459
14.3.1. Concept of Naïve Bayes.....	459
14.3.2. Case, Dataset and Exploration.....	460
14.3.3. Cleansing and Standardizing Text.....	462
14.3.4. Visualize Text as Word Clouds.....	464
14.3.5. Create the Document Term Matrix.....	466

14.3.6. Create Training and Test Datasets.....	467
14.3.7. Create Indicators of Frequent Words	468
14.3.8. Train and Evaluate Model	469
14.3.9. Classify Cases.....	472
Bibliography	473
Index	475

Preface

Does the steady drumbeat for data and analytics in your newsfeed make you uneasy? Are you worried that data and analytics will be the new Excel for which skills went from optional to mandatory for everyone's career? Are you searching for a way in but have not yet found a door other than some prohibitively huge and long undertaking to reach critical mass?

In other words, should you be worried for your career health? Yes, if your role engages with operating systems and Excel. And yes again, if your role in a managerial position entails decision-making upon the interpretation of standard reports from the operating system or Excel deliverables from other role holders.

You should be worried. As a few of these role holders progressively bring data and analytics skills into their roles, they will set a new standard that is immediate, visible and significant to the enterprise. Without the same skills you will not be able to meet the standard. Better yet, the best assurance of career security is to be one of those who sets the standard.

Maybe this is not new bad news to you. You see it coming. Your problem is that you have not yet stumbled across a way into the skills, short of awful. However, there is good news. Because I decided to take the long and awful path, this book is your serendipitous good luck. My bad luck was that a book such as this had never been written.

Why the book is your good luck is evident by my experience up to writing it. As an operational role holder, rather than data scientist, I had to learn much of everything there is to know about data and analytics before I could reduce the explanation of data and analytics to what you, in your operational role, need to know.

It took me six years to learn much of what there is to know. It then took me another more than a year to reduce what I had to learn to what you need to know to vaccinate your career against obsolescence and decline.

Philosophy and Approach

The book is written to explain and demonstrate data and analytics as methods that are relevant to all enterprise operations rather than the universe of all human endeavors. The strategy works because all operations must deal with the same issues and sub-operations. Every operation has, in some form, requirements for setting direction, aggregate planning,

xiv | Data and Analytics Skills for Your Career Security

budget and variance control, planning, scheduling and executing the core task, staffing and optimizing the retention time of the core task in the stages through the operation.

It follows that all operations will thrive on the same data and analytics skills. Regardless of the type of operation, it is obviously impactful that its role holders be able to extract, explore, cleanse and mold to purpose the data captured in their operating systems. Regardless of the type of operation, it is obviously impactful that role holders be able to augment their experience and judgement with the insight that data and analytics make possible. Furthermore, the best augmenting insights are not and never will be available from the operating systems their roles depend upon.

Accordingly, a full explanation by demonstration of what you need to know, must necessarily be in the context of an operational domain. Therefore, as a context to all explanations by demonstration, the book has selected a domain in the management of manufacturing plants as a production asset. The type of operation is known as asset management. For those in the field of asset management, the book is a doubly serendipitous good fortune.

Chapter 1, Operational Availability is the Purpose, introduces asset management with respect to how it plays in the enterprise. It then establishes the measures of performance and the driving aspects to the measures. Because data and analytics should ride on purpose, you are advised to think out the equivalent framing of the operation in which you hold a role and the operations around it.

Of course, data and especially analytics are not, nor can be made, easy subjects. That is even after being bound within only what you need to know rather than all there is to know. As someone who read the book said, “You have to shut the door and turn off the TV.”

However, you will not need to go far before you begin experiencing the payback of your concentration. At each chapter you will return to work with an actionable idea to upgrade the output of your role and the roles of others.

Let’s talk about approaching the book to maximally bring about immediate payback to your organization and you as well. The book is written as the sequence of bringing the practices of asset management to be data driven. Asset management practitioners have got to love that. However, there is alternative path through the chapters for maximal “next-day” payback upon each stepwise accumulation of skills.

You should begin with Chapter 2, Data, Analytics and Software to be Data Driven. You need to know the territory. Just as important, you will discover that you already have at hand everything you need without first engaging management for permission to act on your payback idea or plead with them to buy you specialized software. Also important, the chapter will establish the definitions and demarcations with which you will be able calm

management down by explaining why it is that data drivenness entails learned skills with immediate operational ramifications rather than capital and costs for exotic software and endeavors.

Here is one thing I discovered while learning much of what there is to know about data and analytics. Most of the power of data-drivenness is achievable with only the skills of working with data. Only a small number of ideas, less than 20 percent, call for analytics. In other words, if the forces of the universe commanded that I was to be forever limited to what I could make happen with data, I would be disappointed but still excited about what I have been left with.

Data Skills and Methods

When you know how to extract, explore and mold data into a dataset for purpose, your operation will immediately jump forward in its ramification for the enterprise. As a thought exercise, while you work through Chapter 3, Super Table from Operational Data, imagine what your operation would look like and perform like if every role holder were able to skillfully work with data as normal to their role. In fact, it is a readily achievable vision.

Working through Chapters 3, 8, 9 and 11, you will see in action almost every technique of building what the book calls super tables. The techniques are demonstrated in MS Access because the software is available to everyone by virtue of their organization's MS Office license. Furthermore, the principles of data that are demonstrated in Access travel easily to other software such as Tableau and Power BI.

Chapter 3, Super Tables from Operational Data, explains how data is extracted from operating systems, pulled into Access and joined together in a super table of all related variables. This is in contrast to Excel which leaves us divided and conquered because our data, although related, is present to us in disconnected tables. The chapter then explains how to create summary variables in the super table, a technique which loom huge in the ability to build insight from the data.

Chapter 8, Achieve Entirety of Data, is an important chapter because we must make all of our operational data available to building super tables. The chapter presents three frequently observed obstacles to entirety. To confront them, you will learn what changes must be made to operational roles and systems as necessary to achieve entirety.

Chapter 9, Workload-Based Budget and Variance, and Chapter 11, Budget-Based Schedule and Craft Capacity, demonstrate data skills in the context of expanding super tables to perform tedious, complex operational tasks. The value of data skills is highlighted by demonstrating a collection of related organizational tasks that are not otherwise possible in asset management and their absence has been a big impediment to enterprise

performance. It will jump out at you that complex laborious reporting and control tasks can be reduced to merely updating the source tables at the input side of the super table.

Analytics Skills and Methods

What will be your career health at this juncture of acquired skills? It will be excellent. However, reaching preeminence would be even more excellent. More enticing is that preeminence is within your grasp because now “you know data.”

Some time ago, I met a person in a large corporation whose meal ticket was his ability to produce Excel pivot charts and tables. The tables were not even super tables, nor did he know how to build them. Furthermore, the graphic capability of Excel to the fellow’s claim to fame was based on single-perspective charting that was invented over 100 years ago, some as far back as the 1600’s. Dashboards are an attempt to get past the limitation by placing multiple single-perspective charts in a single display. This is in contrast with layered perspectives in a single chart.

Thumb through Chapter 5, Layered Charting to Know Thy Data. You will see in action what is ggplot2 in the R software. The differences and ramifications will jump out at you. Imaging pulling such power to visualize data and measures into your role. In contrast, my buddy with his Excel pivots would be lost in your dust.

However, there is a bit of bad news. To get to layered visualization, you must step into the R software. As you will know from Chapter 2, R is a powerful and open software available for you or your firm to download without cost, limitations or strings.

The purpose of Chapter 4, The R Software in Action, is to get you up and running in R. Once again, the chapter is used to explain real life insights by demonstration. The chapter demonstrates the procedure to inspect a dataset statistically, reveal missing data, test variables for normal distribution and inspect the correlations between variables.

The philosophy of the book is that frequently occurring code will emerge from the explanations. Additional commonly occurring coding will emerge in the code for layered charting with the previously mentioned ggplot2 as well as from all of the other chapters to explain analytics. And just as for the Access code of the previously introduced chapters, you are left with R templates to substitute in your own variables, and R skills and code you will need to know.

You will arrive at another milestone upon working through Chapters 4 and 5. You will have accumulated the chops to begin moving into analytics. This is good because there is still an elephant in the room that requires analytics. It is to find and cleanse bad data. Of course, we should think of our best solutions for bad data are action taken to prevent a future of bad data in the operating system.

If there is bad data, what will be our strategy to rectify it? Chapter 6, *Unearth and Rectify Bad Data*, presents the types, decisions and schemes to rectify bad data. Several straightforward methods for some types of bad data were offered by Chapter 3. All others will require analytics to find and rectify them.

When some cases to a variable are bad, the question is what should they be? The cleansing process will lean on methods to replace bad cases, if necessary. However, the analytic is a core type for much more than replacing bad data with good. It is regression analytics to determine how strongly, if at all, specific operational variables are related to an outcome variable. Stated in the context of cleansing, the “outcome” variable can be the one with bad cases and we use the regression to estimate what they should be.

The chapters that explain two types of regressions are next in line to strengthen your career security. The analytics for relationships play in advancing operational effectiveness. For an outcome measure of performance, identifying the variables that matter most, or least, is powerful insight for assuring that your role is doing the right things right.

There are three mainstream regressions: linear, logistic and Poisson. The book will demonstrate linear and logistic. Chapter 7, *Relate Operational Variables to Outcome*, explains linear regression. Chapter 14, *Recover Lost Classifications*, explains logistic regression.

Regressions are a big step forward in your skills because they provide one of the most fundamental insights in operational capability. However, regression is not as simple as it would seem. We do not push all seemingly relevant variables into the model, run it and read the answer. Instead, there are considerable steps to choose and evaluate variables, and confirm fit: defined as how well the model accurately predicts the data.

Chapter 7 explains the process step by step. The same process generally applies to all regressions. I have never seen it mentioned in traditional texts that we can use the validation processes of regression to find outliers to the regression rather than only outliers to its variables. These are cases that the model would have never predicted and cases that have too much influence on the parameters of the model. Both are important insight to an operation because outliers may be telling us were to question and improve our operational processes and controls.

The difference between linear and logistic regression is their outcome variable. Linear predicts continuous numeric variables, whereas, logistic predicts classifications. Chapter 7 explains in detail the structure of the models and the distinction between them that rests upon the mathematics of prediction.

Chapter 14, *Recover Lost Classifications*, explains logistic regression while demonstrating a purpose for it beyond exploring relationships and outliers to outcomes. It

can be used to determine what an incorrect classification to a variable should have been. The strength of relationship of predictive variables to good classifications is used to cleanse the bad cases.

However, the purpose of the chapter is dual rather than merely to explain logistic regression. As the title suggests, the overarching purpose of the methods demonstrated is to recover lost classifications.

Accordingly, the chapter will also introduce and demonstrate the method known as naïve Bayes probability. We use naïve Bayes to determine from unstructured free-text variables, such as descriptions and notes, what a classification should be upon the probable occurrence of words in the text.

The next logical extension of your skills is to be able to prove there is a difference, or will be, as the consequence of change, improvement and enforcement of operational procedures and resources. Chapter 13, Prove There is a Difference, explains the body of analytics to make the determination vis-a-vis the operational situation. The analytics are a set of eight models from the types of two-means t test, ANOVA and multilevel.

Chapter 10, Through the Lens of Time Series, introduces another method to spot change through data and analytics skills. We look for patterns in variables presented in time series. However, our interest is not limited to change because overall we want to know what has happened over time.

Time series analytics are not to be confused with a line chart in Excel in which essential insights are lost in the simplicity. The chapter will explain by demonstration how to separate recurring cycles from the core pattern. It will explain how to measure the degree that one period reflects one or more previous periods. It will explain how the same principle is used to identify variables with a lead and lag relationship.

Just as importantly, we need to assure that a core pattern to a series is deterministic rather than a random walk. This is because a random walk can look deterministic. If not deterministic, we need to know to not devise actions or attempt forecasts as if it were.

Finally, in snaking through the chapters, we arrive at a final fundamental characteristic of any operation. It is the time it takes for each discrete core operational task to pass through the stages of the operation. Chapter 12, Elapsed Time Through Stages, explains by demonstration how to determine the statistics of retention in a stage and the chance of exit having remained in the stage for some duration. Just as important, the analytic enables us to determine which variables in the operation are most strongly related to retention and exit, thus, are the levers to reshape both characteristics of the stages that matter most.

Where to Go Next

The book has bound its scope to the de facto favorite mainstream analytics and, for them, what you need to know to build and interpret them. References are provided if you want to take the watch apart rather than merely know how to tell time. What is most exciting is that, with each chapter, your ability to reach into, understand and engage additional methods outside the book's scope will grow and solidify.

I was once told that to qualify as an academic textbook there must be assignments with each chapter. We could say that replicating the demonstrations in each chapter is your assignment. My personal experience is that a powerful solidifying exercise is to emulate each demonstration and explore its code command by command. However, there is an assignment we could make for each chapter.

First, you must ask yourself a question. For which roles of my operation do the chapter's skills and methods apply, and how and why?" Second, you must act. The assignment is to return and install new methods in your role, thus, equip your organization with actionable insights it has never had before. Third, you must share the knowledge. The assignment is to pass each of your newly acquired skills to others by identifying roles for which the chapter's methods could be impactful and helping the holders of the roles to increase their career security by bringing new value to the enterprise through their work in the role.

Welcome to the new world that opens to you as a new age worker with all career security appertaining thereto.

Data and Code to the Book

From literature and the internet, there are massive resources to explain all principles and methods using the R software and, for that matter, just about any software. The standing rule in the R community is that every provided explanation from any source must be accompanied with an example, code and dataset.

The book honors the rule by making the datasets and R code (scripts) to the chapters available to download from the webpage, <https://analytics4strategy.com/data-and-code>. You will need the datasets to emulate the demonstrations. The scripts are placed in the text for explanation by demonstration. However, since the code cannot be copied from the book and pasted into an R session, you have the option to copy and paste scripts into your R session.

Color and Format

The book is a complex manuscript and, thus, presents challenges to formatting and cost. The body of the book is a progression through text explanations, supporting figures, output exhibits and the R and Access code that generated the exhibits. In formatting the pages, all demand a degree of sequential rigidity for being laced together in an explanation by demonstration that can be easily followed.

One issue in formatting is that some output exhibits are visualizations that depend heavily on color to communicate the insight they are coded to give us. This is especially the case for visualizations that demonstrate the advance graphics of the ggplot2 package of the R software.

Unfortunately, the cost to produce the book in color is prohibitive to pricing. To keep the price reasonable, the book is produced in grayscale. The readers, of course, can view each graphic in full color by running the provided R code. Alternatively, the reader can view any visualization in full color at webpage, <https://analytics4strategy.com/book-look-inside>.

As it is said in data and analytics, “the devil is in the details.” When something does not work, it is often a tiny error in typing. Given the number of code blocks and internet addresses, hyphenation has not been used in formatting the text. Accordingly, readers need not wonder if a hyphen in a line of code or an internet address are from formatting or is as it should be. However, this policy occasionally creates some ugly lines in the text that hyphenation would eliminate. I ask for your forbearance.

In a few cases, the placement of output exhibits causes an over large empty white space at the bottom of a page. It is unavoidable given that some sequences must be honored with respect to the associated Access and R code upon which they are generated. Once again, I ask for your forbearance.

So, now onward we go through the fog and the dark where skill and education win over ignorance and superstition every time.

Richard G. Lamb, PE, CPA.

CHAPTER 1

Operational Availability is the Purpose

If we view a manufacturing enterprise as a system of money-making subsystems, it is easy to embrace the importance of asset management as the design, assurance and delivery of availability performance as its defining framework. This is because availability performance is one of the subsystems. The strategic significance is depicted in Figure 1-1 as the interplay of four subsystems to realize return on investment and strategic results.

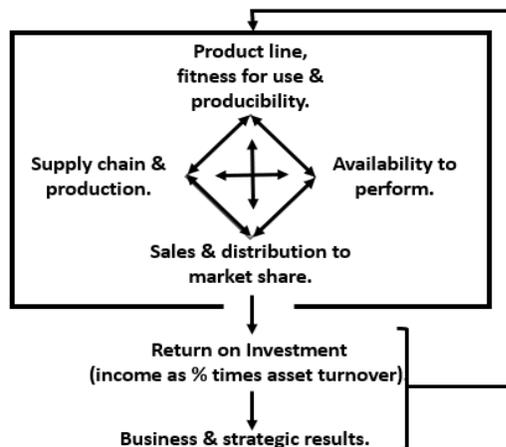


Figure 1-1: Four subsystems determine financial and strategic success.

First is the integration of the product line, and its fitness for use and producibility. Second is the integration of the supply chain and production operations. Third is the subsystem to make the firm's production assets available to perform at levels established as necessary for business success. Fourth are the operations to sell and distribute the capacity to the firm's market share.

The four subsystems almost completely determine the firm's competitive ability to realize income and productivity of assets by synergizing them in an overall competitive strategy. Furthermore, they must be synergized dynamically because short- and longer-term change is the reality for all four subsystems.

The parent firm's business results will suffer mightily if any of the subsystems are not appropriately designed and managed. This has an implication for firm- and plant-level management. Of their roles, a major one is to identify business initiatives that will most

2 | Chapter 1

advance the firm's ability to win returns greater than their industry's average. Initiatives for availability performance will be among the most attractive candidates.

1.1. Attributes of Availability Performance

The exploration of data-driven asset management begins by defining the attributes of its defining operational framework for availability performance and support. Accordingly, it is necessary to understand availability as a probability and how its constituent reliability and maintainability play in the probability.

1.1.1. Availability as Probability

Availability is the probability that a plant, subsystem or item will be in a state to perform a required function at specified standards of performance under given conditions when called for; assuming cost-effective support with respect to working conditions, processes and resources.

At this juncture it is important to contrast availability as a field rather than as a misnamed variable in the calculation of overall equipment effectiveness (OEE). As a variable in its calculation, OEE defines availability as the total time running divided by scheduled production time. The overall OEE computation is availability times performance times quality.

In contrast, availability as a field is concerned with the probability that that run time will be in accordance with a defined level or scenario of performance and support. By level or scenario, we mean that what qualifies as time running will be counted only if the standards established as performance, quality, support and more are met.

As a point of reference, the creators and thought leaders of reliability-centered maintenance (RCM) have serious issues with OEE and explain why in their writings [Moubray 1997]. The issues arise because RCM resides within the field of availability engineering and management and, thus, works on the same definitions that will unfold in the sections to come. Accordingly, the logic of RCM collapses under the definitions and equations of OEE.

1.1.2. Constituent Probabilities

The heading to this section could easily be "reliability with maintainability constitutes availability." This is because reliability and maintainability are performance characteristics which combine as availability performance. Therefore, let's define them more rigorously.

Reliability is the probability an asset will survive for some continuous, trouble-free period under planned operating conditions. Maintainability is the probability of returning

Cost effectiveness for facilities and equipment entail a mixture of expenses and expensed capital. The measure of cost effectiveness in both cases is largely a measure of too little or too much. However, expensed capital is locked in as depreciation expense. The direct expenses are largely for light, heat, etc. Whether there is room for sharpening the overall cost effectiveness of maintainability would be measured against effect on profit margin.

1.5. Data-Driven as the Means

Data-driven asset management is defined as using the firm's operational data to augment the experience and judgement of its operatives, managers, analysts and engineers as they plan, organize, conduct and control the functions, processes and resources of operational availability. The difference between data-driven and traditional asset management is that "possible matches vision."

Only by being data-driven is it possible to drill into the top-level factors of operational availability, discover and subject what matters to data-enabled analyses and reengineer for better outcomes. Only by being data-driven is it possible to confirm that the reengineered outcomes are truly shifting achievable availability upward and toward its peak while reducing the gap between achievable and operational availability. Only by being data-driven is it possible to assure that all is taking place that must take place daily, weekly, monthly and annually to reach and sustain greater and cost-effective operational availability. This is equally so for any type of enterprise operation rather than unique to asset management operations.

The book is timely because the information technology, software and knowledge making "possible match vision" have emerged since 2010. The operational systems we work with as role holders capture every piece of data that is inherent to the functions they support—creating the effect of the plant as model. Our organizations already make software (e.g., Excel and Access of MS Office) available to us as role holders with which to extract and join the data from our systems into the tables of data our systems cannot and never will be able to give us. Full-power analytic software, such as R, is available free without restriction. With it, we gain insights, and ask and answer questions of operational availability we could not before. Just as important, the how-to skills to work with data and analytics are readily available in literature and media.

The remaining chapters will explain and explore the principles and practices of the top-level factors to availability performance as data driven. As already mentioned, the chapters are relevant and applicable to almost any type of enterprise operation.

14 | Chapter 1

The next four chapters will introduce data development, applied statistics and software to reach data-driven asset management. The subsequent chapters will explain how they are woven into the top-level factors of availability performance. Everything will be presented with the application of the previously identified software because any asset management organization can use them to enact the data-driven practices; making what is explained immediately doable.

Bibliography

- Blanchard, Benjamin S. Logistics Engineering and Management. Fourth edition. Prentice Hall, Inc. 1991.
- Jones, James V. Integrated Logistic Support Handbook. Third edition. McGraw-Hill Companies. 2006.
- Lamb, Richard G. Availability Engineering and Management for Manufacturing Plant Performance. Prentice Hall, Inc. 1995.
- Moubray, John. Reliability-Centered Maintenance. Second edition. Industrial Press, Inc. 1997.
- Nowlan, F. Stanley and Heap, Howard F. Reliability-Centered Maintenance. Dolby Access Press. 1978.

Chapter 2

Data, Analytics and Software to be Data Driven

Data-driven asset management is defined as using the firm's operational data to augment the experience and judgement of its operatives, managers, analysts and engineers as they plan, organize, conduct and control the functions, processes and resources of operational availability. The difference between data-driven and traditional asset management and all operations is that "possible finally matches vision."

Only by being data-driven is it possible to drill into the top-level factors of operational availability, discover and subject what matters most to data-enabled analyses and reengineer for better outcomes. Only by being data-driven is it possible to confirm that the reengineered outcomes are truly shifting achievable availability upward and moving it toward its peak while reducing the gap between achievable and operational availability. Only by being data-driven is it possible to assure that all is taking place that must take place daily, weekly, monthly and annually to reach and sustain greater and cost-effective operational availability.

At this juncture it is necessary to establish an initial understanding of data, analytics and insight. Accordingly, the chapter will introduce what we are trying to do and the methods and software with which they are done. The remaining chapters will dive much deeper into what is introduced in this chapter.

2.1. Big Picture

Let's draw a big picture of the what data-driven asset management looks like. To begin, the section depicts a data-driven operation. Next, it will introduce the critical-mass knowledge, skills and software to reach the depicted operation, virtually without cost and up from the grassroots. And finally, the section will establish the basic definitions of data and analytics and, thus, separate the meaningful few from the distracting many.

2.1.1. What It Looks Like

Data-driven asset management was defined in the opening paragraph to the chapter. Figure 2-1 depicts a data-driven operation or process.

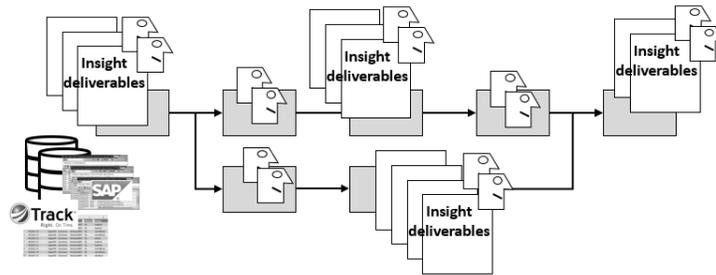


Figure 2-1: What a data-driven operation or process looks like.

What we see is that the processes of a data-driven operation are improved in a particular way. The judgement and experience of the role holders along the process are augmented with insight deliverables.

All activities to beget and manage operational availability flow along the processes of data-driven asset management. At some places along the processes, the best outcomes can only be realized when experience and judgement are augmented with insight deliverables. At each such place, the value of one or more insight deliverable is recognized, built and worked to realize the best of outcomes.

This implies a criterion for any developed insight. Each insight is justified if it makes a difference for the firm's financials and return on investment.

Also depicted in the figure is that the insight deliverables are built upon the data captured in the firm's operating systems. This is because the systems across an enterprise collectively capture every data point generated in the conduct of the collective operational processes. Furthermore, modern systems are designed to make it easy to retrieve their data as standard reports.

Also depicted is that some data may be captured in Excel tables. These tables can be positioned for ready access.

2.2.2. Critical Mass and Grass Root

Say "data-driven" to management and they cringe. They envision a complex, high-tech, deep-capital initiative and every horror that goes with it. However, this is a gross misperception. It has been propagated by purveyors and media as they speak of grand and glorious initiatives.

The reality is opposite to the perception. Almost every imaginable insight deliverable to an operation can be built and managed at the grassroots. This is because the "critical-mass" to becoming data-driven is not high-tech or new-tech. It is the exercise of modern-day knowledge, skills and software. The book is written to explain the critical-mass of data-driven asset management rather than stories of the grand and glorious.

Critical-mass is defined as the threshold knowledge, skills and software that must be in place for an operation to be fully, effectively and efficiently data-driven.

Critical mass has two qualifying characteristics. First, should they happen, the threshold knowledge and skills will travel to up-teched and up-scaled strategies. Second, up-teching from critical-mass will not practicably increase the power of the insight that is extracted from the operation’s data.

What is critical mass for data-driven operations is such that the most subordinate processes can be reengineered as a grass root initiative. This is because critical-mass rests upon a triad of software which is already normal to our work and organizations or which we have free rights to them. Figure 2-2 shows the triad to almost any given insight deliverable.

The data to all insight deliverables are readily accessible from the operating systems and Excel files that have captured them. When located, they are extracted from their sources and joined together in a super table with MS Access.

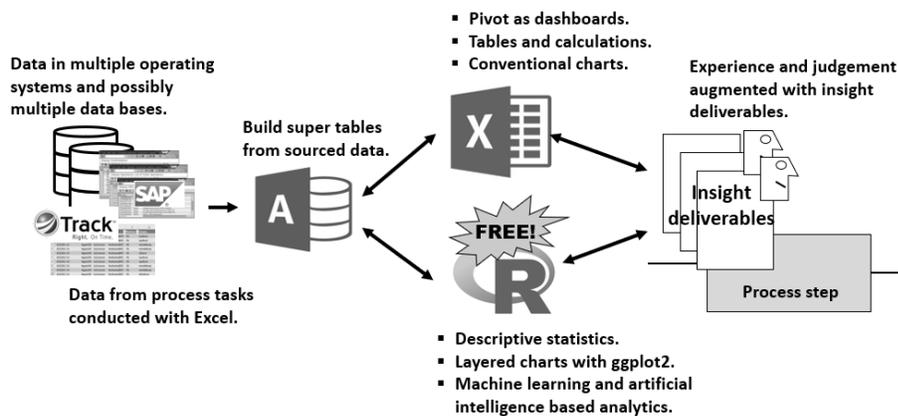


Figure 2-2: The critical mass triad of software to data-driven operations.

Then we will go down one of two paths for each insight deliverable. One is the path to Excel to build dashboards. Alternately, we may go down the path to subject the data to analytics using the free, open-system software known as “R.” By whichever path, there is an insight deliverable at the end of the trail.

This is good place to make a point with respect to grass-root and critical mass. Remember the expression, “Systems talking to each other?” There has been a great deal of progress over the decades toward the vision. However, we are still far from the prerequisite degree of systems integration needed for unconstrained data-driven asset management.

However, in Figure 2-2 we can see that there is de facto systems integration. The constraints to data-drivenness no longer exist.

This because systems “talk to each other” through their data bases. Reaching for data from any available source and building super tables constitutes pseudo systems integration for data-driven asset management. In line with the principle of grassroots, the integration is achieved with easily learned skills rather than big capital.

Of course, there are commercial alternatives for each software of the triad. However, we must be sure that the difference is more than just “prettier.” An organization may opt for a commercial alternative for strategic reasons. However, short of some strategic rationale, everything a commercial offering can do can be done by the triad.

The ramifications and distinction for critical-mass-supported insight deliverables are clear if we compare the software triad to the industrial internet-of-things (IIoT) for condition-based maintenance (CBM). Recall that CBM is depicted in Figure 1-4 as, *inspect items at regular intervals to find potential failures*.

The grassroots triad of Figure 2-2 entails no infrastructure beyond what is natural to any organization. In contrast, IIoT-supported CBM, as shown in Figure 2-3, requires infrastructure in addition to a firm’s existing infrastructure.

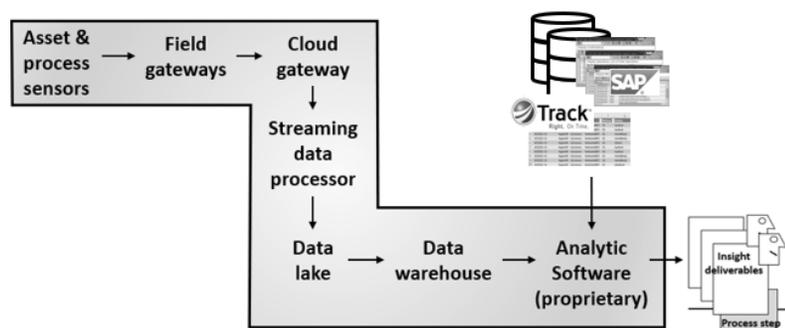


Figure 2-3: Insight deliverables built upon IIoT infrastructure.

New sensors are placed on the process or asset to continually monitor asset and process characteristics. The data of the sensors flow through new gateway infrastructure. Thence, there is additional new infrastructure to deal with the massive streaming, disparate data from the sensors and transforming them to workable structured form. Finally, there are new proprietary analytic software to conduct the analytics of the envisioned insight. Data may also be extracted from the firm’s operating systems and pulled into the analytics.

This is a good place to accentuate a point, In the domain of asset management, CBM is the only type of insight that may require a system akin to Figure 2-3. Therefore, it is

extremely important that we do not allow CBM to be mistaken by management as the essence, definition and overarching purpose of data-driven asset management.

In turn, it is also important that we do not allow IIoT-supported CBM to be mistaken as on-condition maintenance. CBM is one of four alternatives for an on-condition maintenance task to a failure mode—CBM, product quality, primary effects and inspection. The choices are made on worth with respect to safety, environment, operational capability and collateral damage.

Taken a step farther, IIoT-supported CBM is a choice to automate some or all aspects of a CBM solution. Consequently, the issue is the relative worth of IIoT-supported CBM in contrast with conducting CBM tasks with the many less grand and glorious offerings to achieve the same end.

Moubray's experience is that CBM is feasible for 20 percent of failure modes and worth doing for less than half of them. The four categories of on-condition maintenance are suitable for 25 to 35 percent of failures modes.¹

Because IIoT-supported CBM entails the extreme shown in Figure 2-3, it will rarely be a large-scale choice. It may be a worthwhile for a percent or so of cases. Consequently, it is important to not be distracted from the fact that almost 100 percent of insight deliverables to asset management can be done with the triad. For them, the worth is huge, and the cost is almost nothing other than the enterprise's commitment to learn new skills with standing software.

2.2.3. Essential Definitions

There is tremendous hyperbole around data and analytics. Ironically, the hyperbole may be a cause for managements' instinctive aversion to the discussion of data-driven operations. It has also caused tremendous confusion as an obstacle to become data-driven. To get beyond the aversion and confusion it is now necessary to narrow the hyperbole to the meaningful few definitions. With respect to them, the terminology of the hyperbole are expressions of the same thing but with different and flashy wording.

The essential definitions can be narrowed to four. They are data and big data, machine learning, artificial intelligence and algorithms. Many terms have come and gone over the last several years, but the four will remain as the language of data-driven asset management.

¹ Moubray, John. Reliability-Centered Maintenance. Second edition. Industrial Press. 1997. Page 155

Data and Big Data. Data and big data are distinctively different with respect to necessity, technology and organizational abilities. It is an important distinction. This is because big data entails high-tech systems and infrastructure, specialized skills and capital. Data does not.

We tend to think of “big data” in a colloquial sense from working with Excel. In the context of Excel, thousands or hundreds of thousands of rows are “BIG.” It is difficult to work with that much data in an Excel worksheet. Things are laborious once we get beyond a few hundred rows.

However, lots of data does not make it big data. Big data is the case in which data are prohibitively massive or unstructured. A professor of data science proposed a simple litmus test of data versus big data. It is big data if the data or the analytics of the data cannot be worked on our notebook computers.

Unstructured data include disparate types of data such as streaming sensor data, e-mail, document, video, photo, audio and webpage. This is compared to the structured alpha and numeric data that is predominate to operating systems. The purpose of analytics of unstructured data is to transform them to be structured data with which other analytics can be conducted.

The types of data analytics conducted in either arena are the same. The type does not decide whether it is data or big data. However, the important notable reality for data-driven asset management is that there are very few imaginable needs for big data.

Let’s look at a situation that is not big data, but the definition of unstructured data may cause us to assume it is. Other than the massive, streaming data of CBM systems, the only unstructured data in most operating systems are the free-text variables of work order descriptions and notes. Text mining analytics may be used to classify the work orders by the failure mode that triggered them. Remembering the professor’s litmus test, although free text is unstructured, it is possible to conduct text mining on our notebook computers.

The IIoT-supported CBM of Figure 2-3 is an example of big data. Streaming data is massive because the data points are almost infinite in number. Additionally, for some types of monitoring, the data is not structured data and must be transformed by analytics before it can be subjected to the ultimately planned analytics.

Therefore, when the expression “big data” is casually tossed about, we must ask an inspectorial question. Is it hyperbole or big data? If it is truly big data, then we cannot take the grassroot strategy as it is otherwise possible with the triad of software.

And once again, a very important point must be repeated. Almost never will the insight deliverables to asset management entail big data. This is a fundamental reason that data-driven asset management can be grown from the grassroots.

Machine learning, artificial intelligence and algorithms. We need to clarify the interrelated terminology of machine learning, artificial intelligence and algorithms. We will use the two-variable regression analysis shown Figure 2-4 as a frame of reference.

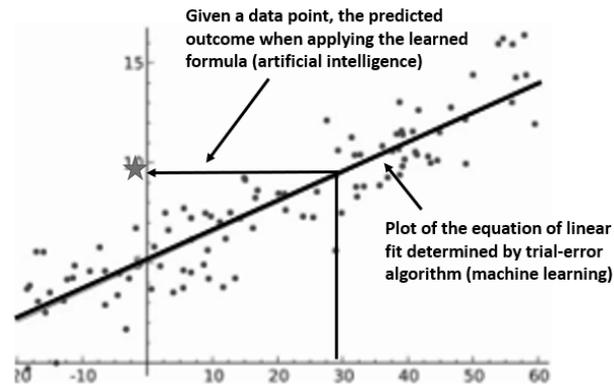


Figure 2-4: A two-variable regression demonstrates machine learning, artificial intelligence and algorithms.

We have all at some time touched regression modeling. However, the concept is the same regardless of the type of model and the number of predictor variables placed in the model.

Machine learning (ML) takes place when we feed the predictor and outcome variables to the regression. The gut algorithm conducts a trial-and-error calculation until “learning” the best fit and returns a formula of an intercept and slope coefficient. The predictive and outcome variables are the axes and the points are their cross-plots. The line fit to the plotted points is the learned outcome of the algorithm.

Most often our interest ends with the returned coefficients for each variable (one in this case) and associated inferences for how strongly, if at all, the predictor variable is related to the outcome variable. Confidence interval to the coefficient will also get our attention. In contrast, artificial intelligence (AI) feeds new cases to the fitted model to predict outcomes upon the “learned” formula.

AI does not distinguish the model. All types of models entail machine learning, and most can be deployed as AI.

When the model is to be deployed as AI, the learning process for some types entail an additional stage of analytics. A portion of the original dataset is held out from the machine learning stage to be a test set. The remaining portion is fed to the model for learning. The test set is subsequently fed to the learned model to evaluate how accurately the “trained” model estimates or classifies the actual outcome of each case in the test set.

If accuracy is acceptable to the intended use, the model is deployed to serve its purpose—augment human experience and judgement. Results better than 85 percent are typically considered acceptable. This is why we must always think of AI in terms of “augmenting” rather than “supplanting” experience and judgement. Hyperbole may lead us to unrealistically expect “supplant.”

2.2. What and Why of Access and “R”

We are all experienced users of Excel. In contrast, experience with Access is unusual and awareness of “R” is rare. Consequently, now is the time in the explanation of data-driven asset management to introduce the means to move data from its source to its use, Access and the analytic core, R.

The section will be an overview of the Access software and R. The discussion of “R” will be extended to introduce what the book calls layered charting. Written for self-directed learners, the remaining chapters will dive deeper into both software in the context of explaining by demonstration data-driven asset management practices made possible by them.

2.2.1. MS Access for Super Tables

There can be no dashboards and analytics without a table inclusive of the variables they are to be built with. This book will call them super tables in contrast to the sub tables that are combined in the super table.

Super tables can be built in “R” with the structured query language (SQL) capability and with additional coding in the rare occasions when it is necessary to go beyond the ability of SQL. The variables from their sources are imported into Excel Pivot and “R.”

However, this requires substantial skills with SQL and “R” coding. That is why MS Access is one of the triad software. The skill requirements shrink to minor and largely entail skills that have become normal to most of us. Therefore, the more practical strategy is to learn how to build tables in Access and then import them to “R” with the “read” function.

This section will introduce the concept and process of building super tables. The Chapter 3 will go deeply into the how-to of super tables. Subsequent chapters will strengthen the readers skills with super tables through the explanations by demonstration of data-driven asset management practices.

Figure 2-5 is a pictorial summary of what is to be done. Any insight deliverable requires all needed variables in a single table. The table must be formatted with variables

as columns and cases as rows. There are no row titles or space and sum lines. All tables in the figure meet the standards.

The figure shows the typical hurdle to building insight deliverables. The needed variables exist in three sub tables. They must be brought together in a single super table. Otherwise, asset management is divided and conquered by the firm’s operating systems.

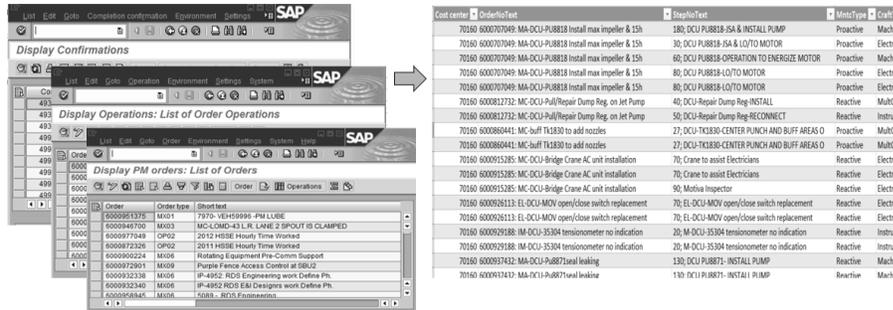


Figure 2-5: Three sub tables combined in a single super table.

Another perspective is that the needed super table does not, cannot and never will exist in any operating system. Furthermore, for those who say, “I would do it in Excel,” there are four points to make.

First, building the envisioned table in Excel is too laborious and limited to be practical. Second, it was said earlier that very quickly a block of data becomes “big” relative to working with data in Excel. Third, Excel is limited to 1.1 million rows of data. Finally, on a personal level, we need to stay modern if we want to stay relevant.

There is a process for building super tables. It is shown in Figure 2-6.

The first step is to locate where the variables of interest reside across the enterprise. Once found, the second step is to identify the standard reports by which to extract the data from their resident operating systems. The third step is to bring them into a query software such as MS Access.

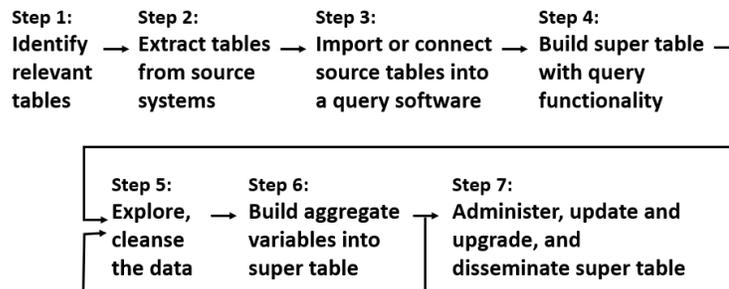


Figure 2-6: The process to build super tables.

24 | Chapter 2

Figure 2-7 shows the action to be taken. The tables of Figure 2-5 are standard reports that, once recognized, are imported into Access. The figure also show that two other non-system tables have been pulled in the query. They were built to make it possible to categorize and clarify the data in ways that were not, and never will be, configured into the home systems.

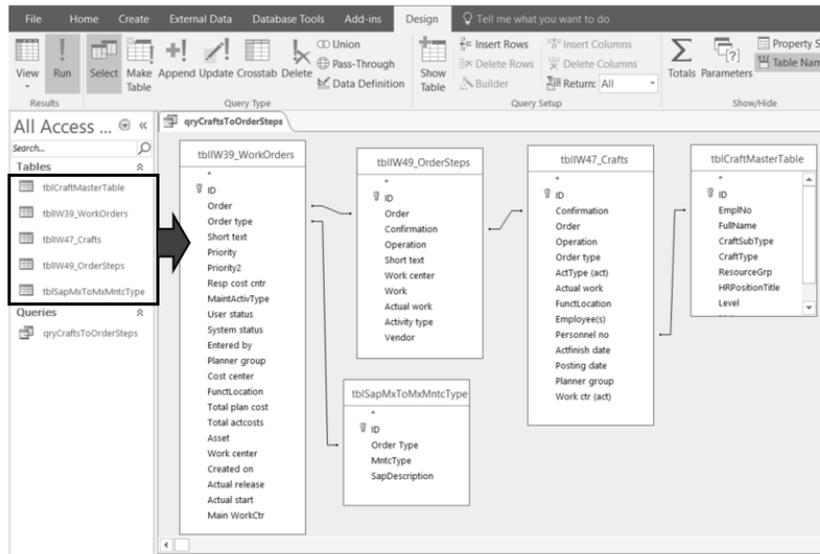


Figure 2-7: Tables imported to the query software and joined as a single table.

Notice the lines between the tables that were pulled into the work area. The line symbolizes that the tables are joined in a massive raw table. Each pair of tables is joined by a unique identifying variable they have in common.

The super table is built in step three. Variables are selected to be in the table as shown in Figure 2-8. By click and drag all desired variables from the joined tables are pulled into the evolving super table That is the purpose of the design grid at the top of the figure.

A lot goes on in the design grid. However, we would find that almost all of what is done draws upon the skills most of us have accumulated throughout our working lives. When the run icon (not shown) is clicked, the super table shown in the bottom part of the figure is generated.

Until confirmed, it is never assumed that the data pulled into the super table is accurate and complete. The next step is to explore the data for issues needing a treatment strategy.

Most times there are simple solutions such as translation tables to be introduced in Chapter 4. In other cases, the table may be pulled into “R” for cleansing with machine

learning and artificial intelligence. For some insight analytics, there is a choice to omit bad data after evaluating the ramifications to the subject insight.

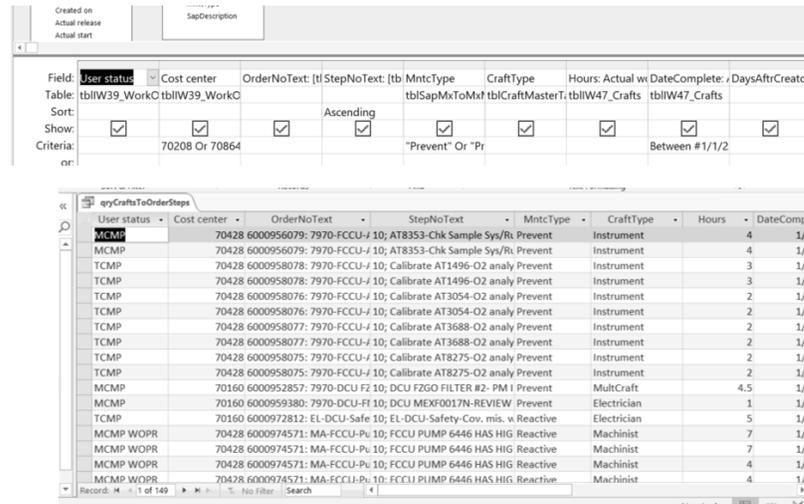


Figure 2-8: Variables pulled into the super table molded for insight.

The sixth step is to build aggregation variables in the super table. The idea is to create new variables in the table. They are totals, counts, averages, standard deviations, , min-max and first-last for groups created upon a set of predictor variables.

All sorts of insight variables are possible when the super table is extended with aggregations. An example is to generate the computed variables for workload-based budgeting and control on actual versus budget. Dual-dimensional budget and variance will be a subject of a later chapter.

Steps two through six are done with standard query language (SQL). The only exception is that some types of cleansing require data analytics. This suggests that one hurdle to data-driven asset management is to grow SQL skills across the organization.

How the hurdle is jumped is the reason for MS Access in the triad of software. This is because SQL runs in the background as we work at the foreground with the skills, we all have as modern workers. Furthermore, besides already part of the Microsoft Office software, it is arguably the easiest of all query software to learn and work with.

Another advantage is that Access stays close to how the sausage is made rather than be hidden from us inside a black box. Accordingly, what is done in the foreground somewhat mirrors the clauses of SQL.

The final step recognizes that any one super table is likely to be built to serve multiple insight deliverables and ad hoc analyses. Therefore, the final step is to form one or more

processes to manage each super table through its build and refine, update and disseminate stages. The process may be owned by the primary beneficiary or by someone with the role of building and administering the table on behalf of all players across and beyond the asset management organization.

This brings another point to the surface. The SQL code, automatically formed in the background as we work in foreground, is available as a view option. Consequently, the super table can be distributed as a txt file just as for an “R” script. The recipient can paste the text in the SQL view and run it. In turn, the recipient can modify the super table in the design view.

There is a partnership between “R” and Access in the triad. At times we may want to formulate variables or reshape the table in ways that are beyond the ability of SQL. When more is needed, the super table can be built in Access, pulled into “R” and powered up. As previously mentioned, we may also want to subject the table to cleansing analytics that are beyond the ability of SQL.

Some analytics are built with data that must be shaped specifically to a model’s algorithm. When the case, the bibliography literature explaining the model will, of course, introduce and explain the “R” functions to reshape the data.

2.2.2. “R” for the Analytic Core

Something fascinating has happened. Software have become available that are open systems and freely available to any individual to install on their computer and any organization can make part of its IT system. They are also being pulled into commercial software.

These software are not being offered under some sales strategy such as a “trial period” and a “free” compared to a “professional version.” Nor are they weak compared to their strongest commercial competitors.

The analytic software “R” is one of such offerings. A testimonial to its strength and unrestricted accessibility is that Tableau and Power BI have seamlessly incorporated “R” into their offerings rather than develop a comparable proprietary capability.

“R” can be downloaded and installed from the website <https://www.r-project.org/>. There are YouTube videos explaining the simple download process which takes less than 15 minutes.

Other than full-powered, open and free, there are additional reasons that make “R” critical-mass to data-driven asset management. The pinnacle aspect is that through “R” the asset management organization gets its capability for descriptive statistics, layered charting, data cleansing, and machine learning and AI.

“R” is actually a collection of thousands of “packages” for working with almost every imaginable analytic. Every analytic is conducted with a “function” and its associated “arguments.” We identify the packages and functions we need to conduct an envisioned insight deliverable. Thence, we do not write code—we type or paste the function code in our R-session as a go-by and adjust it to purpose.

Below is an example of a package, function and arguments. The `lm` function for linear regression is available from the “stat” package. The arguments are designated within the parenthesis of the function. The function and its arguments are. . .

```
lm(formula, data, subset, weights, na.action, method = "qr", model
   = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
   contrasts = NULL, offset, ...)
```

Notice that a function is set up by the choices we make for its arguments. Explanations and examples of the options are readily available from the internet. However, we rarely touch most of the arguments because the defaults are typically the desired option. Accordingly, the shown code may reduce to `lm(formula, data)`.

The packages are created and maintained by individuals and organizations around the world in accordance with standards of creation and care. Each package is accompanied with a full explanation of its functions and arguments. Additionally, the explanation includes examples and data with which we can see them in action and experiment.

An extremely important characteristic of “R” is that online support is highly evolved, vast and free. However, we are not limited to online sources. Literature explaining the principals and methods of statistic and analytics with “R” is plentiful. As they explain the principles of statistics, they demonstrate them with “R.” As they do, the texts additionally explain each line of code. Consequently, the texts serve concurrently as texts on analytics and the “R” software.

The bibliography to the chapter includes the best available text for every type of analytic. “Best” is defined as a practical working depth explanation; able to be a go-by. This is compared to deep discussions of underlying theory and mathematics.

This book parallels examples from the bibliography literature rather than examples from asset management operations. The generalized examples will have an obvious go-by fit to the aspects of asset management being discussed. Because generalized examples can be go-by’s, it is much more important that the readers have at their avail a full-depth explanation of the analytic rather than a domain-specific one.

The rationale of bibliography-paralleled examples is demonstrated by the seeming simplicity of the previous two-variable regression analysis. Setting up and interpreting the model is only a tip of the iceberg. Below the surface there are many matters of selecting variables and validating the model. Covering the full depth of every analytic would make the book a 2,500 or so page venture far beyond anyone’s willingness to write or read.

Chapter 4 will present and explain how to work with “R.” The explanation of this chapter will be to give the reader the lay of the land. What is explained in the Chapter 4 will be extended in subsequent chapters as R is put in play for data-driven asset management practices.

Figure 2-9 show the primary three windows of “R.” Like all software they can be moved and sized.

At the left side is the console window. In it, we can place all commands and get a continuation or output upon pressing “enter” at the end of each line of code.

The center section is the script window. We are not required to use it rather than the console. We do because the window allows us to work with code in a more friendly, flexible manner. Another reason is that what is coded can be saved as a script file.

The difference between the script and console windows is that commands are typed in the console for each occasion, whereas they can be edited and run repeatedly from the script window. Otherwise, the difference is that the output of running the script appears in the console and graphic windows but never in the script window.

If the script or console code contain commands for graphic outputs, they will appear in the graphic window. Obviously, it is the right-most window of the figure.

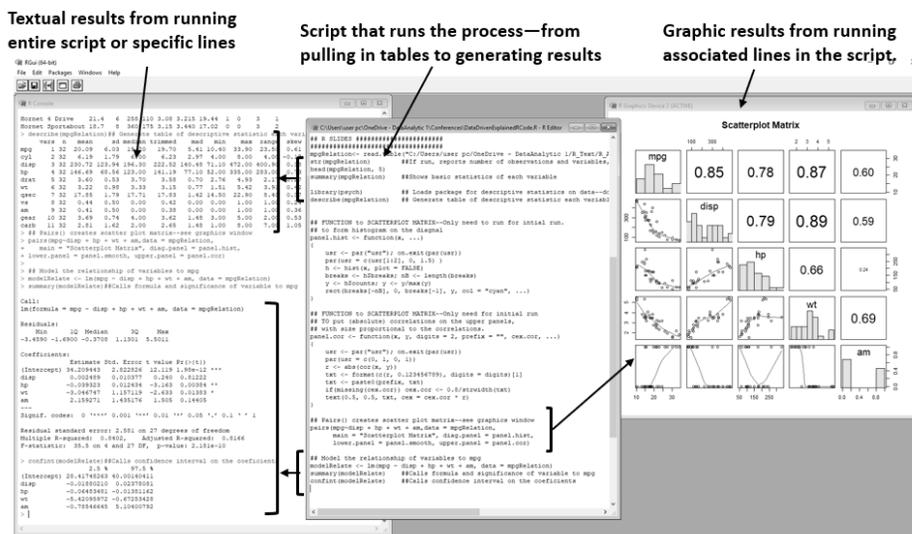


Figure 2-9: Console, script and graphic windows of R.

The code of the script window can be commanded to run in its entirety or by highlighted lines. Just as valuable, the script allows a solution developed by one person to be distributed to others as a script file. Let's note here that if the script file name is extended with .txt, it becomes a text file, able to be read and edited as a Notepad file.

Although using code may appear to be geek-like scary, coded software has a big advantage over the graphical user interface (GUI) software we have grown accustomed to. Dispersing a GUI-based solution requires a lengthy instruction document and all the difficulties that entails. Scripts can be dispersed as a file of code with explanations placed in the code. Just as important, the recipient does not need to follow a documented instruction as one does for GUI. Instead, the user only needs to load and run the script.

The first line of shown script code is a function that imports the data to the planned analytics. Beneath that, other functions explore and inspect the data in text and graphic form. At the bottom are the functions to the analytics we seek. The script's content will be fully presented and explained Chapter 4.

2.2.3. Layered Charting

Now is a good time to introduce an important breakthrough to insight: layered charting. Layered charting is a big leap in the ability to extract visual information from data. The "R" package to create layered charts is ggplot2.

Most data are still visualized with types of charts invented as far back as the 1600s and no more recently than the 1800s. Now layered charting allows the visualization of data as information in almost endless ways.

The book defines layered charting as presenting information in layers. The difference can be seen in Figure 2-10. Traditional and layered charting are shown side-by-side.

The matrix of the figure summarizes the differences. For one, traditional charting is limited to the named types such as cross plot, bar and column, pie, line, spider, etc. The layered charts have no type or name because they are named by the insight they give.

Traditional charts are limited to the two variables of the axes. In contrast, the number of variables that can be pulled into a layered chart is only limited by practicality.

Traditional legends are limited to the categories of the charted variables. Layered charting can use legends as variables.

Another important differentiation is the capacity for a large number of data points. Visual granularity is lost when there are too many data points. Layered charting offers many ways to regain granularity. Some are shown in the figure.

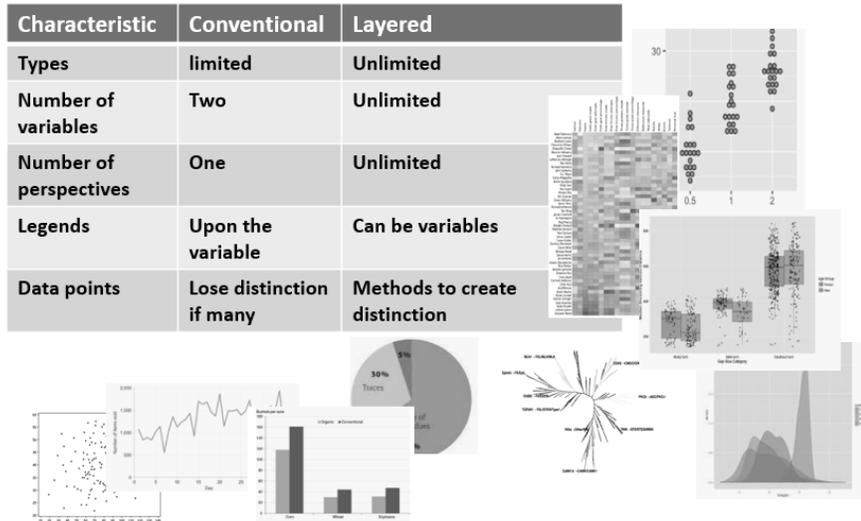


Figure 2-10: Contrasts between traditional and layered charting.

Figure 2-11 is an example of layered charting built upon the variables by which cars are evaluated and compared. It demonstrates possibilities for presenting the KPIs of asset management.

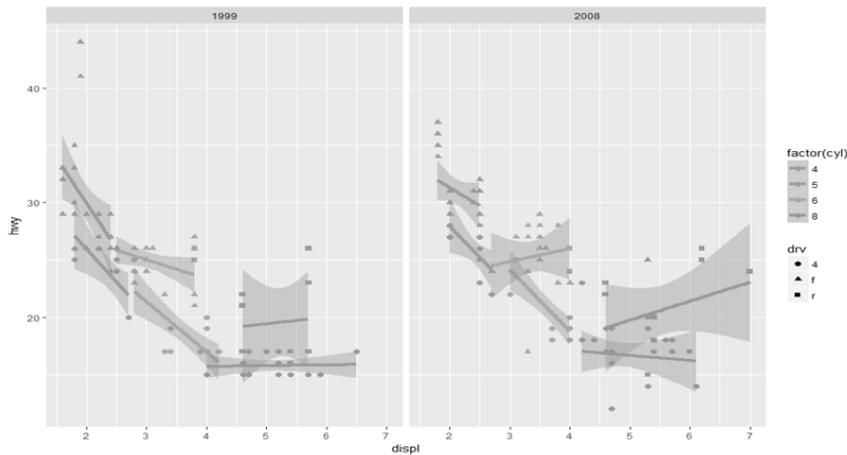


Figure 2-11: Layered charting to present measures of performance.

In the figure, we see the interplay of five variables: highway mileage, displacement, cylinders, drive and years. All are related to the axes of displacement and mileage. Accordingly, we see a cross plot relationship of the two variables with a linear fit and confidence intervals to the fit. Traditional charting would show a negative relationship—mileage falls as displacement increases.

However, the shape of the points would suggest that there is more to the picture. A traditional two-variable cross plot and linear fit may be misinformation.

The cylinders and drive are layered into the chart using legends as the method. A new picture emerges. The relationships vary with the added variables. In some clusters the relationship is positive rather than negative. When year is layered into the chart, it is revealed that the relationships have changed with time. One relationship has even changed direction.

2.3. Insight Deliverables

At the beginning of the chapter insight deliverables were depicted as woven into the processes of asset management. The principle is that the outcomes of the processes would be better than if without the insight.

There are four types of insight deliverables. As named by this book, they are system reported, recountive, know-thy-data and modeled.

Each will be introduced in this section. Subsequent chapters will further expand upon them in the context of explaining how to bring specific sub processes of asset management to be data driven.

An observation. It is easy to envision data driven as exotic and grand. However, the majority of insight deliverables are recountive and know-thy-data. This is an extremely important point because the skills to build and work them are of the type that are easily and quickly absorbed and dispersed. Consequently, once we know what we want to achieve in the effectiveness and efficiency of a process, the recountive and know-thy-data deliverables are the easiest of the challenges for getting there.

2.3.1. System Reported and Recountive Insight

System reports are the insights that our systems have been configured to give us. The number of asset management organizations for which system reports are the extent of insight is shrinking.

The number of organizations that function with system reports and recountive insight is growing. This is evidenced by the growing use of Excel Pivot in contrast to spreadsheets in which the data and report are fused. Chapter 8 will introduce the problem and solution to fused data.

In contrast to modeled insight, recountive insight is built directly upon the data of operational systems rather than processed through analytics. Consequently, recountive

insight limits the organization to asking and answering questions of who, what, when, where, how much and indicators. In other word they recount history.

As shown in Figure 2-12, recountive insight is packaged and delivered with Excel Pivot or somewhat prettier commercial alternatives. If the data to the Pivot are super tables, the insights are beyond what can be revealed from individual system reports. This is because we are able to slice-dice-drill for insights that not visible in system reports, until their data are joined in a super table.

We can power-up recountive insight by including layered charting. Excel Pivot and commercial dashboards are largely limited to traditional charting. The combination of conventional and new-age visualization is a good example of the critical-mass analytic software, “R,” making it possible to build data-driven processes up from the grass roots.

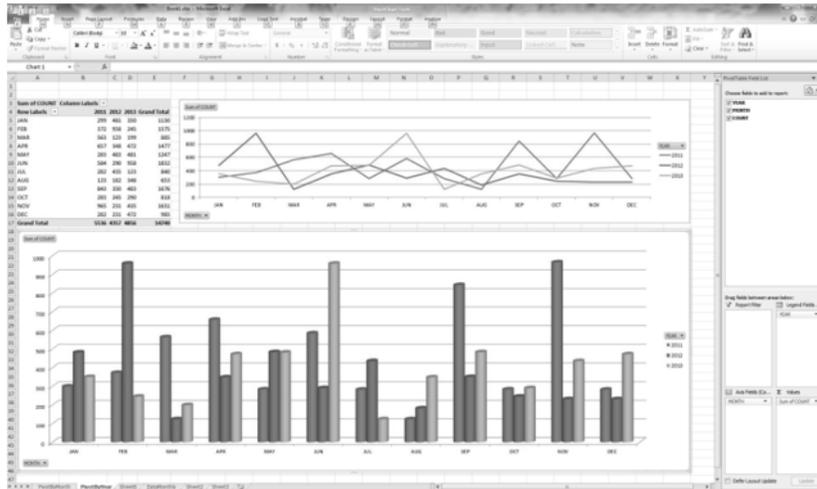


Figure 2-12: Recountive insight deliverables with Pivot of the triad.

2.3.2. Know-Thy-Data Insight

One-time and periodic inspections of process data often beget deep insight. One reason is that the persistent compliance to operational processes is readily confirmed through its data. Another reason is that an initial and unfolding inspection of data often reveals hidden realities, findings contrary to common belief and lurking misinformation.

Know-thy-data insight is the case of a table of data explored from descriptive, graphic and statistical perspectives. Figure 2-13 shows some of the possibilities generated with “R.” Query-type probing with Access and Pivot are also powerful methods to seeking know-thy-data insight.

The upper right is an example of data variables summarized descriptively when the super table is fed to the summary function of “R.” For each variable we can inspect the min-max, median, mean, quartile, categories, counts and number of missing records.

The lower left is an example of the data presented in graphic form. As previously mentioned, the possibilities for graphic exploration are intriguing because “R” brings non-traditional graphics to the data.

The upper left is a combination of numeric and graphic insight. In it there are eight insights to each variable and the correlations between them.

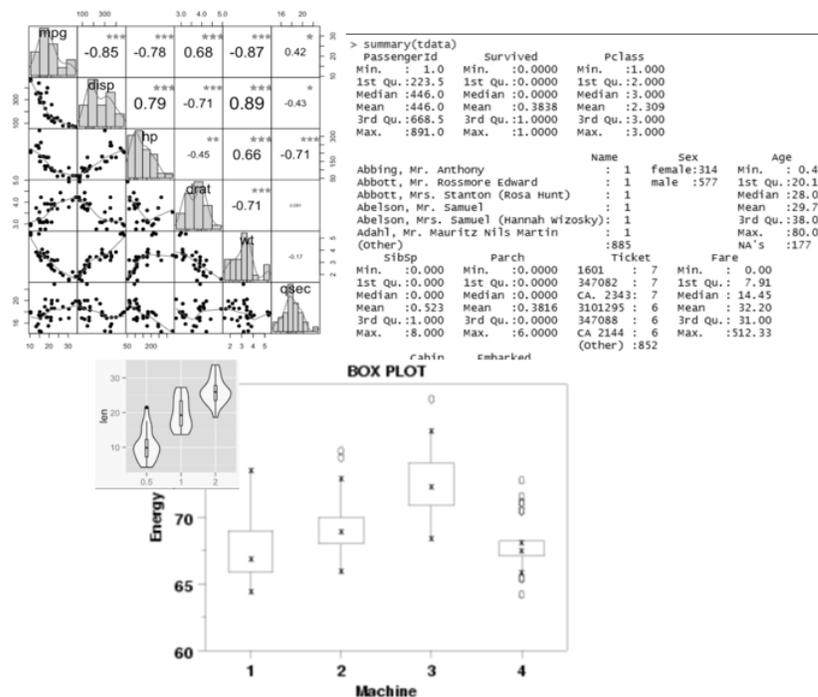


Figure 2-13: Examples of know-thy-data generated with “R.”

2.3.3. Modeled Insight

The final category of insight deliverables, as named by this book, is modeled insight. Insight is gained as the data flows through analytic models for machine learning and AI.

Consequently, and in contrast to recountive insight, we can ask and answer six additional types of insights—relationship, difference, time series, duration, classification and apparenity.

Relationship Insight. Which asset and process variables are most strongly related to a performance of interest? Many of us have asked and answered such questions with linear regression.

Figure 2-14 shows that there are three types of regression models. They are linear, logistic and Poisson.

A model can entail multiple predictor variables. However, note that single predictor models are shown in the figure so that we can see the underlying fitted shapes of the three outcomes. The shapes hold with two or more predictor variables but are not graphically depictable.

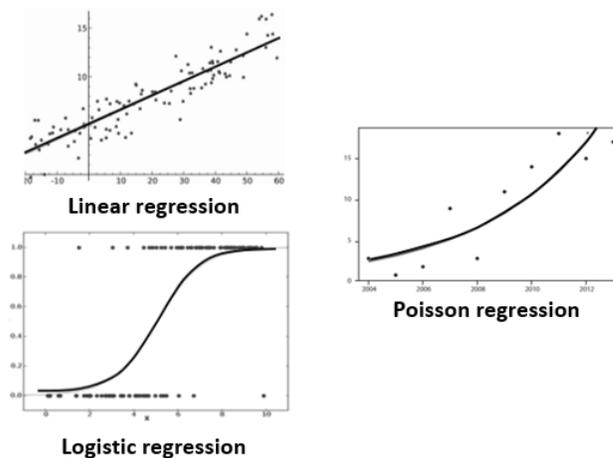


Figure 2-14: Three types of regressions for relationship questions.

The three types could be subjected to the same sets of predictor variables. The difference is the report-out we seek from the outcome variable.

Linear regression deals with numeric outcomes such as cost, hours, productivity and numeric indicators. The fit is linear, and its outcome can range from positive to negative infinity.

What if the score is not linear and dually-infinite? What if the fit falls between 0 and 1? In operational processes, many behaviors are binomial—e.g., working or not working. Many others are multinomial—e.g., option 1 or option 2 or option 3.

Logistic regression reports out as the probability of binomial or multinomial outcomes—strength of conviction. For example, in a wrench study, the model can predict the likelihood of a “found-working” outcome. Another is to predict the likelihood an entry is out of compliance with what should have been recorded.

There may be missing or suspicious numerical or classification entries in the super table. Linear regression, as AI, can be used to impute missing entries. As AI, logistic regression can flag suspicious classifications and cleanse bad classifications.

The Poisson regression models occurrences. They report out as counts or rates. For reliability or process compliance they are respectively probabilities for the number of occurring failures or noncompliances. Rate ties failures and noncompliances to time interval, area and other groupings.

Often there is money in finding outliers. They can make the insights based on them to be misinformation, thus, a payoff upon spotting them. Or they may point us to some phenomenon that, although hidden, reveals intriguing ramifications. The regressions allow us to apply sophisticated methods to find outliers.

Difference Insight. How do slice-dice combinations of asset and process variables comparatively effect a performance of interest?

A whole window of inquiry opens once we discover a fundamental truth. Because two or more numeric outcomes are different does not mean they are different. For example, just because a KPI changes after an improvement program has been implemented does not mean a practicable difference has occurred.

We will use Figure 2-15 to explain the overarching concept of analysis of variance (ANOVA) to engage in this extremely meaningful type of questioning.

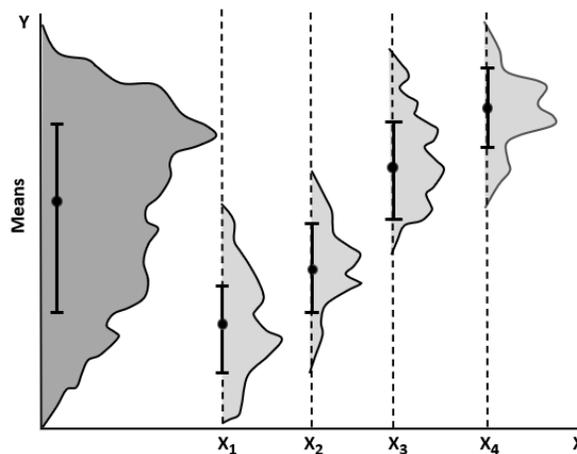


Figure 2-15: The concept of analysis of variance (ANOVA)

Shown at the left, the outcome variable in the model has its statistical mean and confidence interval to the mean. In statistics this is called the base or null model.

What if we grouped the data upon one or more predictor variables? Shown in the figure is a model with four outcome groups to a predictor variable. They occurred out of the choice of predictor variables. Each group has its own mean and confidence interval to the mean.

The big question is which pairs are truly different even though their means differ? Do the confidence intervals to the means of the pairs overlap? If so, they are only different due to sampling error.

It would seem that we could answer the question by simply making pairwise determinations. The problem is that what is called “family error” is attached to each pairing. This makes it more likely we would conclude there are differences when there are not.

An example of family error makes the point against simple pairwise analysis. Imagine that the confidence interval established for the analysis is 95 percent. However, the true confidence for each pairing is an unacceptable 74 percent (0.95^6).

What’s called the two-mean t-test is used if, for example, Figure 2-15 were only a breakout of two groups. Comparisons between three or more groups must be evaluated with ANOVA-type and multilevel modeling rather than pairwise comparisons.

The determinations are made with post-hoc adjustments and contrasts of variance. With post-hoc adjustments the confidence intervals are made statistically wider (conservative), thus, making it less likely to wrongly accept a pairings as different. Alternately, contrasts of variance evaluate groups by the comparative portion they explain of the total variance of the base (left most) model of Figure 2-15.

Finally, we should note that there are circumstantial variations to ANOVA. They are one-way and multi-way ANOVA, ANCOVA, repeated measures and mixed multilevel, and MANOVA. All but MANOVA will be expanded upon in later chapters.

Time Series Insights. What are the components that underlie the summary-level-only history that operating systems are limited to providing? Figure 2-16 shows the primary components of a time series from which the question seeks to gain insight.

First it is necessary to extract “cycles” from the summary plot in the top panel. Upon doing so, we can inspect the pattern of cycles. Are they staying steady with time or are they changing? Seen in the bottom panel, the cycle is swinging wider with time beginning with the step up in the trend.

Cycle can be of great interest in a time series. Let’s take the occurrence of notifications for needed corrective maintenance. Intuitively, we would expect the notifications to occur randomly. Are we instead observing yearly, monthly and weekly cycles? More importantly, is there a message for asset management in the cycles.

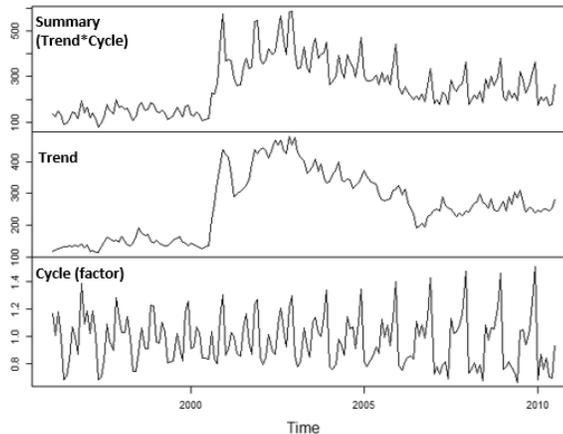


Figure 2-16: Cycle removed from the summary level to reveal the trend of the series.

This leaves the “trend.” We are interested in its shape and its message for asset management. However, it is necessary to confirm that it is what we think it is. Is it deterministic, random walk or random?

Random is easy to spot. However, a random walk may appear as deterministic. Rather than deterministic, a trend’s shape may reflect what is called autocorrelation by which a period is influenced by one or more previous periods.

As an example, let’s take a study of actual versus planned craft productivity per scheduled work order. After an initiative to improve job plans and field supervision, is the observed trend in productivity a deterministic one? Alternately, is the trend only reflecting autocorrelation rather than a deterministic input-output relationship?

It is also necessary to question if there is autocorrelation. in the variance to the series. Undiscovered, we may otherwise be acting on misinformation. This is because autocorrelation will report out a different than true confidence interval for the series.

The principle of autocorrelation has an additional and exciting ramification. The same mathematics can be applied between variables: cross-correlation: to find lead-lag relationships. The trend series of one variable demonstrates a pattern that appears later in another. For example, can we see a pattern in corrective maintenance workload that follows a pattern in plant capacity utilization?

A final point. Prediction is often mistaken as forecasting. However, there is a definitional difference. The distinction is analysis within and beyond the data.

Prediction is what has been explained to this point—within the data. Forecasting uses the findings of prediction to project beyond the data and into the future. The future is being forecast upon what prediction has revealed of the past.

A forecast is built by projecting the trend into the future. The cycle series is also projected into the future and attached to the projected trend. An appropriate confidence interval is placed on the forecast after adjustment for autocorrelation. Of course, we must first question if we can safely assume that the influences of the past will hold into the future.

Finally, we should note here that there are a range of models with which to work with time series. They are Holt-Winter, series regression, ARMA and ARIMA.

Duration Insight. What is the probability an asset or process condition will hold for some time and then what is the probability the condition will end? To answer the questions, Figure 2-16 shows the three plots to survival and hazard analysis: survival, hazard and cumulative hazard.

As point of reference, the hazard function is the life curves of figure 1-3. Therefore, a good example is “on-condition” maintenance. Having survived to the time of inspection, the inevitability of a functional failure event may be revealed. Once revealed, cumulative hazard will give us a sense of the risk of a functional failure during the time allowed for the orderly corrective maintenance process—mean time to maintain.

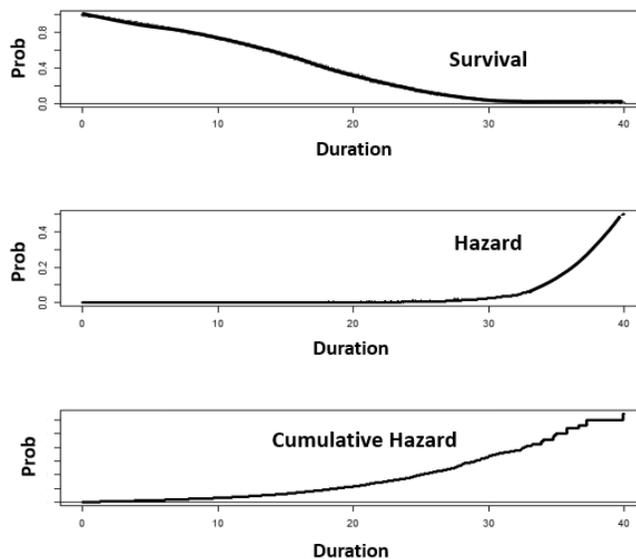


Figure 2-17: The three perspectives of duration analysis.

If the cumulative hazard is too great, we will find ways to shorten the interval of the corrective maintenance process. Duration questioning will play heavily in reengineering the interval. Each stage along the critical path of administrative, logistic and maintenance

activities will likely be subjected to duration questioning. The insight will be the shape of the curves—what are they, what can they be and what will we reengineer them to be.

Let's make a side point. The data to reengineer the sub intervals of the maintainability interval are complete and accurate. This is because most computerized maintenance management systems automatically collect status history as a work order passes through its stages from notification to completion.

There is another point to make. The mathematics of hazards make it possible to determine which contextual variables influence the patterns of Figure 2-16. This is exciting because we may find that it is possible to reshape the hazard curve and, in turn, the survival and cumulative hazard curves.

We should note here that there are a range of models with which to work with duration questions. They are Cox regression, Cox proportional hazard, Cox mixed-effects, cumulative incidence, proportional hazard regression, Weibull and Crow-AMSAA.

Classification Insight. Inherent to the data of any operation are the many classifications captured in its process management systems. They are the categorical variables in operational systems. If we comply with our operational procedures, the consequent quality of all classifications is almost unavoidably perfect. In turn, we set off a virtuous cycle of data-driven analysis, making and execution of strategic, tactical and real-time decisions.

The cold reality is that compliance is problematic for classifications that are not automatic within a system. Especially for those that entail some degree of still less than fully informed conclusions. The classifications that are most quick to mind are failure modes and codes. Any degree of sloppy misclassification is doubly problematic because so much of reliability engineering depends upon analytics to which the classifications are the essential feedstock data.

What has been lost by incorrect or omitted classification can now be found. There are data analytic methods to recover the classifications. This book will present the de facto two favorite and most practical methods. They are logistic regression and naïve Bayes.

They are distinctive by virtue of the types of data they enable us to work with to classify cases. Previously introduced logistic regression allows us to predict classifications on structured data. The outcome is returned as conviction of the outcome. Naïve Bayes classifies on the probabilities of occurrences of events in the data. Accordingly, it allows us to predict classifications upon the free-text variables in our dataset, regarding each word as an occurrence in the computation of probability of being one of two or more categories to a classification.

Both allow us to recover the classifications that are bad data. In fact, for various reasons, most of the classifications in operating systems are good data. Most notable is that they offer two opportunities for predictive insight.

We may want to understand the strength of relationship of operational variables to a categorical outcome. That was introduced as an insight for relationship. However, we may want to further understand, case by case, the probability and confidence limits to an outcome classification for the different permutations of predictor variables. Logistic regression allows us to get the insight from our data.

Apparency Insight. Are there hidden predictor variables to the performance of assets and processes? Figures 2-18 and 2-19 show two of the most popular models to seek out hidden variables—decision tree and K-Means. A distinction between them is to be directed and undirected. A decision tree model is given both predictor and outcome variables. In contrast, a K-Means model is given predictor variables but no outcome variable.

Figure 2-18 is a decision tree model. We seek predictive rules as variables leading to numeric and categorical outcomes.

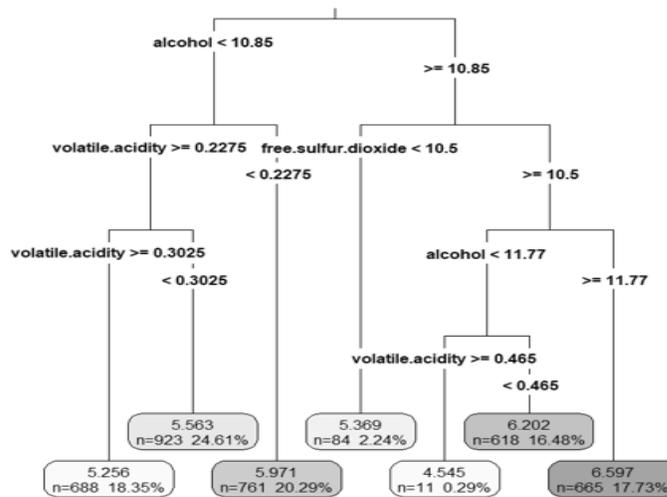


Figure 2-18: Decision tree with numeric outcomes.

At the tree branches, the model determines which one of all provided variables and their splits are most significant to the branching decision. Consequently, the variables are being sliced-diced into variables hidden within them.

A K-Means model also seeks predictor variables by slice-dice. However, there is not an outcome variable against which to decide the slice-dice. Instead, the model reveals clusters of similarity. When the clusters emerge, it falls to the experts in the asset

management operation to determine and name what they indicate. Some readers will recognize the similarity in concept to principal component analysis (PCA).

Figure 2-19 shows the case of six hidden variables teased out of two predictor variables. The decision to call for six clusters is determined by an “R” function that evaluates the data to determine the number of significant clusters to seek. However, there is no limit on the number of variables to model other than the interpretative challenge explodes.

The revealed tree and cluster variables can be joined back into the super table. The consequence is to tease insightful variables out of the source data that the firm’s operating systems cannot. The purpose is to increase the insight power of the models. The variables may also be created to overcome technical problems to a model such as collinearity, too many variables, etc.

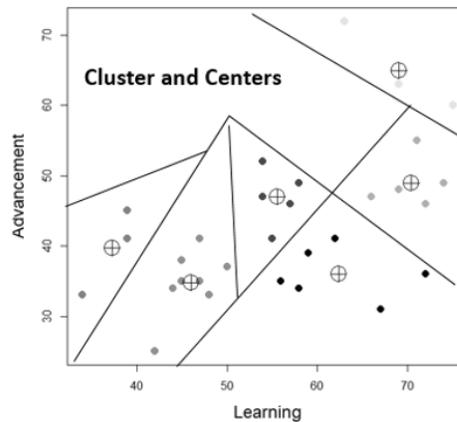


Figure 2-19: Six hidden variables teased out two input variables.

There are other models. They variously fall into categories of rule and cluster, and directed and undirected. If interested in them, the reader is directed to the book, Machine Learning with R, by Lantz text for explanation and demonstration.

Bibliography

- Alexander, Michael and Kusleika, Richard. Access 2016 Bible. John Wiley & Son, Inc. 2016. Parts 3 and 4.
- Bostrom, Goran. Event History Analytics with R, CRC Press. 2012
- Cowpertwait, Paul and Metcalfe, Andrew. Introductory Time Series with R, Springer. 2009.

42 | Chapter 2

- de Vries, Andries and Meys, Joris. R for Dummies. John Wiley & Son, Inc. 2015.
- Field, Andy and Miles, Jeremy. Discovering Statistics Using R. Sage Publications, Inc. 2012.
- Finch, Holmes. Multilevel Modeling Using R. CRC Press. 2014.
- Grolemund, Garret and Wickham, Hadley. R for Data Science, O'Reilly Media, Inc. 2017.
- Lantz, Brett. Machine Learning with R. Packt Publishing. 2015.
- Matloff, Norman. Art of Programing R. No Starch Press. 2011.
- Moubray, John. Reliability-Centered Maintenance. Second edition. Industrial Press. 1997.
- Oesko, Suljan. Pivot Tables In-Depth for MS Excel 2016. Suljan Oesko. 2017.
- Wickham, Hadley. ggplot2: Elegant Graphics for Data Analysis. Second edition. Springer. 2016.
- Zeileis, Achim, Kleiber, Christian and Jackman, Simon. Regression Models for Count Data in R. <https://cran.r-project.org/web/packages/pscl/vignettes/countreg.pdf>

Chapter 3

Super Tables from Operational Data

The definition of a data-driven asset management bears repeating. It is defined as using the firm's operational data to augment the experience and judgement of its operatives, managers, analysts and engineers as they plan, organize, conduct and control their processes.

The definition is repeated because this chapter will focus on two key words in the definition—operational data. Because data-driven asset management draws heavily on the firm's data, it must have the organizational ability to build super tables as the gateway to its operational data.

Let's establish a standard by which an organization would be judged as one with the organizational ability for super tables. For one, most role holders in the asset management operation would be conversant in super tables. Some role holders would have taken or been given the opportunity to acquire the skills to build super tables. Meanwhile, there has been a shrinking number of role holders who are neither conversant nor skilled in super tables but have the necessary basic skills to engage them in their assigned process tasks.

Furthermore, leadership would advocate for organizational development strategies to progressively increase the proportion of role holders who have the skills to build tables. As the proportion grows, the evolving effectiveness and efficiency of the asset management organization will have been observable. Just as importantly the personnel to the asset management enterprise would express an expectation for a brighter career.

The chapter serves two purposes toward becoming data driven. First, it is written for readers who are seeking to acquire the skills to develop super tables. As the training platform, the chapter will introduce, demonstrate and explain practices an SME reader would recognize as essential to topflight asset management. By emulating the practices with their own data or data provided with the chapter, they will gain the skills to develop any envisioned super table.

Second, the chapter is written for leaders, operatives and consultants who recognize their need to become conversant. For them, remaining relevant requires that when an insight deliverable is being imagined and developed, they must be a full participant in its discussion.

A careful read will make them conversant. Alternately, they can assign the task to prepare and present a concise, clear explanation to a person who has become conversant or fully skilled. They, in turn, can use the chapter to form the presentation.

The example of the chapter is constructed with representative variables from a leading CMMS. The data of the example is contained in the file SuperTable_Datasets.xls available for download at the webpage provided by the preface.

Although representative, all identifying variables of the data have been either masked or redacted. The hour and dollar variables have been scrambled such that readers should not attempt to compare what is demonstrated to their own plants. In other words, the data is representative but no longer has a plant.

The chapter will first establish some initial perspective. It will then introduce a sequence of three cases as data-driven practices. For each case, the chapter will demonstrate and explain how the data are built into an enabling super table. Especially notable of the cases is that few asset management organizations have the skills to conduct the demonstrated practices even though they are fundamental and essential to asset management. For them, the chapter will be an immediate step into the new age.

The reader will achieve full competency from the chapter. The competency will be extended and hardened by Chapters 9 and 11 in data-driven dual dimensional budgeting and variance control, job planning and scheduling. However, after reading the chapter, readers are encouraged to read Chapters 8 through 13 of the text, “Access 2016 Bible. The comprehensive tutorial resource.” Furthermore, every feature and function introduced in the chapter and the text can be viewed in a several-minute YouTube video.

Because the variables are common to any CMMS, readers are encouraged to emulate with their own data what is demonstrated—use the chapter as a go-by. It is invariable that people who begin with a go-by, immediately find themselves ranging out to build newly imagined super tables.

3.1. Perspective

Chapter 2 introduced what we are trying to do. Figure 2-5 showed that we query for standard reports from our systems. They collectively contain the variables we need to build our intended insight deliverables. However, we are divided and conquered by our systems unless we can join the variables in a single super table. This section will establish some basic perspectives of super tables. We begin with three truths.

First, almost all modern operating systems allow their data to be extracted in table format as standard reports. The reports are formatted as columns (variables) and rows

(records or cases). Columns are titled as headers. Rows are never titled. Nor are there empty or summary rows.

Second, tables from any one or more systems can be joined in a single super table. The only requirement is that they must have a common identifying variable with which to join them.

Third, bad data is rarely a deal-killer. We are rarely dependent upon one variable to get an insight. Furthermore, cleansing will often neutralize the flaws.

A final perspective. The process of building super tables entails two stages—foundational and aggregational.

The first stage builds the foundational table to one or more insights. The stage joins source sub tables in a massive raw table of all source variables and then pulls the variables we want into the intended super table. Additional variables to the super table are typically formed or calculated upon the source variables.

The second stage designs aggregation variables into super tables. The concept is to specify groups in the foundational table. In turn, aggregation variables are created for the groups. They can be count, sum, average, standard deviation, variance, min-max and first-last.

3.2. Scenarios for Demonstration

The scenarios of the Chapter, as explanation by demonstration, will develop a super table and two aggregations for the purpose of gaining insight to craft hours to a report month. The hours will be narrowed to those that were engaged by the maintenance tasks of the work orders that were completed during month.

The scenarios are a consequential topic with which to explain super tables. Craft hours are the largest and most controllable expense of asset management. As data, they are also the life blood to all sorts of insights that are not available from a CMMS.

We will use three standard reports that are natural and typical to any CMMS. First are work orders. Second are the maintenance tasks to each order. Third are the hours per individual craft to the maintenance tasks that engaged them. The data, as subtables, are available from the file SuperTbl_Chap3.xlsx which is available for download at the webpage provided by the preface.

Some standard reports to a CMMS offer a considerable list of variables for selection. This is typical to the standard report of work orders. The three tables were extracted from the CMMS with not only the variables we know we want to employ in the planned insight deliverable but also with those we may find ourselves wishing we had included as the deliverable takes form.

The chapter will work through a sequence of three cases. Each is an extension upon Case 1 to build a foundation super table with the three standard sub tables from the CMMS.

Cases 1 and 2 extend from the super table to demonstrate an important, fundamental practice for asset management. Some operations evaluate a month's work orders based upon exploring the top ten in cost. However, the problem is that the top ten may legitimately be the top ten rather than indicative of something of interest.

Case 2 of the scenario will search for true outliers. It will use a statistical Z-score standardized method. As a result, we can ask ourselves which orders are beyond some percentage of occurrence such as 95 percent. The analysis will seek outliers for investigation among groups as permutations of cost center and maintenance type.

Case 3 will take an important further step for seeking out true outliers. The purpose is to conduct a more realistic search. The case works around a debilitating weakness that is inherent to how the representative CMMS is configured.

The configuration classifies orders by type (e.g., preventive, on-condition, corrective) but not by craft lead (e.g., machinist, mechanical, electrical, instrumentation). The distinction is difficult because more than one craft is typically engaged in an order.

Therefore, Case 3 will classify records in a way the representative CMMS does not. It will seek outliers among groups as permutations of cost center, maintenance type and lead craft.

Although not a case, there will be a final demonstration. The demonstration will compare the findings of the two cases to the top-ten method and to each other.

The chapter will follow the steps introduced in Chapter 2 and flowcharted in Figure 2-6. Recall they are as follows:

1. Identify relevant tables.
2. Extract tables from source systems.
3. Import or connect source tables into Access.
4. Build foundation super table.
5. Explore and cleanse the data.
6. Build aggregate variables into the super table.
7. Administer, update, upgrade, and disseminate super table.

This chapter will speak to the first four steps to identify, extract, import or connect and build the foundation super table, and the sixth step to build aggregate variables to the super table. The explore and cleanse step will be a topic of Chapters 5 and 6. explaining how to get our data and maintenance process under control. Finally, with respect to the seventh step, the chapter will speak to some organizing issues within the Access file.

3.3. Case 1: Foundation Super Table

We begin by building the foundation super table. Along the way, the section will explain the process, and the principles and practices of building tables.

3.3.1. Identify, Extract and Import or Link

As the first step, we have found that all pertinent data to our envisioned super table reside in three tables to the CMMS. They are work orders, tasks to the orders and craft hours to the tasks.

Operating systems typically give us a choice to download to Excel, Access and other destinations. The demonstration will take the trail through Excel.

The next step is to download the identified source system tables to an Excel file. I recommend placing them in a single file. As a staging area, we may want to conduct basic preparations of the data. We may want to change headers. We may want to conduct some know-thy-data activities. Of course, we can conduct the same work in Access, but some tasks may be easier in Excel.

The third step of the process is to pull the extracted tables into Access. Once again, a simple step. We merely follow pop-up windows. Start the process by clicking the New Data Source icon on the Access Ribbon under the External Data heading of the menu.

We will be given four choices—from file, data base, other services and online services. Given that we have chosen to download our tables to Excel from the CMMS, we would now select the from file option and select Excel. If we had downloaded the system tables to another Access file, we would have selected the from Database option and selected Access.

Figure 3-1 shows another important distinction for managing data-driven processes and insights. Notice the list of tables. There are two additional tables to the three we previously identified and extracted from the CMMS. They will be explained later in the chapter.

At this point, an important point to notice is the symbol to the left of the bottom-two tables. This means that the tables have not been imported. Instead, they are linked to the source—in this case the Excel file at which they are managed. Each time the query is run, it will reach outside for the latest version of the data. The option to import or link is made available as we follow the process beginning with selecting a source.

The difference is noteworthy because the seventh step is to administer sub tables and super tables. The users may link rather than import some or all tables so to be assured that they have the latest edition of a done-by-one, used-by-all source. If we import data and it

48 | Chapter 3

is later updated or kept clean elsewhere, our insights may be misinformation unless they are linked the done-by-one edition.

The purpose of the two linked tables and how they are developed is explained in the next section. As external to the CMMS, they are what the book calls translation tables.

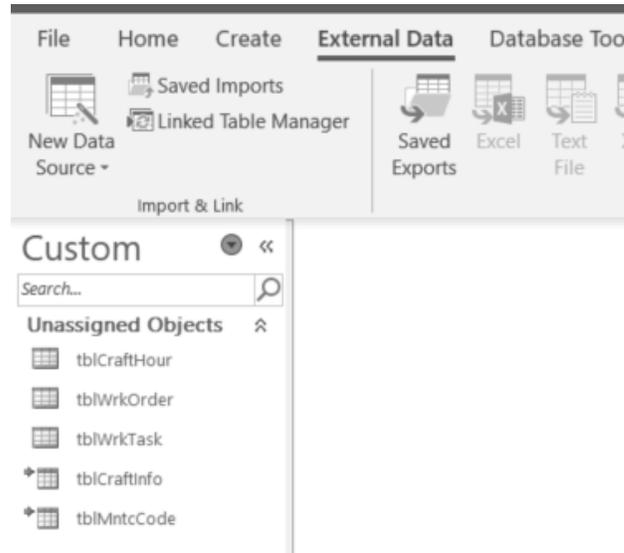


Figure 3-1: Three downloaded tables and two clarifying tables.

Now that the tables are loaded, we select the Create option in the menu and then the Query Design icon in the Queries section of the ribbon. As shown in Figure 3-2, the Select query will open automatically with the Show Table pop-up window.

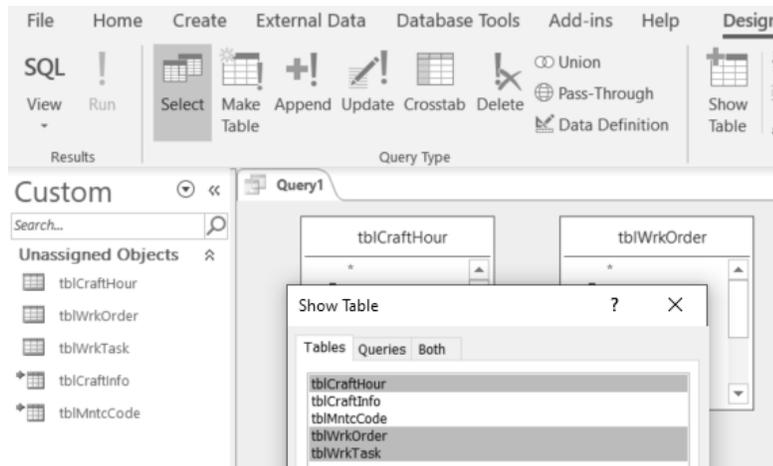


Figure 3-2: Tables imported and linked to Access for building super tables.

In the figure we have extracted the variable of order type code from the CMMS. Thence, we have created the two variables MntcType and MntcDescription as alternatives to the OrderType code. From Access, we can either import or link to the translation table.

It follows that we can return at any time to the translation table and add clarifying variables. An example would be a variable based upon the distinctions of the task types for reliability-centered maintenance.

The other translation table in the object pane allows the super table to classify craft types by useful categories. With the translation table, tblCraftInfo, the hours by individual crafts are translated to different designations rather forcing us to work with the many fragmented but equivalent titles in the CMMS. By the table, 42 craft titles have been translated to 8 categories.

Over time, expect that an entity will develop and maintain a collection of translation tables. They will be called into existing and developing super tables.

Accordingly, think in terms of storing the translation tables as a collection in a single Access file, place them for access by anyone and manage them as a role in the data-driven asset management operation.

When we run a query linked to the translation tables, the latest version is pulled into the run. Consequently, users whose role entails tasks to update insight deliverables can know that the source data is up to date by virtue of being linked to the administered source.

3.3.3. Overview of the Query Process

The ribbon under the tab, Design, reveals that there are six types of queries. However, most of everything is done in a select query. The others are supplemental tools to build, update and administer the select query. How they play will be explained opportunistically as this and later chapters unfold.

The definitions of the six types of queries (shown in Figure 3-4) are as follows:

- **Select:** Builds the super table from one or more subtables. Aggregation is constructed in the select query.
- **Make Table:** Converts a select query to a “hard” table.
- **Append:** Adds rows of data to an existing table.
- **Update:** Changes rows to variables in a table or query.
- **Delete:** Removes rows to variables in a table or query.
- **Crosstab:** Makes long tables wide—e.g., a variable of months transformed to a table with variables for each month.

Our first intent is to open a new select query. Click Create in the menu bar and then Query Design in the Queries section of the ribbon. Access will open a new select query.

Right click the query’s name tab, select save and give the query its name (qryCase1_Foundation).

Next, we drag the tables we want into the query and join them. Thence, we drag the variables we want into the design grid. In the grid, we mold the variables to our super table.

Figure 3-4 shows the layout of the design view. We can see how the pieces come together. To get a deeper look, the sections of the layout will be enlarged and explained in the paragraphs to come.

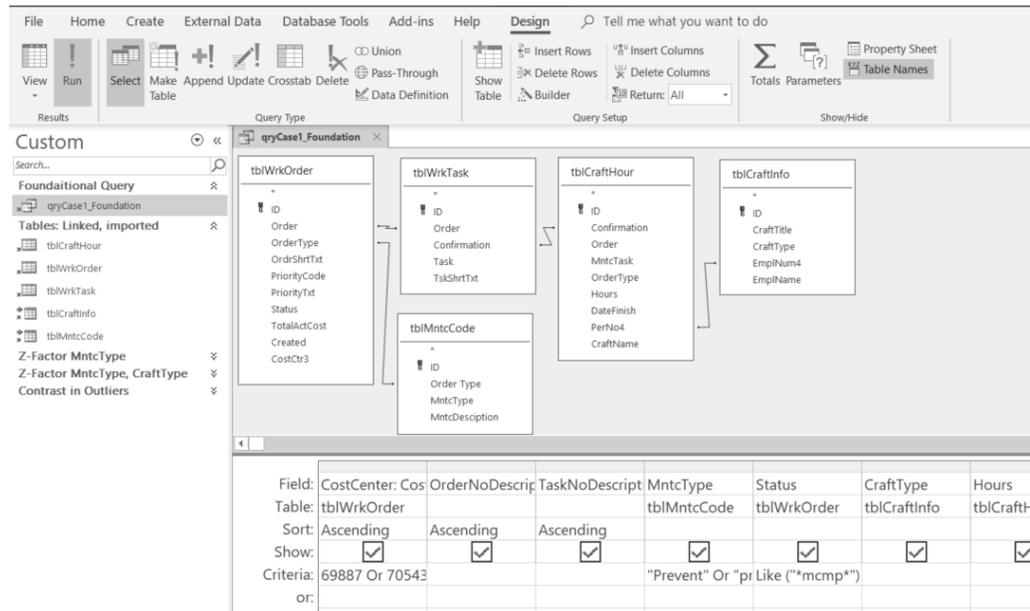


Figure 3-4: Design view to the super table.

Figure 3-5 shows what geeks call objects—tables, queries, etc. On inspection of the objects view, we can see the original five source tables. However, other objects emerged as the queries of the chapter were developed. Section 3.8 will explain the objects view and how objects are organized as shown in the figure.

This is an opportunity to recommend a good practice. Notice the “tbl” prefix to table objects and “qry” prefix to the query object. In a list of objects, the prefixes allow us to readily distinguish one type of object from another. The value of the practice looms large when there are many objects in an Access file.

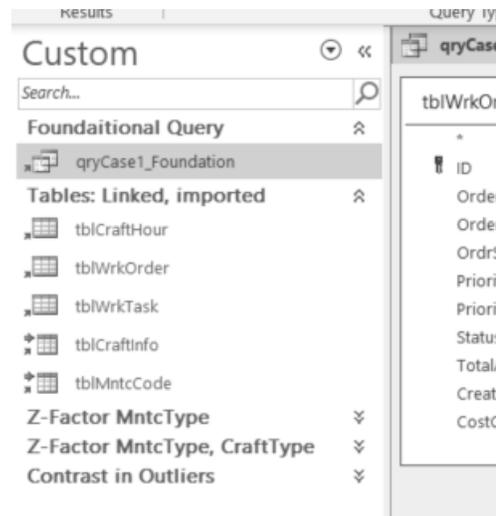


Figure 3-5: List of objects associated with the super table.

Now let's look at how the sub tables are joined in the super table. Figure 3-6 shows that all five tables have been pulled into the query. However, let's note again that queries can also be pulled into a query. We are not limited to tables. This will be seen to happen for cases 2 and 3, and the comparison analysis.

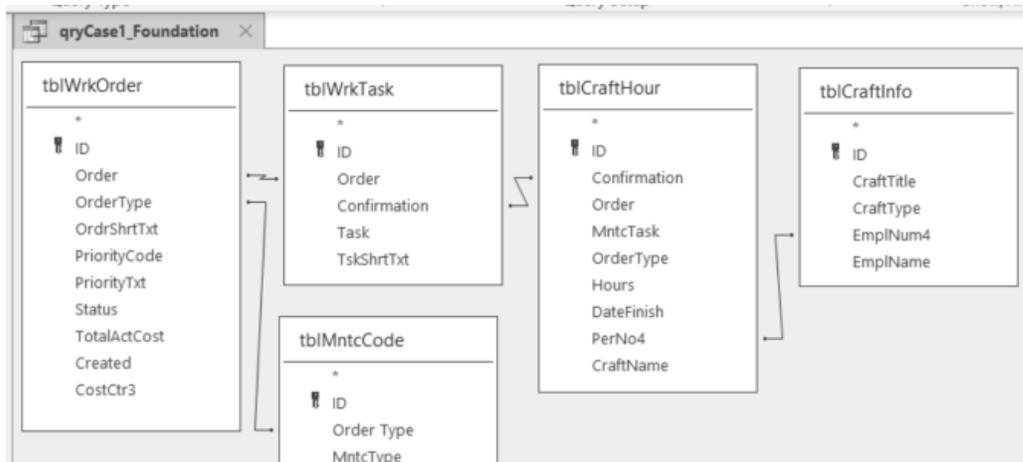


Figure 3-6: All five tables to the super table and their joins.

Recall that tables are joined by an identifier variable they have in common. Upon inspection we can see the `tblWrkOrder` and `tblWrkTask` tables are joined on order number, and the `tblWrkTask` and `tblCraftHour` tables are joined on confirmation number.

The previously explained two translation tables are also engaged. The tblWrkOrder and tblMntcCode tables are joined on the order type variable. The tblCraftHour and tblCraftInfo tables are joined on their respective employee identifier variables.

The last join demonstrates an important point. The joining variables need not be of the same name. They must only be of the same type—numeric or character.

Joins between tables have types. They are classified as inner, left, right and outer. The types and ramifications of each will be explained in the next section.

Figure 3-4 showed the design grid at the bottom of the design view. It is the area that the joined tables are molded into our envisioned super table. What we see in the grid is the code that will return the super table.

Let's establish a reference point. Standard Query Language (SQL) is the universal code to pull data from one or multiple relational databases and build tables. However, the intent of Access is that the code we place in the design grid has become normal to our daily tasks.

When we are placing what we have become friendly with in the grid, the SQL code to actually do the deed is forming behind the curtain. This is called query by example (QBE). As we work in the grid, we are telling Access what we want to be coded as SQL. Therefore, we do not need to learn SQL because the QBE functionality does it for us.

3.3.4. Joins and Their Ramifications

Figure 3-6 showed the tables as joined by the line between a common variable to pairs of objects. However, there is more to it. As mentioned earlier, the join line represents one of four types of joins. We will use Figure 3-7 to explain them. Then we will see how to make the choices for our super table.

To the left in the figure are two tables—TableA and TableB—with order number as the common identifier variable. However, the cell contents are not one for one. Consequently, the join we chose will return different super tables as shown at the right of the figure.

The inner join will return a table in which the records are the ones in which both cells are populated with identical order numbers. Only the two such records are returned. All others do not appear in the super table.

The left join returns all populated cells from the left table. The non-matching cells in the right table appear as empty cells. In contrast, the right join will return the opposite outcome to the left join.

create an outer join table, pull it into the query and join the tables through it. Chapter 11 will demonstrate the power of outer joins to analyze work attainment.

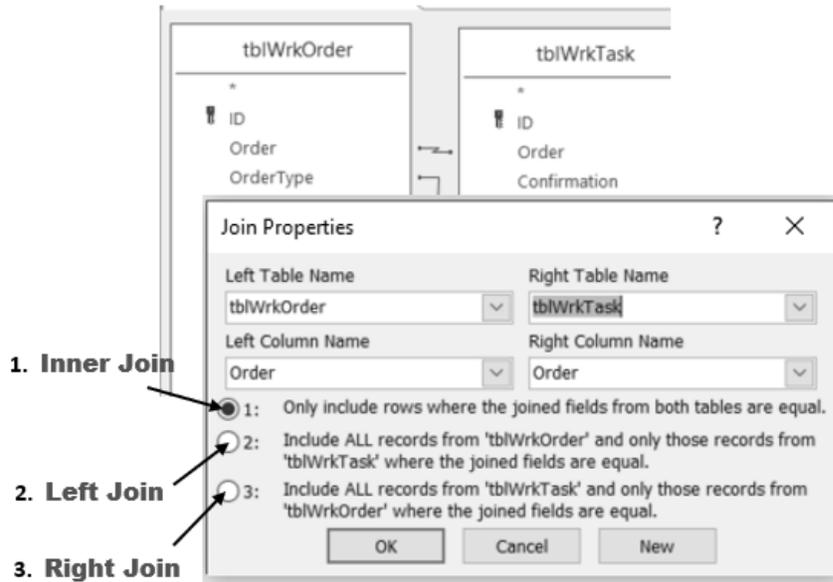


Figure 3-8: Select the join type, if not the default inner join.

The joins to the super table, qryCase1_Foundation, are provided in Code 3-1. They are presented as they would appear in the Join Properties pop-up window (Figure 3-8).

Code 3-1: qryCase1_Foundation joins

Side:	Left	Right
Table:	tblWrkOrder	tblWrkTask
Variable:	Order	Order
Join:	Inner	
Side:	Left	Right
Table:	tblWrkTask	tblCratHour
Variable:	Confirmation	Confirmation
Join:	Inner	
Side:	Left	Right
Table:	tblCratHour	tblCraftInfo
Variable:	PerNo4	EmpNum4
Join:	Inner	
Side:	Left	Right
Table:	TblWrkOrder	tblMntcCode

Variable:	Order Type	Order Type
Join:	Inner	

This is a good place to introduce the Append query. An append query is a simple matter. Rather than present a full discussion, the reader is advised to find a several-minute YouTube demonstration by browsing the internet for “append query access.”

In this section, the Append query is introduced as a tool to build an outer join. More commonly, it will arise when we are updating tables in Access rather than in Excel. It is usually more practical to append updates in Access. Another reason is that a table, periodically appended in Excel, may eventually grow to hold more records than Excel has the capacity to hold.

3.3.5. Coding the Design Grid

It is a choice of personal style. When extracting each table from its source system, consider selecting all potentially useful variables rather than only the ones for the currently developing super table. As we build our super table or use the data for other super tables, we are often glad we did.

Now we select the variables of interest to the super table named qryCase1_Foundation. This is done by dragging each variable from the design upper view into the Field row of the design grid. In most cases the Table row will populate automatically. Thence, four other aspects remain to be coded as it is necessary—Sort, Show, Criteria and Or.

The code as placed in the design grid of the super table is provided in Code 3-2. The code will be explained by the paragraphs to come. The alternatives to the shown code will also be provided along the way. Therefore, the reader will be introduced to almost every code they will ever need for any envisioned insight.

Code 3-2: qryCase1 Foundation—grid code.

Join	See Code 3-1		
Field	CostCenter: CostCtr3	OrderNoDescription: [tblWrkOrder].[Order]&": "&[OrdrShrtTxt]	TaskNoDescription: [Task]&": "&[TskShrtTxt]
Table	tblWrkOrder		
Sort	Ascending	Ascending	Ascending
Show	Y	Y	Y
Criteria	69887 Or 70543 Or 70107 Or 69839		
Or			

Super Tables from Operational Data | 57

Field	MntcType	Status	CraftType
Table	tblMntcCode	tblWrkOrder	tblCraftInfo
Sort			
Show	Y	Y	Y
Criteria	“Prevent” or “Proactive” or “Reactive”	Like(“*MCMP*”)	
Or			
Field	Hours	DaysPastCreated: [DateFinish]-[Created]	Created
Table	tblCraftHour		tblWrkOrder
Sort			
Show	Y	Y	Y
Criteria			Between #1/1/2011# And #2/27/2012#
Or			
Field	Order	Confirmation	
Table	tblWrkOrder	tblWrkTask	
Sort			
Show	Y	Y	
Criteria			
Or			

The first variable identifies the cost center. The variable CostCtr3 was dragged down and its source table, tblWrkOrder, automatically populated the Table row of the grid.

However, we want to give the variable clarity in the super table. We prefer CostCenter as the named variable. Accordingly, we rename it—give it an alias—by placing CostCenter: (name, colon and space) in the Field row of the grid.

Not all variables in the grid may need to appear in the returned super table. Accordingly, the Show row allows us to cause a variable to not appear although it is in the grid. The default is to appear. Otherwise, we would deactivate the check box (depicted as Y/N in Code 3-2) to the variable.

The Sort row commands the table to be returned with the variables as ascend, descend or no order if empty. In the code, the table will be sorted in ascending order from left to right—cost center, orders to cost center and tasks to order.

In the Criteria row, we mold the table by filtering the records to each variable. With respect to CostCenter we have reduced our table to 4 of the plant’s 95 cost centers.

Notice the Or operator in the code of the Criteria row. The operator indicates that we want all the records to the four centers—all else excluded. An And operator in the variable

3.4. Case 2: Seek Outlier Work Orders

Plants are often limited to seeking outlier orders with crude methods. One is to investigate the top ten orders in cost or proxy to cost. The problem is that maybe some or ten are supposed to be the top ten. Exploring them reveals no opportunities.

With respect to the data of the super table, a better method is to seek the outliers with respect to cost center and maintenance type. Better yet is to include order lead craft type to the grouping.

Unfortunately, the example CMMS is not configured with a variable for lead craft. Craft lead is only sporadically noted in the description to each order. By the way they are noted in the description text, we could tease out the classifications with string functions. However, the classifications would only be partial because compliance has been less than 100 percent.

This section will demonstrate and explain aggregation variables. It will conduct a search for outliers to demonstrate aggregation in action. Two cases will be presented in this and the next section.

Case 2 of this section will demonstrate aggregation with the natural variables to the representative CMMS. Case 3 of the next section will apply aggregation to tease a craft lead variable out of the CMMS—find hidden variables. Although not demonstrated in the chapter, we would want to join the classifications as a variable to the super table—making it more super.

However, let's note the alternative to aggregation in Access. The aggregations of Access are also available in Excel Pivot. Furthermore, Pivot is the first choice because it allows greater interactive and visual exploration than is natural to Access.

The difference is that aggregations in Pivot cannot deal with complex developments such as Cases 2 and 3. Of course, aggregations developed in the Access super table can be explored and further aggregated in Pivot.

3.4.1. Measurement for Outliers

We will apply a statistical test for outlier orders with respect to variance from average—Z-Score standardized. The idea is that we want to spot the orders with spending beyond some percent of all orders in a group of orders.

The Z-score test calculation is:

$$\text{Z-Score Standardized} = (R - \text{Avg}) / \text{Std Dev} \quad (3-1)$$

Where:

R = Individual record.

Avg = Average or mean of the records to each group.

StdDev = Standard deviation of the records to each group.

For the demonstration of aggregation, we will assume the data is normally distributed. In life, we would test the assumption and act accordingly. Chapter 4 introduces and demonstrates graphic methods to test the assumption with histograms (Figure 4-7) and q-q plots (Figure 4-8). There is also the Shapiro-Wilk normality test function—`shapiro.test()`—in “R.” The set up and interpretation of the function can be found on the internet.

Next is the issue of deciding the percentage on which to base the Z-Score. It is a choice for the data-driven asset management organization to make. Figure 3-14 provides scores associated with a range of choices.

The Z-scores will be associated with a group of orders. Case 2 is grouped upon cost center and maintenance type.

Percent	Z-Score (+/-)	
	One-sided	Two-sided
90.0	1.29	1.65
95.0	1.65	1.96
99.9	3.10	3.27

Figure 3-14: Z-Score associated with percent outlier.

The measured variable of interest to both outlier demonstrations will be craft hours. Hours, rather than dollars, are a better window into engaged maintenance capacity. Furthermore, hours submit well to related calculations such full time equivalent (FTE)

For the demonstrated cases, it will be assumed that the asset management operation wishes to investigate orders for which hours equal or exceed 95 percent of all orders to their respective groups. Accordingly, we will demonstrate with one-sided 95 percent. The associated Z-score is equal to or greater than 1.65.

If we were interested in orders beyond the two tails of the distribution, we would look for orders with a Z-score equal to or greater than 1.96 absolute. It is called the two-sided test.

We may exercise the two-sided test to seek oversized and undersized orders. Undersized orders may flag a self-inflicted fatal problem to ever achieving maximally effective and efficient asset management operations.

We may be observing the effects of craft hours not being accurately recorded to the orders that incurred them. This may, in turn, suggest that not all the orders at the upper

extreme truly overran the work plan. Some have recorded to them hours engaged by other orders.

3.4.2. Activating Aggregation

Aggregation variables are built within a select query as shown in Figure 3-15. With a table or query in the design view, we check the Totals (summation symbol) icon in the Access ribbon. Figure 3-15 shows the action to develop the query qryHoursOrder and the resulting table. The code is provided by Code 3-3, qryHoursOrder.

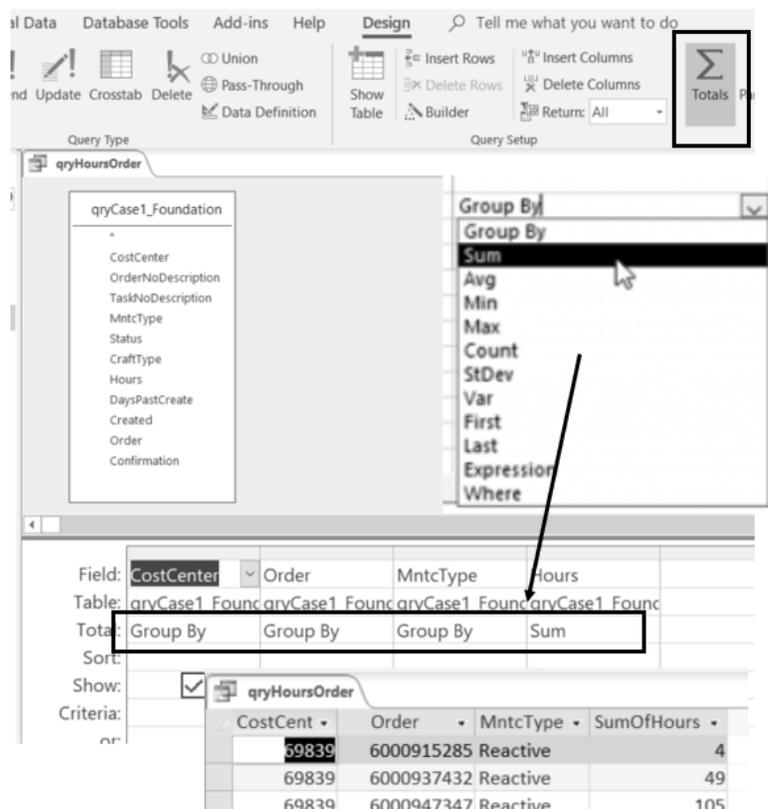


Figure 3-15: Clicking the summation icon adds a new row in the design grid.

Upon doing so, notice the change in the design grid. The Total row appears.

The lower portion of the figure shows the result. Without aggregation, each order would appear as a record for each individual craft engaged for each day of the order—time sheet records. Instead, there is a single record for each order. The hours incurred for all individuals to each order have been summed.

In the Total row, for each variable, we select from ten options for summation—group-by, sum, average, min, max, count, standard deviation, variance, first and last.

Notice in the figure and the code, we have pulled the foundation super table into the query. In the figure we have grouped on cost center, order and maintenance type.

The Hours variable is aggregated with the option to Sum. However, we are not limited to one aggregation per variable. We can repeat the Hours variable as any one of the non-grouping options for aggregations. This will be seen as the demonstration unfolds.

Nor are we limited to a single non-group variable in the super table. For example, we could add another variable and select options for it (not demonstrated).

The possibilities of aggregation will appear in action throughout the remaining chapter. Most noteworthy are the immense ramifications. This is because so many insights involve calculations with group variables. The two cases of the chapter are only several of almost infinite possibilities.

Code 3-3: qryHoursOrder—join and grid code.

Join	None: qryCase1 Foundation is sole source.			
Field	CostCenter	Order	MntcType	Hours
Table	qryCase1_Foudatio n	qryCase1_Foudatio n	qryCase1_Foudatio n	qryCase1_Foudatio n
Total	Group By	Group By	Group By	Sum
Show	Y	Y	Y	Y

3.4.3. Planning the Aggregation

The demonstrated foundation super table was a straightforward, follow-our-nose piece of work. However, sometimes it is good to flowchart what we are trying to do. Figure 3-16 is a flowchart of queries to be created and joined to arrive at the envisioned insight deliverable—qryZScoreOrder.

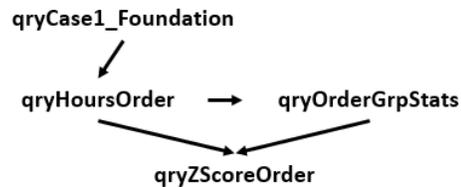


Figure 3-16: Case 2 flowcharted to the final insight deliverable.

3.5. Case 3: Seek Outliers with Lead Craft

Case 3 will extend the demonstration and explanation of Access skills by teasing a classification from the CMMS that it has not been configured to capture. We are seeking a variable that is hidden among others.

The ability to classify orders by lead craft is obviously fundamental to managing a maintenance operation. Without the classification, a maintenance SME would expect that averages calculated solely upon what is naturally provided by the representative CMMS are potentially misleading in the search for outliers.

A solution is an opportunity to demonstrate and explain the First-Last aggregations. We will look for outliers based on the groups of cost center, maintenance type and lead craft. Orders will be classified upon the assumption that the craft with the greatest hours to an order classify the lead craft.

3.5.1. Planning the Lead Craft Aggregation

As advised for Case 2, we should flowchart the sequence of queries to reach the end Z-score. Figure 3-18 flow charts the case. The next section will describe how each object is developed.

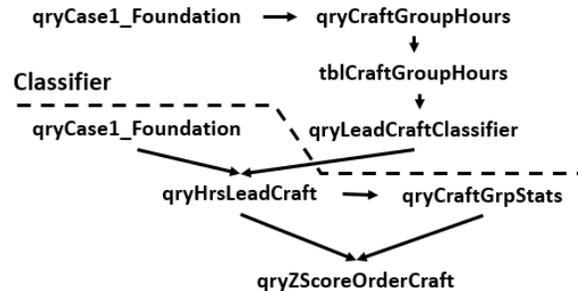


Figure 3-18: Flowchart to Z-score for orders grouped on cost center, maintenance type and lead craft.

Below the broken line, the sequence of queries that parallel Case 2. The difference is that the groups include lead craft. Above the line are additional queries that perform the lead craft classifications—the classifier.

3.5.2. Arriving at Z-Score with Craft Lead

The object `qryCraftGroupHours` is built with `qryCase1_Foundation` as its only input. Code 3-6 provides the join and grid code to build it.

Code 3-6: `qryCraftGroupHours`—join and grid code.

Join	None: <code>qryCase1_Foundation</code> is the sole source.
-------------	--

Field	CostCenter	Order	MntcType
Table	qryCase1_Foundation	qryCase1_Foundation	qryCase1_Foundation
Total	Group By	Group By	Group By
Sort	Ascending	Ascending	
Show	Y	Y	Y
Field	CraftType	Hours	
Table	qryCase1_Foundation	qryCase1_Foundation	
Total	Group By	Sum	
Sort		Descending	
Show	Y	Y	

The upper portion of Output 3-4 shows what is returned by the code. Some orders have engaged multiple crafts. One such order is boxed in the figure. The hours for all engaged crafts have been summed by craft type. They are ordered from largest to smallest by virtue of the descend command for the Hours variable.

CostCenter	Order	MntcType	CraftType	SumOfHours
69839	6000915285	Reactive	Electrician	4
69839	6000937432	Reactive	Machinist	49
69839	6000947347	Reactive	Machinist	77
69839	6000947347	Reactive	Electrician	16
69839	6000947347	Reactive	Instr/Elec	9
69839	6000947347	Reactive	MultCraft	3
69839	6000957907	Prevent	MultCraft	4.5
69839	6000959888	Prevent	MultCraft	13.5
69839	6000961312	Prevent	MultCraft	7.5

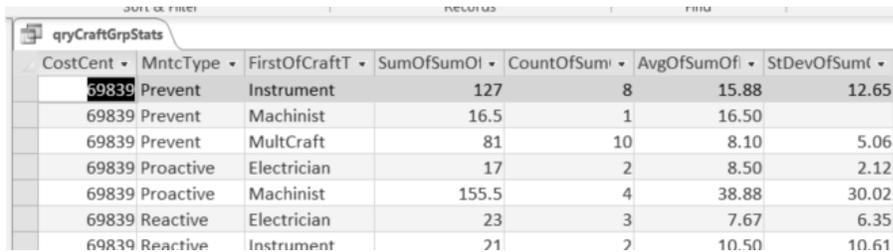
CostCenter	Order	MntcType	FirstOfCraftT
69839	6000915285	Reactive	Electrician
69839	6000937432	Reactive	Machinist
69839	6000947347	Reactive	Machinist
69839	6000957907	Prevent	MultCraft
69839	6000959888	Prevent	MultCraft
69839	6000961312	Prevent	MultCraft
69839	6000961792	Prevent	MultCraft

Output 3-4: Hours sorted by craft hours and then selecting the first line to each order.

The next step is to build the craft classifier query—`qryLeadCraftClassifier`. It is formed by running a query upon the returned data of `qryCraftGroupHours`.

However, there is a problem if the `qryLeadCraftClassifier` object is built with `qryCraftGroupHours` as its source data. When we confirm the result in the returned table,

section of the ribbon. Then, at the bottom of the table window, confirm that the sum of hours and order counts match what we started with from the foundation super table.



CostCent	MntcType	FirstOfCraftT	SumOfSumOf	CountOfSum	AvgOfSumOf	StDevOfSum
69839	Prevent	Instrument	127	8	15.88	12.65
69839	Prevent	Machinist	16.5	1	16.50	
69839	Prevent	MultCraft	81	10	8.10	5.06
69839	Proactive	Electrician	17	2	8.50	2.12
69839	Proactive	Machinist	155.5	4	38.88	30.02
69839	Reactive	Electrician	23	3	7.67	6.35
69839	Reactive	Instrument	21	2	10.50	10.61

Output 3-6: Statistics for the 32 groups of cost center, maintenance type and craft.

As shown in the flowchart, now we will combine the hours of each of the 128 orders and the group statistics in a query that is developed to compute the Z-score. Furthermore, rather than look at all orders, the query will filter to orders equal or greater than the threshold Z-score. Code 3-10 provides the join and grid code to qryZScoreOrderCraft.

Code 3-10: qryZScoreOrderCraft—join and grid code.

Join	Inner: qryHrsLeadCraft and qryCraftGrpStats on CostCenter Inner: qryHrsLeadCraft and qryCraftGrpStats on MntcType Inner: qryHrsLeadCraft and qryCraftGrpStats on FirstOfCraftType.		
Field	CostCenter: CostCtr3	Order	MntcType
Table	qryHrsLeadCraft	qryHrsLeadCraft	qryHrsLeadCraft
Show			
Criteria			
Field	Craft: FirstOfCraftType	OrderHrs: SumOfHours	GroupHrs: SumOfSumOfHours
Table	qryHrsLeadCraft	qryCraftGrpStats	qryCraftGrpStats
Show	Y	Y	Y
Criteria			
Field	GroupCount: CountOfSumOfHours	GroupAvg: AvgOfSumOfHours	GroupStDev: StDevOfSumOfHours
Table	qryCraftGrpStats	qryCraftGrpStats	qryCraftGrpStats
Show	Y	Y	Y
Criteria			
Field	ZScore: ([SumOfHours]-[AvgOfSumOfHours])/[StDevOfSumOfHours]		
Table			
Show	Y		

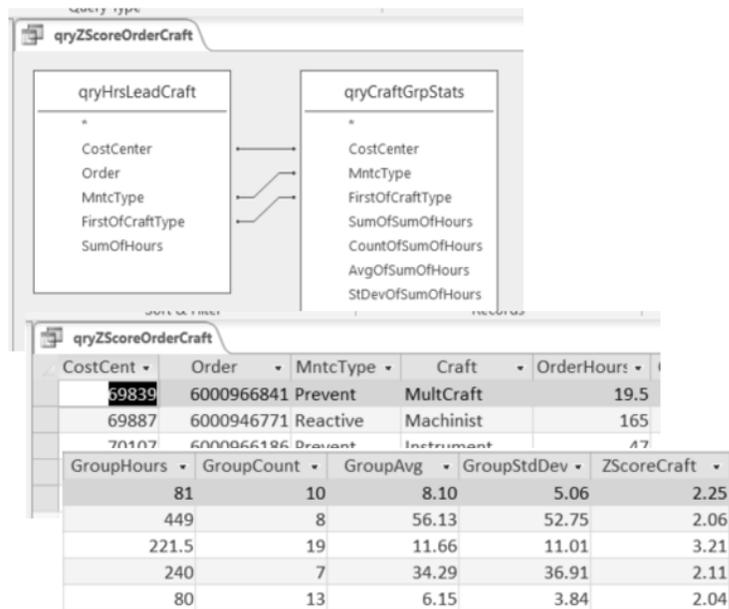
Criteria	>=1.65
----------	--------

What is returned by the joins and code are shown in Output 3-7. The first thing to note is the triple join. As mentioned earlier, an alternative strategy is to concatenate the three join variables in both tables and then join on them.

The Z-score calculation is placed in the field row. The threshold Z-score is placed as a criterion to the ZScore Field.

As presented for Case 2, we may choose to build a T-score variable. The chapter leaves it to the readers to extend what was described for Case 2.

Once again aliases are used heavily. They can be seen in the field cells to all variables. This makes the final product more readily consumed by those to which it is disseminated.



Output 3-7: Z-score for lead craft and the triple join to create it.

3.6. Comparison of Findings

As the final demonstration, let's compare the findings of the cases. We will compare approaches with respect to identifying outliers with ever sharper methods—top ten mostly costly, CMMS standard variables (Case 2) and with the added craft lead variable (Case 3).

The number of outliers is reduced from 11 to 5 when we seek outliers as Case 2 as compared to Case 3. Six of the outliers revealed by Case 2 also appear in the top ten. Two of the outliers revealed by Case 3 make an appearance. Cases 2 and 3 have three orders in common.

As mentioned earlier, having followed the chapter, its readers have become fluent in building super tables with Access. Therefore, this is the place at which the book will cut its readers loose. Rather than show the queries and codes to make the comparisons, the paragraphs to follow will describe how they can be done and leave the reader to execute.

The queries for Cases 2 and 3 have revealed a respective number of outliers. To determine how many appear in the top ten it is necessary to create a query to return the ten most costly orders.

With the Case1_Foundation object as the source data to the query, create a sum aggregation on hours with orders as the group. While in the design view, find the Return box located in the Query Set Up section of the ribbon and enter ten.

Next, for each Z-score case, create a query to determine the number of orders that also appear in the top ten. The input data are the newly created top-ten query and the respective Z-score queries. Join on order.

Finally, we determine how many orders appear in both outlier groups. Pull the Z-score queries as source data into a newly created query. Make an inner join on order.

Of course, if we had extended the analysis to the T-score, there would have likely been different results. The reader is invited to extend Cases 2 and 3 to include the T-score and then make the same comparisons.

3.7. SQL in the Background

This is a good place to graphically and dramatically demonstrate why data-driven asset management has become a modern reality. Not too awfully long ago what the chapter has demonstrated required nothing short of a data scientist. Only they had the skills to write the necessary SQL code. Remember the days when we waited for IT to get our data?

We can see in the SQL code to develop the qryCase1_Foundation object that specialized skills were a huge obstacle. Imagine being required to write the SQL code shown below; correct syntax and all. Or imagine the training that is required to be able to write the code.

```
SELECT tblWrkOrder.CostCtr3 AS CostCenter, [tblWrkOrder].[Order] & ": " &
  [OrdrShrtTxt] AS OrderNoDescription, [Task] & ": " & [TskShrtTxt] AS
  TaskNoDescription, tblMntcCode.MntcType, tblWrkOrder.Status,
  tblCraftInfo.CraftType, tblCraftHour.Hours, [DateFinish]-[Created] AS
  DaysPastCreate, tblWrkOrder.Created, tblWrkOrder.Order,
  tblWrkTask Confirmation
FROM (((tblWrkOrder INNER JOIN tblWrkTask ON tblWrkOrder.Order =
  tblWrkTask.Order) INNER JOIN tblMntcCode ON tblWrkOrder.OrderType =
  tblMntcCode.[Order Type]) INNER JOIN tblCraftHour ON
```

```
tblWrkTask.Confirmation = tblCraftHour.Confirmation) INNER JOIN
tblCraftInfo ON tblCraftHour.PerNo4 = tblCraftInfo.EmplNum4
WHERE (((tblWrkOrder.CostCtr3)=69887 Or (tblWrkOrder.CostCtr3)=70543 Or
(tblWrkOrder.CostCtr3)=70107 Or (tblWrkOrder.CostCtr3)=69839) AND
((tblMntcCode.MntcType)="Prevent" Or (tblMntcCode.MntcType)="proactive"
Or (tblMntcCode.MntcType)="reactive") AND ((tblWrkOrder.Status) Like
("*mcmp*")) AND ((tblWrkOrder.Created) Between #1/1/2011# And
#2/27/2012#))
ORDER BY tblWrkOrder.CostCtr3, [tblWrkOrder].[Order] & ": " &
[OrdrShrtTxt], [Task] & ": " & [TskShrtTxt];
```

The modern reality is that an SME need not know a lick about the shown code. As an object is built in Access, its SQL code is automatically formed in the background.

Rather than locked in a sealed box, the code is still available to us. It can be viewed in the SQL option of the View icon. This is important because there is something to love about SQL code in the background.

What if for some reason we wanted to distribute the query but not as an Access file? We can by pasting the code in a txt file. The recipient only needs to create a new query, paste the code in the SQL view window and run the query.

Of course, Access is not the only software available to build super tables. The code also makes it possible to transfer what has been built in Access to other software, including “R.”. Because SQL is a universal language, the code can be pasted and run in any software’s functionality for SQL.

3.8. Administration of Tables

The section “Scenarios for Demonstration” of the chapter presented seven steps to develop and manage super tables. The point of the seventh was that the query and hard tables rarely have importance as a one-off or to a single user. Accordingly, we should give thought to how they will be made available to everyone—done-by-one, used-by-many.

For example, the foundation super table serves almost any interest or as a source table to other super tables. Another example is the classifier query for lead craft. It is a table many would want to engage in their analyses.

However, let’s use this place to introduce administration within an Access file. The reason is accentuated by the left-most panel of Figure 3-20. There are 18 tables and queries involved in the chapter’s demonstrations and explanations.

Fortunately, we can organize them as shown in the right pane to the Navigation Options window. The window pops up upon right-clicking in the open space of the objects panel. Notice that the Access file has been organized to track the sequence of the chapter.

In the window we have choices for organizing our objects—Tables and related views, Object type and Custom. They appear in the left-most panel.

By selecting Custom, we can organize our tables and queries as we want. Below the panels of the pop-up window, categories can be added, deleted and renamed. The order of the categories can be established by dragging the titles.

Rather than dedicate a great deal of book space, the readers are advised to query the internet for a YouTube demonstration. Search the internet for “access navigation panel YouTube.”

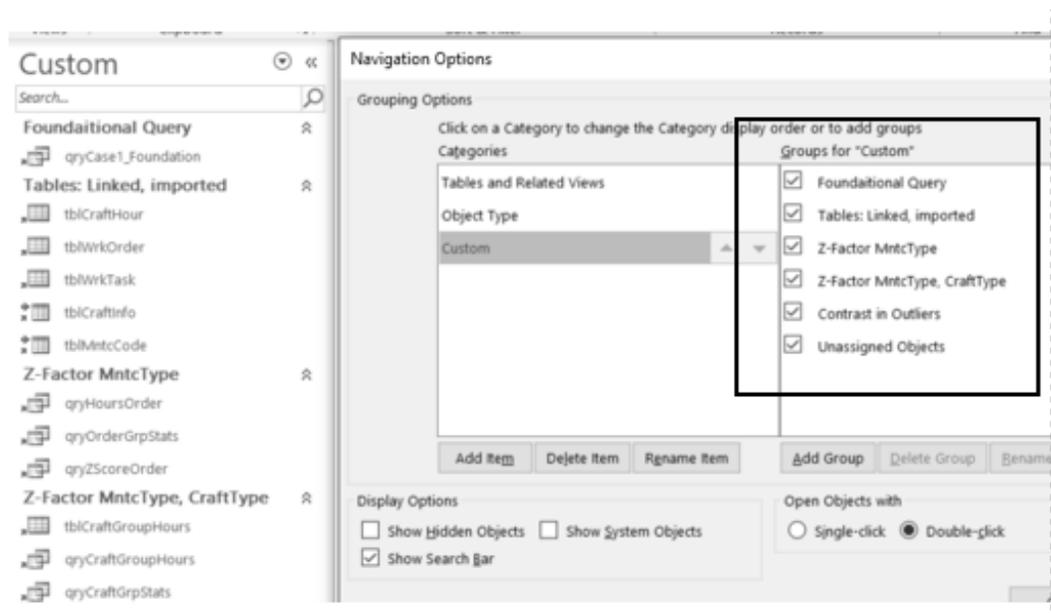


Figure 3-20: Navigation capability to organize the objects of Access.

Bibliography

- Alexander, Michael and Kusleika, Richard. Access 2016 Bible. John Wiley & Son, Inc. 2016. Parts 3 and 4.
- Oesko, Suljan. Pivot Tables In-Depth for MS Excel 2016. Suljan Oesko. 2017.

Chapter 4

The “R” Software in Action

To qualify as data-driven, an asset management organization must acquire a critical-mass of skills in the “R” software as its analytic core. The purpose of this chapter is to hand the skills of “R” off to its readers.

The chapter will begin by explaining how the hand-off of skills will be accomplished. It will next explain the process to install “R” on personal computers and then run a session.

The remainder of the chapter will work through a hands-on case in correlation analysis and brief dabbling in linear regression. Furthermore, the readers can use the case, as a go-by, to work with the data of their own roles.

The data that was formed to demonstrate the methods of the chapter are available in the file ScoreComplexPrep_Dataset.csv. The file is available for download at the website provided in the preface.

The R script which is explained throughout the chapter is also available from the same webpage as the file RSoftwareInAction.R.txt. The extension “.txt” has been added to allow placement on the webpage as a notepad file. The extension must be removed from the file name to make it directly loadable into an R session as explained in Section 3.2.

Additionally, the path element in the R script is the author’s. The reader must replace them with their own path. The cases are flagged as <path> in the code presented throughout the chapter and the book.

4.1. Approach to Reach Critical Mass

The approach to the chapter is to capitalize on the plentiful published literature that draw upon the ”R” software to explain and demonstrate every type of statistical analysis. Just as important, the same body of literature explains the “R” code to each topic of analysis.

The explanation is not intended to be an “all there is to know” explanation of the principles and practices of the demonstrated analyses. Instead, the demonstration is limited to working with “R” while explaining each introduced analytic to a critical mass of depth. The overarching objective of the chapter is to cause its readers to “run around in R.” Throughout the explanation, readers will be introduced to the most commonly and frequently occurring code in “R.”

The preface promised to present what the reader “needs to know” rather than “all there is to know.” The “all there is to know” in the case of the chapter are explained by Chapters 6 and 7 of the book, “Discover Statistics with R, [Fields and Miles, 2012].

Throughout the book, the Fields and Miles text will be referenced for the “all there is to know” of statistical principles, correlation analyses, regression and ANOVA. Other texts will be referenced for time series, survival-hazard and hidden variables analyses.

The published example of the chapter is intended to be a standard methodology. It is exactly the steps we would take to measure and test the correlation and linear fit of the variables that are normal to any enterprise operation. As the chapter progresses, the reader can take what is being presented and put it to work in the evaluation of their own operation.

4.2. “R” Installation and Session

There need not be an organization, rather than grassroots, initiative to acquire the “R” software. This is because “R” is freely available for download at <https://www.r-project.org/>.

Also, at the site are instructions for download, as well as manuals for working with “R.” Be aware that there are YouTube videos to demonstrate the download and installation process. Furthermore, friendlier texts on the subject of working with “R” are listed in the chapter’s bibliography.

As employees, our computers are integrated within an IT system. Consequently, we will likely need the system administrator to our computer to conduct the download and installation of “R.” The administrator will also be required to load packages into the installed software. Packages will be explained later in the section. In all cases, the administrator’s task is minor and quick.

Like all modern software, a session is opened by clicking its icon. Figure 4-1 shows the opening view—the console.

The nature of the console is that each line of code runs upon pressing the Enter key. At the end of a completed function, the associated output is returned in the console. It is not possible to place a set of commands and then get the collective output. In other words, the console is not a friendly place to work; so we don’t.

The hugely more workable view is the script window. It is shown in Figure 4-2. As can be seen, we can start a new script or open an existing one.

Upon clicking the Open Script option, we navigate through our local or server directory to upload the script we want. If we are writing a new script, upon selecting New Script, a blank script view will open. We ultimately save the script as a new file. The procedures are identical to what we are all accustomed to.

```

RGui (64-bit)
File Edit View Misc Packages Windows Help

R Console

R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

```

Figure 4-1: The console view of “R.”

The Figure 4-2 shows the action of opening a script file that was previously developed and saved. A little geek-talk here. A script file is distinguished by its .R extension—seen in the window frame. If we further extend the script name with .txt we would get a Notepad file of the code.

The script window is where we can write, edit and run code instead of working in the console view. In contrast to the console, we can work in a fully flexible manner. We also control when we want the output to be generated relative to the command code.

Along with the code of the analysis, the script can also serve as a document of explanation and paper trail. Notice the hash marks in the code. They convert code to unexecuted text and, thus, allow us to place notes and explanations throughout the script.

We also see the hash marks preceding some command codes. For example, hash marks are placed before the `install.packages` function. The placement prevents a one-time command from running indiscriminately as part of every full run. At the same time, we make the command in the working code a paper trail to the code.

There are two ways to run the code. First, highlight the code to be run. Thence, press the run icon on the icon bar at the top of the “R” window for script. The other is to place the cursor in the code line and press the `ctrl-r` keys. Another way is to highlight a piece of code and press the `ctrl-r` keys.

Chapter 2 explained that everything in “R” is done with functions. In turn, the functions are available from their resident packages. There were approximately 10,000 packages as of 2020.

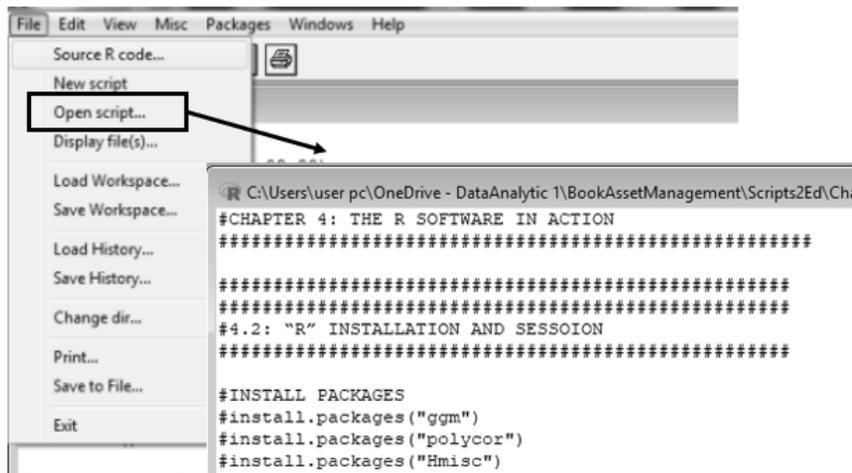


Figure 4-2: Script view, were we do our work.

The most commonly called functions are automatically installed with the initial installation of “R” and automatically opened in all sessions. Others are installed as they are needed for the analytics at hand.

Seven packages are required for the analyses demonstrated in this chapter. The code to call them is the `install.packages` function applied as follows:

```

install.packages("ggm")
install.packages("polycor")
install.packages("Hmisc")
install.packages("psych")
install.packages("ggplot2")
install.packages("r1m")
install.packages(MASS)
install.packages(pastecs)
  
```

An important detail can be now introduced. “R” is case sensitive. We will get an error message if we violate case. Upon an error message, checking for case is part and partial to checking our code for spelling and punctuation vis-a-vis the go-by—in geek, checking our syntax. Furthermore, we can check each function as we code it and then move to the next.

As we work through the go-by literature to an analytic, the packages will be identified as the code is being presented. Often the packages are arcane. For example, the `ggm` package is required for “graphical Markov models.” However, we have no need to know what that means or what that is—only that we must call it up.

Figure 4-3 shows the process to install a package. For example, the `install.packages` function with reference to `ggm` in the brackets is highlighted and run. The window appears

offering choices of mirror servers located around the world. We pick one, click OK and the package will be automatically downloaded and installed in the “R” software. As mentioned, most of us would turn to our system administrator for the task.

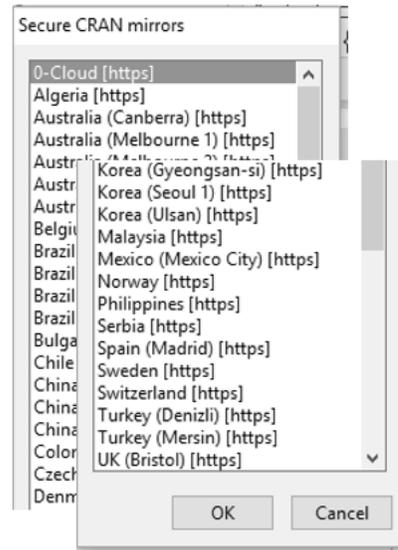


Figure 4-3: Packages are downloaded and installed from a mirror server.

Once installed, it is necessary to open the package in a session. A package only needs to be installed to “R” when it is used for the first time. Thence, for any session using the package, it is only necessary to open it once upstream to its role. The shown `library` function does that for each package with the code:

```
#LIBRARIES TO LOAD
library(ggm); library(Hmisc); library(polycor); library(psych)
library(ggplot2); library(r1m); library(MASS); library(pastecs)
```

4.3. Basics of “R” Demonstrated

Packaged as correlation and regression analyses, the section will introduce the most commonly observed coding. As they emerge in the demonstration, the code will be explained. In this way the reader will become comfortable with following published explanations as actionable go-byes to apply in asset management.

The demonstration will begin with introducing the case and the questions we want to ask and answer with the data. Thence, it will explain how to load data into the session and gain an initial “know-thy-data” perspective of it. Next correlation and partial correlation

analyses will be introduced and conducted. Finally, to demonstrate fundamental principles and code to setting up most types of models, the chapter will step lightly into linear regression. Chapter 7 will dive deeply into linear regression.

4.3.1. Demonstration Case

Suppose we are given the dataset (ScoreComplexPrep_Dataset) charted in Figure 4-4. They are the cross plots of complexity to score and preparation to score. The straight line is the linear regression fitted to the cross-plotted points. The correlation coefficient is included.

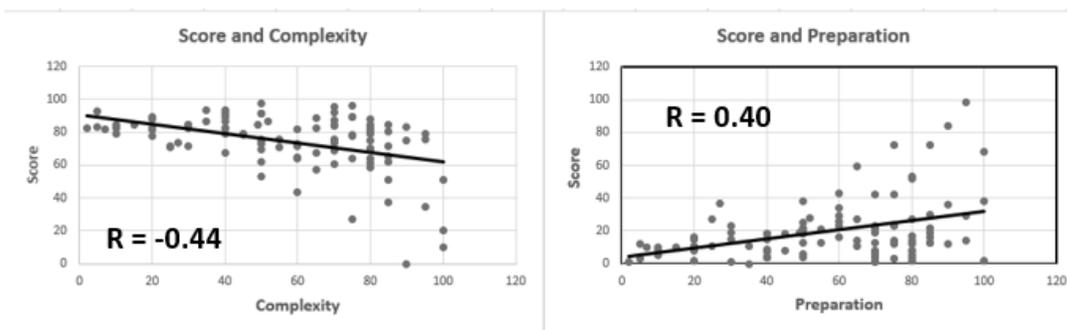


Figure 4-4: Cross plots of paired variables with a fitted line.

Obviously, the example is not for any specific type of operation. However, it is directly relevant to any operation whenever our interest is the correlation between variables. We would load our table as directed for the example and change the names of the variables in the code. In other words, the example can be used as our go-by or tool.

In fact, there is a case in asset management literature that explores the comparative correlations of maintenance task types to injuries. The example should be applied to confirm for ourselves the inferences made so confidently in the literature but without the same inspection.

Answers to the following questions will emerge as the case unfolds:

- What is the statistical nature of the data—average, median, spread, etc.?
- Is the data normally distributed?
- Are complexity and preparation significantly correlated to score?
- How much of the variance in score is shared by complexity and preparation?
- Do the correlations hold if we isolate what is unique between each combination—partial correlation?
- Are complexity and preparation significantly predictive of performance?

```
head(TaskData)
#INSPECT MAKEUP OF VARIABLES
str(TaskData)
```

The head function returns a small table representative of the parent table. We can see that one, Code, is a house-keeping variable of no use to us. We can also see that we could run our analytics while controlling for ProdUnit, which we will not in this example.

From the str function we get further descriptions of the variables. We see that we have what is called by “R” a “data frame” with 103 observations (records). Three of the variables are integers, one a numeric and one a factor. For the factor variable, ProdUnit, we see that the categories are UnitA and UnitB. For all, we see the first some records as space allows.

```
> #INSPECT FIRST SIX CASES
> head(TaskData)
  Code Prep Complex Score ProdUnit
1    1    4  86.298   40   UnitB
2    2   11  88.716   65   UnitA
3    3   27  70.178   80   UnitB
4    4   53  61.312   80   UnitB
5    5    4  89.522   40   UnitB
6    6   22  60.506   70   UnitA
> #INSPECT MAKEUP OF VARIABLES
> str(TaskData)
'data.frame':  103 obs. of  5 variables:
 $ Code      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Prep      : int  4 11 27 53 4 22 16 21 25 18 ...
 $ Complex   : num  86.3 88.7 70.2 61.3 89.5 ...
 $ Score     : int  40 65 80 80 40 70 20 55 50 40 ...
 $ ProdUnit  : Factor w/ 2 levels "UnitA","UnitB": 2 1 2 1
```

Output 4-1: Head and summary views the data.

The second dimension of insight gives us a statistical overview of our dataset rather than the basic “what it is” presentation. Two such functions are summary and describe. They are coded as follows and the outputs are shown in Output 4-2:

```
#DESCRIPTIVE STATISTICS
summary(TaskData)
#ADDITIONAL DESCRIPTIVE STATISTICS
describe(TaskData)
```

The summary function provides shape-type insight. They are min-max, mean, median and quartiles for integer and numeric variables. The categories and counts are returned for the factor variable.

The describe function provides additional information. In contrast to the summary function, it provides standard deviation, mean trimmed of extremes, mean absolute

deviation from median, skew and kurtosis as measure of the variable from the normal distribution, range and standard error to the mean.

```
> #DESCRIPTIVE STATISTICS
> summary(TaskData)
      Code      Prep      Complex      Score      ProdUnit
Min.   : 1.0    Min.   : 0.00    Min.   : 0.056   Min.   : 2.00   UnitA:51
1st Qu.: 26.5   1st Qu.: 8.00    1st Qu.:69.775  1st Qu.: 40.00   UnitB:52
Median : 52.0   Median :15.00    Median :79.044  Median : 60.00
Mean   : 52.0   Mean   :19.85    Mean   :74.344  Mean   : 56.57
3rd Qu.: 77.5   3rd Qu.:23.50    3rd Qu.:84.686  3rd Qu.: 80.00
Max.   :103.0   Max.   :98.00    Max.   :97.582  Max.   :100.00
>
> #ADDITIONAL DESCRIPTIVE STATISTICS
> describe(TaskData)
      vars  n mean  sd median trimmed  mad  min  max  range  skew
Code      1 103 52.00 29.88 52.00 52.00 38.55 1.00 103.00 102.00 0.00
Prep      2 103 19.85 18.16 15.00 16.70 11.86 0.00 98.00 98.00 1.95
Complex   3 103 74.34 17.18 79.04 77.05 10.75 0.06 97.58 97.53 -1.95
Score     4 103 56.57 25.94 60.00 57.75 29.65 2.00 100.00 98.00 -0.36
ProdUnit* 5 103 1.50 0.50 2.00 1.51 0.00 1.00 2.00 1.00 -0.02
      kurtosis  se
Code          -1.24 2.94
Prep           4.34 1.79
Complex        4.73 1.69
Score          -0.91 2.56
ProdUnit*     -2.02 0.05
```

Output 4-2: Two statistical views of the dataset.

Finally, we want graphic insight. Figure 4-4 was a traditional preliminary insight. We will demonstrate two additional cases. First is the case of a pairs-panel as shown in Output 4-3. Second is a graphical assessment of the distribution of the variables as a normal distribution. It is shown in Output 4-4.

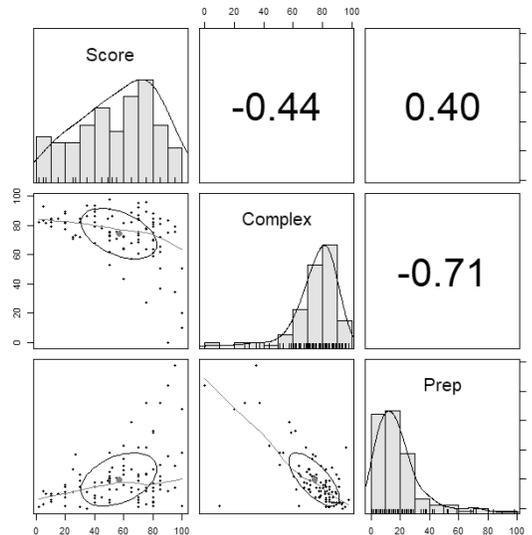
First let's look at the graphic from the `pairs.panel` function of the `psych` package for a tremendous amount of insight. In it, we can inspect the shape or distribution of each variable in the dataset. We can inspect the cross-plot relationship of each variable with all other variables. The fitted smooth line gives a sense of pattern to the cross plots. The oval shape is called the correlation ellipse, the more elongated the greater correlation. The large dot indicates the mean value of plot points.

The code to return the pairs panel of Output 4-3 is as follows:

```
#PAIRS PANELS
pairs.panels(TaskData[c("Score", "Complex", "Prep")])
```

It is good practice to confirm that our data is normally distributed. Doing so is a mandatory step to evaluate the truthfulness of an analytic. If the finding is not a normal distribution, we use an alternative method to the analytic or transform the variables within the analytic.

We were given the cross-plot and line fit (Figure 4-4) of Complex and Prep plotted to Score. Upon inspection, we can see that the points do not fall in a consistent band around the plotted linear fit. This was the first clue that the data are not normally distributed. Furthermore, the histograms of Output 4-3 do not look like normal distributions.



Output 4-3: Pairs panel perspective of the numeric variables in the dataset.

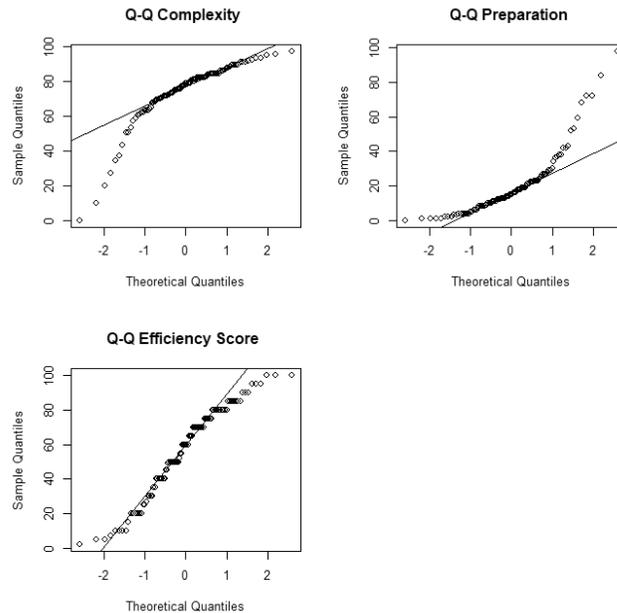
We can test normalcy with graphic insight. However, we should note that the `shapiro.test` function can be used to test if the data is significantly different from the normal distribution.

The graphic method is to build and view a q-q graphic as shown in Output 4-4. If the points of a variable fall along its line, the data is normally distributed.

In the following code to generate the figure there are several basic elements of coding to recognize:

```
#Q-Q GRAPHIC FOR NORMALITY
par(mfrow = c(2, 2))
anx<- qqnorm(TaskData$Complex, main = "Q-Q Complexity ");
  qqline(TaskData$Complex)
rev<- qqnorm(TaskData$Prep, main = "Q-Q Preparation ");
  qqline(TaskData$Prep)
exm<- qqnorm(TaskData$Score, main = "Q-Q Efficiency Score ");
  qqline(TaskData$Score)
par(mfrow = c(1, 1))
```

First, how did we get the three charts in one figure? The answer is the `par` function in the first line and its argument `mfrow = c(2,2)`). The code creates a 2-by-2 matrix of graphics. If we change the `c()` element we can create any combination of rows and columns.



Output 4-4: A plot point of a normal distributed variable will fall along the straight line.

We have built an output sandwich with the `par` function. Note that the `par` function appears again at the end of the block of code. By coding `c(1,1)`, any graphic after the line of code will present in a single frame.

We can also see layered charting in action. Inspect the line of code beginning just after `anx<-`. In it is the code to generate the q-q graph for the Complex variable.

The code line includes the `qqnorm` and `qqline` functions. Both are individual graphs of the subject variable to the line of code.

If we highlighted and ran the `qqnorm` code, we would get the plotted of points. If we did the same with `qqline` code, we would get a straight line. It represents how a plot of data with the mean and variance of the data should form as quantiles if it were a normal distribution.

As a coded command, the `qqline` element overlays the straight line on the plotted points. In this case, we can readily conclude the data is not normally distributed.

In the output we receive once again the coefficient and p-value to the pair. Additionally, we now receive the upper and lower limits to 95 percent confidence. If we want to explore the limits for the computed p-value, we would do so by the setting the `conf.level` argument in the `cor.test` function to the confidence level equal to one minus the p-value.

There is another fundamental point to make. Notice that “0” does not appear between the limits of a 95 percent confidence interval. We would expect this because the computed p-value is 5 percent or less.

```
Pearson's product-moment correlation

data: TaskData$Complex and TaskData$Score
t = -4.938, df = 101, p-value = 3.128e-06
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.5846244 -0.2705591
sample estimates:
      cor
-0.4409934
```

Output 4-8: Confidence limits to the Complex-Score pair.

4.3.5. Partial Correlation Analysis

Now is the time to introduce two types of correlation and the difference. They are correlation (R) and correlation squared (R²).

As previously defined, correlation is a measure of the extent to which two variables move together with respect to their own means. Correlation-squared is the extent that the variance in one is shared by the other.

The latter of the two is computed as `cor` squared. The code using Pearson and Spearman is simply as follows and the output is shown in Output 4-9.

```
#R SQUARED
cor(TaskData2, method="pearson")^2
cor(TaskData2, method="spearman")^2
```

```

> #R SQUARED
> cor(TaskData2, method="pearson")^2
      Score  Complex  Prep
Score  1.0000000 0.1944752 0.1573873
Complex 0.1944752 1.0000000 0.5030345
Prep    0.1573873 0.5030345 1.0000000
> cor(TaskData2, method="spearman")^2
      Score  Complex  Prep
Score  1.0000000 0.1637126 0.1224264
Complex 0.1637126 1.0000000 0.3868459
Prep    0.1224264 0.3868459 1.0000000

```

Output 4-9: Correlation squared for shared variance.

Both measures of correlation can be misleading because pairs hide their correlation to a third or more variables. Therefore, we need to determine the unique correlation between two variables rather than disregard the correlation shared with other variables. The concept is shown in Figure 4-5.

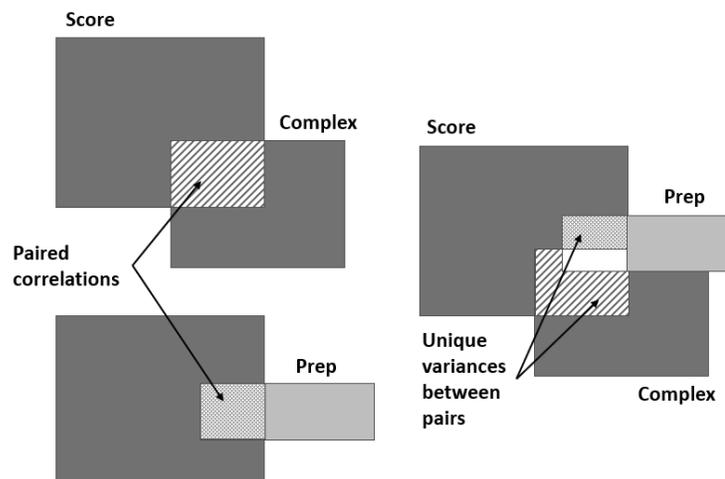


Figure 4-5: Full- and partial-correlation between two variables.

To the left we see the comparative paired correlations between Complex and Score, and Prep and Score. Shown is what the calculation of correlation and correlation-squared have been to this point. It is also the typical perspective of the meaning of correlation.

To the right in the figure, we can see the lurking issue. Some part of the correlation of the paired set of Score to Complex is shared with Prep.

Therefore, correlation and correlation-squared can be misleading if we disregard associations with other variables. There are, of course, functions for that. We want to determine what is called partial correlation and confirm its significance.

Let's look at the partials of the Score-Complex pair. The code is as follows and the output is presented in Output 4-10.

```
#PARTIAL CORRELATION TO S
cor(TaskData2, method="pearson")
pcExAn<- pcor(c("Score", "Complex", "Prep"), var(TaskData2))
pcExAn
pcExAn^2
pcor.test(pcExAn, 1, 103)
```

The `cor` line of the code returns the full correlation that was generated with the same function earlier in the chapter. The `pcor` function will return a partial correlation. The output of the `pcor` function is assigned to the `pcExAn` object.

The partial correlation is returned when the object is called up by the next line of code. In the figure it is apparent that the full- and partial-correlations are very different. The true correlation is far weaker than we would be led to believe with full correlation. In fact, it is below the rule-of-thumb threshold of 30 percent as a strong correlation.

```
> #PARTIAL CORRELATION TO S
> cor(TaskData2, method="pearson")
      Score      Complex      Prep
Score  1.0000000 -0.4409934  0.3967207
Complex -0.4409934  1.0000000 -0.7092493
Prep    0.3967207 -0.7092493  1.0000000
> pcExAn<- pcor(c("Score", "Complex", "Prep"), var(TaskData2))
> pcExAn
[1] -0.2466658
> pcExAn^2
[1] 0.06084403
> pcor.test(pcExAn, 1, 103)
$tval
[1] -2.545307

$df
[1] 100

$pvalue
[1] 0.01244581
```

Output 4-10: Partial correlation to Score-Complex pair.

Mathematically we know that the before and after correlation-squared will also be even more disparate. When calling the code line `pcExAn^2`, the correlation-squared measure is returned.

Now let's look at the `pcor` function. It has two arguments. The second tells the function from which table the variables are taken from. The first shows three variables in the `c` function. The first two are the variables of interest. The third is the variable for which there is shared correlation and the one we want to exclude from the partial-correlation.

However, know that the `pcor` function does not limit us to sets of three variables. The function allows us to engage any number of variables. The first two in the list are always the partial correlation variables.

The function `pcor.test` returns three insights. The important one is `$pvalue`. It shows that the partial correlation holds. It is less than the 5 percent to judge the correlation as significant. However, because it is substantially greater than the p-value (<0.001) of the full correlation, the significance of correlation is much less.

To determine the confidence limits to the partial correlation, there are three arguments to the `pcor.test` function. The first is the object model `pcEXAN`. The second is the number of controlled variables; "1". And the third is the number of observations.

The Excel table of Figure 4-6 summarizes the findings for the respective pairs. Complex has a significant unique correlation to Score performance, whereas Prep does not.

Type	Exam-Anxiety			Exam-Revise		
	R	R^2	P	R	R^2	P
Full	-0.44	0.19	<0.001	0.40	0.16	<0.001
Partial	-0.24	0.06	0.012	0.13	0.02	0.18

Figure 4-6: Findings from full- and partial-correlation analysis.

If we were a subject matter expert in the core task of the operation, we would ponder why what seems intuitively correlated to us is not. We would have discovered to seek other variables with respect to preparation because not all approaches appear to be equally effective.

4.3.6. Correlation Compared to Regression

The section will demonstrate regression analysis in contrast to correlation analysis. It is an opportunity to introduce the structure of "formulas" that appears typically in most analytic models.

There are multiple steps to build and evaluate linear regression. Here we will look only at the first step which is to determine the best predictor variables to the model. The step may begin with the correlation analyses of variables as shown throughout the chapter. Thence, we would evaluate the model with different configurations of variables. Finally, we would retain the predictive variables that are significant to the outcome variable.

Although not introduced here, a number of steps take place to discover and work around issues to the model until it is judged as valid. Although not introduced in this chapter, the steps are explained in Chapter 7.

As mentioned earlier, we must not forget that the coefficients of regression and correlation are different measures. A correlation coefficient indicates the extent to which two variables move together with respect to their own means. A regression coefficient indicates the relationship of a unit change in a predictor variable to the outcome variable. However, the issues of significance and confidence are equally involved.

A reason for demonstrating regression is that, as models, they are set up in much the same way for most types of models and by most software. The `help(lm)` code would take us to the site to present and explain the following code:

```
lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE,
   x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL,
   offset, ...)
```

Just as for correlation analysis, the `lm` function is set up by the choices we make for its arguments. Explanations and examples of the options are readily available from the internet and will not be presented here. However, as for many functions, we rarely touch most of the arguments because the defaults typically apply. Accordingly, the shown code may reduce to `lm(formula, data)`.

The model to this example reduces to the following:

```
#MODEL OF MAIN EFFECTS
TaskPred <- lm(Score ~ Complex + Prep, TaskData)
summary(TaskPred)
confint(TaskPred)
```

We are left with two arguments in the model function, `lm`. One is the reference to the source table of data; `TaskData`. The other is the model or formula of variables.

The code is assigned to an object we have named `TaskPred`. It is our model. In it, we will learn the strength of relation the predictor variables, `Complex` and `Prep`, have with the `Score` variable.

In the model, the syntax “`~`” is the equivalent to “`=`.” To the right of the symbol are the predictor variables. To the left is the outcome variable.

The `summary` function, with our model as its argument, calls up the model shown in Output 4-11. Also shown, the `confint` function, with the model as its argument, returns the 95 percent confidence interval.

Chapter 5

Layered Charting to Know Thy Data

The foundational achievement of being data driven is to “become at one” with our operational data. Becoming at one with our data has two stages. First is to “know thy data.” Thence, second is to gain and maintain the truthfulness, entirety and accessibility of our data.

This chapter will speak to know-thy-data. The next, Chapter 6, will speak to truthfulness, bad data, and Chapter 8 will speak to the entirety and accessibility of data. Chapter 7 will introduce, explain and demonstrate the data analytics we would most likely use to test and cleanse data for truthfulness.

Chapter 2 introduced layered charting as far beyond what has been our standard of the past, Excel charts. Layered charting is done with the ggplot2 package in the R software.

Because of what it makes possible in the exploration of our data, the chapter is as much about layered charting as it is about know thy data. The explanations and demonstrations to know thy data will likewise be explanations and demonstrations for layered charting and, thus, ggplot2.

With the examples and demonstrations of the chapter, the reader will be able to substitute in the variables of their maintenance and reliability operations. Once converted to actual cases, any script can be pulled up and run with our periodically refreshed operational data. Consequently, once our exploratory scripts are written initially, we can routinely call up a huge amount of insight in a matter of moments.

As it is throughout the book, the software R will be used to demonstrate and provide templates to the methodologies to know our data. Accordingly, the chapter is written with the expectation that its readers have read Chapter 4 and, thus, know how to run around in R as instructed in this chapter.

The data that was formed to demonstrate the methods of the chapter are available in the Excel file titled, SkillsForCareerSecurity_Datasets.xlsx. The data file is available for download from the webpage provided in the preface.

The R script titled, LayeredChartToKnowThyData.R.txt, with which the methods and templates throughout the chapter are accomplished is also available from the same webpage. The extension .txt has been added to allow placement on the webpage as a notepad file. The extension must be removed from the file name to make it directly loadable into an R session as explained in Section 4.2.

Occasionally, a block of code specifies a path to source data and saved outputs. The cases are flagged as <path> in the code. The reader must replace the code with their own path.

Section 4.2 explained that packages typically need to be loaded to an R session. They are loaded with the `library` function if previously installed with the `install.packages` function.

It is good practice to list the packages at the beginning of an R script rather than load them at the point of necessity. We can load them collectively at the beginning of the session by running the following block of code:

```
#LIBRARIES TO LOAD
library(xlsx); library(mice); library(ggplot2)
library(qqplotr); library(ggm); library(Hmisc)
library(polycor); library(rlm); library(MASS)
library(ggpubr); library(psych); library(nlme)
```

5.1. Inspect for Missing Data

A super table often entails many variables and thousands of records. Possibly salted throughout are missing data: empty cells. However, manually searching for them through many of thousands of cells can be laborious and without assurance of spotting all cases.

Instead, we can engage the triad of grassroot software—Access, Excel and R—to determine which variables have missing data and which records across the variables include missing data. The triad, as an integration of software, was explained in Section 2.2.2.

The steps to search out missing data with the triad are as follows:

1. Import the super table into Excel from Access.
2. Inspect the table for flags to missing data.
3. Import the super table into R from Excel.
4. Generate a plot and table summarizing the cases of missing variables.
5. Generate a table of records with missing data in Access.
6. Omit records with missing data from the super table—optional.

Ultimately, there will be decisions for omitting or retaining the found cases. It is possible that a missing case will prevent or undermine a computation to a sought insight. However, we must remember that omitting records leaves us with less data for envisioned modeled insights. Accordingly, we may choose to omit missing data at the level of generating the insight deliverable for which inclusion prevents or distorts a computation or model.

However, further investigation subsequent to the simple discovery while coming to know thy data revealed the problem. The plants the mind numbing, intricate, laborious methods to fill out timesheets drove supervisors to record hours only to the largest work orders.

This discovery emerged at a time when a big investigation was being launched to figure out how to reduce the cost of a particular type of big job for which there were some costly workorders. Furthermore, the costs for equivalent work orders were not roughly equal. The excessive, erratic cost was not the reality. Missing data created the perception, and it became reality.

Step 6: Omit records with missing cases from the super table--Optional. We could return to the source tables and delete the records with missing data. However, some records in the super table occur as the result of joining subtables. An example is the use of translation tables to cleanse data as explained in Section 3.3.2. An empty cell in the super table flagged bad data in a source table.

Removing records from the tables we have extracted from our operating systems is normally not an advised practice. We always want to know that the subtables to the super table are as found in their home operating systems.

We can surgically omit records from the super table in Access. Output 5-4 showed the result of using the plot or table shown in Output 5-3 to filter from the super table all records with missing data. That was the purpose of the Access code.

Now we can omit records from the super table by changing the `IS NULL` to `IS NOT NULL` for each group. Each group will then return only records with full data. If we wanted to omit all records with missing data, we can delete the `OR` rows in the design grid and insert `IS NOT NULL` in the Criteria row for each variable we know to have missing data.

We have inspected for missing data. With the inspection we have decided how to treat it. At this point we should know that to deal with missing data we can simply include the argument `na.rm = TRUE` in our R functions and missing data will be ignored in the conduct of a model or calculation of the function.

5.2. Visual and Statistical Inspection

Now that we know of any missing data in our super table, we need to ask other questions of our single- and multiple-dimensional variables. What are the variable types? What are the distributions of the variables with respect to center and spread? Are there distorting heterogenous subsets within the variables? How correlated are the variables to each other? There will also be questions that will only occur to us as we through our data.

Modern-day analytics allow us to visualize our data rather than be limited to numerically presented statistical reports. As the old saw goes, “a picture is worth a thousand words.” Section 2.2.3. introduced layered charting. The package, `ggplot2`, within R was introduced as the graphic software with which to build layered insight.

Section 4.3.3. demonstrated some of the base graphic capability of R. This chapter and future chapters will largely present graphics with the `ggplot2` package.

This chapter will explain and demonstrate `ggplot2` as templates to methods to inspect and know our data. However, readers are encouraged to at least read Chapter 2 of the book, “`ggplot2`; Elegant Graphics for Data Analysis,” second edition by Hadley Wickham. Analysts who aspire to be new age, thus, top drawer in their work should read the first eight chapters.

This chapter will resist the urge to put glamor and polish on the graphs it introduces as methods. The possibilities are immense. It is left to the reader to apply the full scope of the Wickham book.

Accordingly, the templates of this chapter will stick to the core code to each. In this way, the meat of the graphic methods to know our data will remain highly visible to the demonstrated explanations.

A philosophy of R is that every package must be fully explained and that explanations must be available on the internet. Furthermore, all explanations must be demonstrated with go-by examples from which we can steal code and each example must be accompanied with a dataset with which we can try the code ourselves. This section will use an R-provided dataset (`mpg`) to explore the miles per gallon data with respect to models, drive, transmission and six other characteristics.

The dataset is relevant and interesting to all of us as car owners. It is commonly used in posted examples to `ggplot2`. More importantly, it has characteristics which are like maintenance and reliability data. The data of our CMMS and other sources, like the `mpg` dataset, contain numeric variables such as hours, dollars, quantities and dates. However, like the example set, many more of the variables are categorical such as cost center, priority, maintenance type, order by craft type, crafts and failure codes.

The steps we will take for the visual and statistical assessment of our dataset are as follows:

1. Load and survey the data.
2. Test numeric variables for normal distribution.
3. View correlations between variables.
4. View centrality and spread of numeric variables.

5. View characteristics of categorical variables,
6. View variables over time.

5.2.1. Load and Survey the Data

Of course, we must first load the dataset into our R session. We will use the R-provided mpg dataset. It can be pulled into the session by entering and running the code `data(mpg)`.

However, the reality is that our maintenance and reliability data will never come from within an R package. It will be data we have built in Access as a super table and placed in an Excel file for dissemination. Therefore, in line with our reality, the R-provided dataset has been stored in our Excel file as the `mpgDataSet`.

However, note that we can also load data directly from Access. It is left to the reader to seek the method from the internet.

Accordingly, just as for any maintenance and reliability super table, we would pull the `mpgDataSet` dataset into the session from its Excel file with the following code:

```
#Load table from Excel and assign to object
mpgKtd<- read.xlsx(
  "C:\\<path>\\SkillsForCareerSecurity_Datasets.xlsx",
  sheetName="MpgDataSet", header=TRUE)
```

In the code we can see that the dataset is located as the worksheet `mpgDataSet` in the Excel file `SkillsForCareerSecurity_Datasets.xlsx`. We are using the `read.xlsx` function because our data is in an `xlsx` file. We would use other similar functions for other file types such as `csv` and `dat`. The code assigns the dataset to the data frame object named `mpgKtd`.

Now that the dataset has been loaded, we should survey it in various tabular formats. Chapter 4 introduced them. In their coded form, they are as follows:

```
##Survey views of data
head(mpgKtd)
str(mpgKtd)
summary(mpgKtd)
md.pattern(mpgKtd)
describe(mpgKtd)
```

If we wish to see what the data looks like in its table form, the `head` function will return the first six rows of the `mpgKtd` dataset. If more or less rows are desired, we would code the function as `head(data, n =)` where `n` is the number of rows.

If we want to view the entire table, we would merely highlight and run `mpgKtd`. If we want to view rows at the end of the dataset, we would use the function `tail`.

The function, `str`, returns basic information on the makeup of the dataset and its variables. Output 5-6 shows what is returned.

```
> str(mpgKtd)
'data.frame': 234 obs. of 11 variables:
 $ manufacturer: Factor w/ 15 levels "audi","chevrolet",...: 5 5 5 5 5 5 5 5 5
 $ model       : Factor w/ 38 levels "4runner 4wd",...: 10 10 10 10 10 10 10 10 10
 $ displ      : num 1.6 1.6 1.6 1.6 1.6 1.8 1.8 1.8 2 2.4 ...
 $ year       : num 1999 1999 1999 1999 1999 1999 ...
 $ cyl        : num 4 4 4 4 4 4 4 4 4 ...
 $ trans      : Factor w/ 10 levels "auto(av)","auto(l3)",...: 9 3 9 9 3 9 4 .
 $ drv        : Factor w/ 3 levels "4","f","r": 2 2 2 2 2 2 2 2 2 ...
 $ cty        : num 28 24 25 23 24 26 25 24 21 18 ...
 $ hwy        : num 33 32 32 29 32 34 36 36 29 26 ...
 $ fl         : Factor w/ 5 levels "c","d","e","p",...: 5 5 5 4 5 5 5 1 4 5 .
 $ class      : Factor w/ 7 levels "2seater","compact",...: 6 6 6 6 6 6 6 6 6
```

Output 5-6: The dataset viewed through the `str` function.

We are informed that the `mpgKtd` dataset is data frame and of the counts for records and variables. Below that we can inspect the names of the variables, their types and their first some records.

The definitions of the variables are obvious. However, the variable, `fl`, is not. It is fuel type.

We can see, upon loading, that the character variables (e.g., `model`) of the dataset have been appropriately interpreted by R as factor variables. For them, we are informed of the categories or levels to each.

There are of course other types of variables that do not occur in the data frame. Furthermore, the type of any one variable can be converted to another. Functions are available for all conversions that make sense.

For example, what if we wanted to convert the cylinder variable to a factor from numeric. We would use the `as.factor` function. If already a factor, we could use the `as.numeric` function to convert to a numeric variable. If we wanted to change the factor variables to character, we would use the `as.character` function.

The function, `summary`, provides information of which some overlap the `str` function and some are additional to `str` function. The output is shown in Output 5-7.

For the numeric variables, the `summary` function provides statistics for centrality and spread. The statistics of centrality are mean and median. The statistics of spread are the second and third quartiles and min-max.

For the factor variables, the function returns lists and counts for each category. However, if the categories exceed six, we only get summary information on the six with the greatest counts. All else is lumped as “other.”

The `md.pattern` function was demonstrated in the previous section. It would show no missing data in the dataset if we ran it here.

```
> summary(mpgKtd)
  manufacturer      model      displ      year
dodge      :37   caravan 2wd      : 11   Min.    :1.600   Min.    :1999
toyota     :34   ram 1500 pickup 4wd: 10   1st Qu.:2.400   1st Qu.:1999
volkswagen:27   civic              : 9   Median  :3.300   Median  :2004
ford       :25   dakota pickup 4wd : 9   Mean    :3.472   Mean    :2004
chevrolet  :19   jetta              : 9   3rd Qu.:4.600   3rd Qu.:2008
audi       :18   mustang            : 9   Max.    :7.000   Max.    :2008
(Other)    :74   (Other)            :177

  cyl      trans  drv      cty      hwy      fl
Min.   :4.000  auto(14) :83  4:103  Min.   : 9.00  Min.   :12.00  c: 1
1st Qu.:4.000  manual(m5):58 f:106  1st Qu.:14.00 1st Qu.:18.00 d: 5
Median :6.000  auto(15) :39  r: 25  Median :17.00 Median :24.00 e: 8
Mean   :5.889  manual(m6):19      Mean :16.86  Mean  :23.44  p: 52
3rd Qu.:8.000  auto(s6) :16      3rd Qu.:19.00 3rd Qu.:27.00 r:168
Max.   :8.000  auto(16) : 6      Max.   :35.00  Max.   :44.00
          (Other) :13

  class
2seater : 5
compact :47
midsize  :41
miniivan :11
pickup   :33
subcompact:35
suv      :62
```

Output 5-7: The dataset viewed with the function, summary.

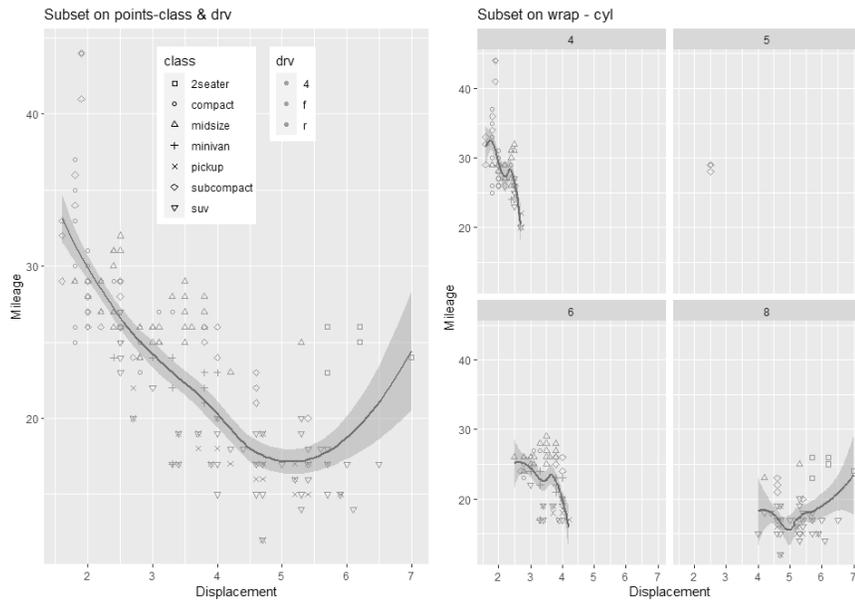
The `describe` function provides additional insight and, of course, some that overlap with the previous functions. As shown in Output 5-8, in contrast to the `summary` function, it provides standard deviation, mean trimmed of extremes, mean absolute deviation from median, skew and kurtosis as measure of the variable from the normal distribution, range and standard error to the mean.

```
> describe(mpgKtd)
  vars  n  mean  sd median trimmed  mad  min  max  range
manufacturer* 1 234  7.76  5.13  6.0  7.68  5.93  1.0  15  14.0
model*        2 234 19.09 11.15 18.5 18.98 14.08  1.0  38  37.0
displ         3 234  3.47  1.29  3.3  3.39  1.33  1.6  7  5.4
year         4 234 2003.50 4.51 2003.5 2003.50 6.67 1999.0 2008 9.0
cyl          5 234  5.89  1.61  6.0  5.86  2.97  4.0  8  4.0
trans*       6 234  5.65  2.88  4.0  5.53  1.48  1.0 10  9.0
drv*         7 234  1.67  0.66  2.0  1.59  1.48  1.0  3  2.0
cty          8 234 16.86  4.26 17.0 16.61  4.45  9.0 35 26.0
hwy         9 234 23.44  5.95 24.0 23.23  7.41 12.0 44 32.0
fl*        10 234  4.63  0.70  5.0  4.77  0.00  1.0  5  4.0
class*      11 234  4.59  1.99  5.0  4.64  2.97  1.0  7  6.0

  skew kurtosis  se
manufacturer* 0.21  -1.63 0.34
model*        0.11  -1.23 0.73
displ         0.44  -0.91 0.08
year          0.00  -2.01 0.29
cyl           0.11  -1.46 0.11
trans*        0.29  -1.65 0.19
drv*          0.48  -0.76 0.04
cty           0.79  1.43 0.28
hwy           0.36  0.14 0.39
fl*          -2.25  5.76 0.05
class*       -0.14  -1.52 0.13
```

Output 5-8: Example of insight to the numeric variable, hwy, with the describe function.

linear plot would have been substantially influenced. Also note that we could have placed both line fits in the graph as a quick validation of a linear fit.



Output 5-17: Scatter plots subset on points and facets.

The right-most chart returns the left but is subset on the number of cylinders—a discrete variable. Now we can see the influences on the smooth fit of the first graph. Note the bottom right facet clearly reveals the source of the upward turn in the smoothed line.

Also, note the appearance of a strange subset that we would not have easily spotted in a scatter plot of many points. Are there five-cylinder vehicles in the dataset? Or, are there records in the dataset that need to be cleansed?

Let's explore the three blocks of code behind Output 5-17 for new elements of code. They are as follows:

```
#Left panel
#Compare data as scatter and subsets
#With subset on points
scatSubPt<- ggplot(mpgktd, aes(x = displ, y = hwy)) +
  geom_point(aes(color = drv, shape = class)) +
  geom_smooth() +
  scale_shape_manual(values=seq(0,6)) +
  labs(x = "Displacement", y = "Mileage") +
  ggtitle("Subset on points-class & drv") +
  theme(legend.position = c(.5, .8),
        legend.box = "horizontal")
```

```

scatSubPt

#Right panel
#With subset on points and facets
scatSubPrFc<- ggplot(mpgKtd, aes(displ, hwy)) +
  geom_point(aes(color = drv, shape = class)) +
  geom_smooth(aes(displ, hwy)) +
  scale_shape_manual(values=seq(0,6)) +
  facet_wrap(~cyl) +
  labs(x = "Displacement", y = "Mileage") +
  ggtitle("wrap added to subset on points") +
  theme(legend.position = "none")
scatSubPrFc

##Alternate code to the chart
scatSubPrFcAlt<- scatSubPrFc +
  facet_wrap(~cyl)
scatSubPrFcAlt

#Side-by-side
#Plot scat and scatFac
scat1x2<- ggarrange(scatSubPt, scatSubPrFc, ncol = 2, nrow = 1)
scat1x2

```

In the first block of code, we can see the base plot as the `ggplot` component. Notice how the x and y variables are coded in the aes expression as `x =` and `y =`. Because all other arguments are explicitly named, we can implicitly code the x and y variables. This more typical practice will be seen in the next block of code and for the remainder of the examples.

The `point` geom causes the scatter plot. The mapping code, `aes(color = drv, shape = class)`, subsets the points on color for drive and shape for class. The `smooth` geom places a statistically fit line over the scatter plot to visualize the pattern of the correlation.

The default to the `smooth` geom is `loess`. For it we have options for the degree of smoothness. The argument `span =` allows a range of 0.0 to 1.0.

There are alternatives to the `loess` fit. They are `lm` for a linear fit, `gam` for greater than 1,000 observations and `r1m` for reducing the sensitivity of the fit to outliers. Enacting the choice is made with the `method =` argument.

Let's speak to the shape legend. We have based shape on class. If we ran the chart, we would get one for which only six of the seven classes have been given a shape.

To get over that, we must call for shapes with the `scale_shape_manual` geom. The argument `values=seq(0,6)` assigns a shape to each category. If we wanted to select other than the 7 shapes from the 24 available choices, we would replace the `seq()` function with

a `c` function coded to list our choices. The readers are left to find the choices on the internet; an easy task and good experience.

Next, notice the expression `theme(legend.position = c(.5, .8), legend.box = "horizontal")`. The `legend.position = c(.5, .8)` code sets the placement of the legend with respect to the x and y axes. For each axis there is a range of 0 to 1. For example, the combination of `c(1,1)` would position the legend at the upper right of the chart. Meanwhile, the argument `legend.box = "horizontal"` places the legends side-by-side in the figure rather than stacked by default.

The second block of code returns the right-most chart to Output 5-17. Its primary distinction is to break the left-most chart into facets.

However, there is one other difference in its code. The `legend.position = "none"` in the theme function. To create space, the legend has been removed.

The third block demonstrates an important trick for efficient coding and the lazy amongst of us. We could have created the same graphic that was returned by the second block of code. The style appends the `facet_wrap` function to the first graph and returns the second.

We are seeing the fourth block of code for the first time. It causes the charts of the first and second blocks of code to be returned side-by-side as a single output.

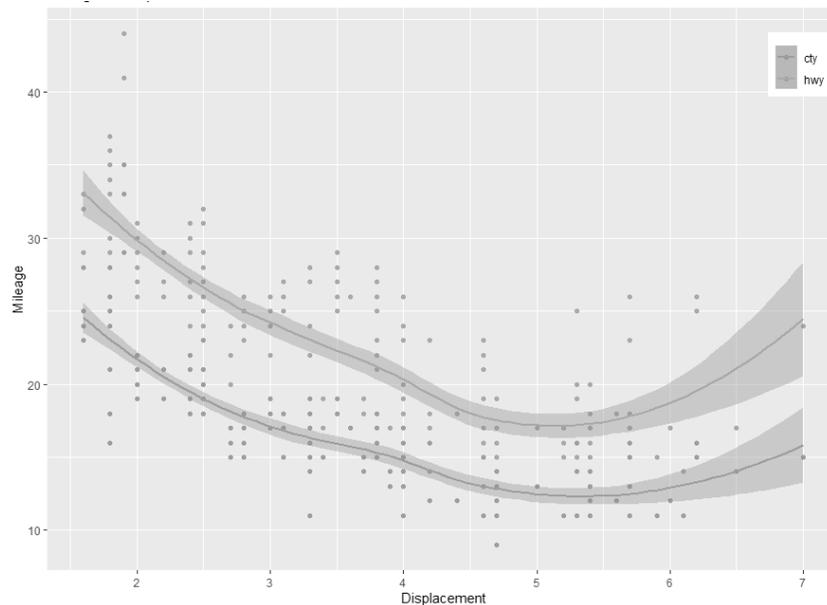
Notice that we have assigned the output to an object; `scat1x2`. We are using the `ggarrange` function of the `ggpubr` package. The charts are designated by their assigned name as objects. Thence, we have specified a single row and two columns.

The `ggarrange` function allows any number of charts by virtue of specifying the charts and number of rows and columns. They will be returned in the order called for by the `ggarrange` function.

We could easily create a dashboard with the function. In one sweep, we could run code to load the updated input tables to the graphs, the code to each graph and finally the `ggarrange` function to generate the dashboard.

Better yet, we can create a function to include all code such that we only need to call the function and all else updates and appears on our monitors for inspection. Building such a function will not be demonstrated here.

Another powerful way to subset is to place multiple base plots in a single graph. We can do this because they have axes in common. Output 5-18 shows two pairs of scatter plots and smooth charts as a seemingly single chart.



Output 5-18: Two scatter graphs presented as a single graph.

Let's look at the code to the returned the graph. Some important techniques are hidden below the surface. The code is as follows:

```
#Multiple base plots
scat2Chts<- ggplot(mpgktd, aes(displ, hwy)) +
  geom_point(aes(color = "hwy")) +
  geom_smooth(aes(color = "hwy"), se=TRUE ) +
  geom_point(aes(displ, cty, color = "cty")) +
  geom_smooth(aes(displ, cty, color = "cty"),
    se=TRUE) +
  labs(x = "Displacement", y = "Mileage") +
  ggtitle("Mileage vs Displacement") +
  theme(legend.position = c(.95, .9),
    legend.title = element_blank())
scat2Chts
```

The `ggplot` function sets up the base graph in association with the subsequent pair of point and smooth geoms. The second graph to display city mileage is returned by the second pair of geoms.

Notice in the second pair that the x-y variables do not match the pair in the `ggplot` function. The x variable is the same, but the y variables has become `cty`. This is the same for the smooth geom. The x-y of the second chart overrides the x-y to the `ggplot` function that supports the first charted plot.

```
#Lines per group with overall linear fit line
grpLines<- ggplot(Oxboys, aes(age, height)) +
  geom_line(aes(group = Subject,
    color = Subject), size = 1) +
  geom_smooth(method = "lm", size = 3,
    se = FALSE, color = "black",
    linetype = "dashed")
grpLines
```

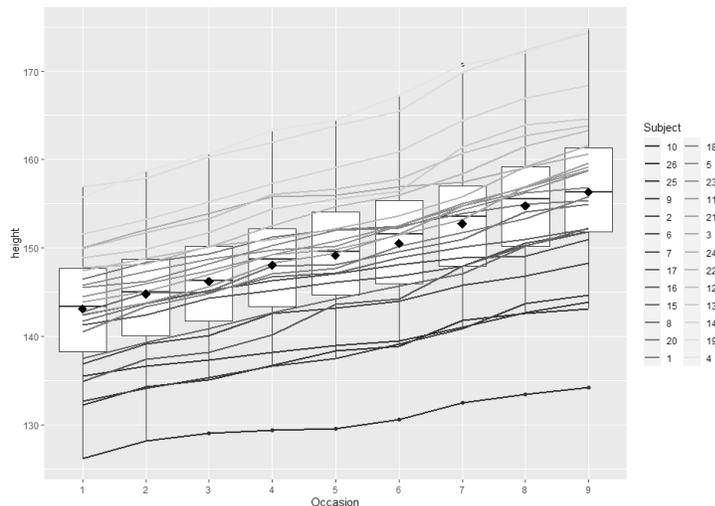
First, the code calls for a `line` geometric for each group and subsets on `color` as the legend for the subject boy. Second, the code calls a `smooth` geometric for the data as a whole.

Notice that the group code appears only in the `line` geom but not the `smooth` geom. It's a good example of layers engaging the same variables but presenting them with different purposes.

Let's show another play on the same idea. Our longitudinal variable will be "occasion" rather than age. In maintenance and reliability, the occasion could be month.

This time we want to inspect height with respect to the subject group. In maintenance and reliability this could be cost center. It could also be maintenance type, etc.

We can get a KPI-type perspective by generating a boxplot and average at each occasion. And of course, we can inspect all sorts of issues through the returned Output 5-31.



Output 5-31: Centrality and spread of height for each subject at each occasion and individual trends.

Below is the code to the output. In it, it is only notable that there are no new surprises. We have arrived.

```
#Plot by group and boys and mean at occasion
grpBox<- ggplot(Oxboys, aes(Occasion, height)) +
  geom_boxplot() +
  geom_line(aes(group = Subject,
    color = Subject), size = 1) +
  geom_point(stat="summary", fun="mean",
    color = "black", size = 4, shape = 18)
grpBox
```

There is another perspective of time dated variables called times series analysis. It will be fully explained and demonstrated in Chapter 10. It was briefly explained in Section 2.3.3 and shown in Figure 2-16.

The function `ts` converts our data into a time series object. With a time series object, we can separate seasonal and longer cycles from the trend of the data using other functions. Once cycle is removed, we can assess if the trend in our data is deterministic, random walk or random.

We would also test for autocorrelation between data points. That is when a variable in one period is somewhat influenced by outcomes in one or more previous periods.

In the world of maintenance and reliability, we love to speak of lead-lag indicators but offer no methodology other than assumptions. Time series analytics allow us to inspect our data for such relationships for highly correlated patterns between both variables at one or more reporting intervals. The relationship is called cross correlation in contrast to auto correlation to a single variable. It will be explained in later chapter about sustaining effective availability.

5.3. Save and Disseminate

The output to an R session is saved in the form of its R script. In turn, we call the saved script into an R session and run it.

It is likely that we will want to disseminate the tables and visual perspectives that we have formulated for our needs or in service of others. If so, we can distribute the R script.

With the code, recipients have the liberty to modify and refine the outputs to inspect their own data. Or they may use the output as templates to explore for perspective to other issues and KPIs.

This assumes the recipient has a threshold level of skills in R. Teaching to the threshold was the purpose of Chapter 4. However, we may want to send the know-thy-data deliverables to others without the threshold skills in R.

The simplest approach is to use the snipping tool to prepare a Power Point or Word deliverable. The alternative is to export the charts to a pdf file and, in turn, distribute the file. This can be done with the following code:

```
#Save to a pdf file
#Turn dev on, export to pdf file
pdf("<path>\\Dissim.pdf", paper="USr")
print(list(serUE, serTimeUE, pthPlt))
#Turn device off
dev.off()
```

The function `pdf()` is called an output device to send the output to a pdf file. As said in R, the function turns the device on. Within the function, we code the path to where the output is to land, the name of the created file and the paper type. We are using the US dimensions and have rotated the path to be landscape. The `print()` function and its argument, `list()`, specify the graphs by their name as an object. Finally, the code `dev.off()` closes the `pdf()` device.

It is important to stress the importance of closing the device; run the code line. If we do not, when we subsequently call for an individual graphic, as we have all along, the R graphic window will remain empty.

Bibliography

Wickham, Hadley. *gplot2, Elegant Graphics for Data Analysis*. Second Edition. Springer 2016.

Chapter 6

Unearth and Rectify Bad Data

From Chapter 3, we know how to build super tables. In turn, from Chapter 5 we know how to “know thy data” aided by layered charting. Now we are ready to cleanse our data. We are ready to unearth and rectify bad data.

Cleansing our data entails a sequence of assessments. Does our data have integrity; can we hang our hats on it? If not, which variables of the super table do we not trust? And which of them do we need to rectify? Finally, for the data we need to rectify, what is to be our strategy?

6.1. Bad Data and Rectification

Let’s begin by distinguishing between the types of “bad” data. Some are spotted in the know-thy-data stage that was explained and demonstrated in Chapter 5. Some are obvious upon perusal. Others are obvious by common sense. Others must be teased out with statistical methods and machine learning, artificial intelligence (ML-AI) models.

Of course, it is obvious that missing data is bad data. If the record containing one or more variables with empty cells is important to our planned insight deliverable, it contains actionable bad data. Section 5.1.2 explained and demonstrated how to generate a table of all records with missing data along with record- variable-specific counts for missing data.

Other bad data is apparent by common sense. For example, a large percentage of work orders have no recorded hours. Or close to 100 percent of the failures recorded to pump assemblies were attributed to pump rather than some reasonable number for other primary subassemblies. And of course, misformatted and misspelled classifications are bad data.

Then there are bad data that we cannot spot through mere inspection. We only know that they might be present. How do we know if the hours recorded to some work orders are bad data? How do we know if there are inappropriate classifications lurking amongst our records? Finding them is the biggest of all challenges and without an easy solution.

Sometimes, we can sidestep the challenge. This is because rectification is not a preordained action in response to every incident of bad data. Instead, we must decide whether to pursue rectification. Is it worthwhile?

For example, we may seek insight to the centrality and spread of a variable. Eliminating records with missing data may not significantly affect the insight as long as there is still statistical mass and representative spread.

Alternatively, we may seek insight with respect to observations. Without rectification, our insight can be distorted with an incomplete picture.

Rectification is to either eliminate bad data or impute estimates to numeric data and corrected classifications to categorical data. A deciding issue for eliminating records with bad data are the ramifications from working only with the number of remaining good records.

Eliminating records for bad variables may remove so many records from the dataset that we no longer have statistical or observational mass for the insight we seek. The collective permutations across variables may constitute a large share of the total records. Section 5.1.7 presented the analytics with which to make the judgement to rectify or abandon.

Even when elimination does not affect an insight deliverable, we should still leave all records with missing data in the super table. Instead, we cause the elimination to happen behind the curtains of the specific insight deliverable.

If eliminate or ignore are not appropriate strategies, we must travel a more difficult road. We must use ML-AI models to impute the best prediction of what the bad data should be.

Rectification strategies are subdivided with respect to two types of data; numeric and categorical. The types of bad are as shown in Figure 6-1.

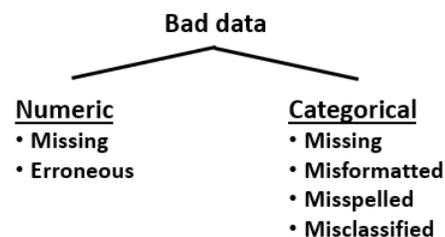


Figure 6-1: Types of bad data by type of variable.

Other than missing as bad, numeric data has lurking within it “potentially” bad data until proven as bad. We are looking for data that does not seem right or somehow does not fit.

However, there is a question to ask. Is each such observation erroneous or is it interesting? For example, the decimal was slipped or “7” was misread and entered as a “2.” Alternatively, we may find upon audit that there is not an error but there is an interesting discovery.

If the data point is interesting, we should not delete it. Otherwise, we would make the same decisions as we made for rectifying missing data. Erroneous data is equivalent to missing data.

In the cases of missing and erroneous numeric data, we may choose to impute an analytically derived estimate. As will be explained in sections to come, we would use ML-AI models to make the estimate. Chapter 7 will explain and demonstrate the most practical model for numeric data.

Other than missing, categorical data entails a different kind of bad. There are three types in addition to missing: misformatted and misspelled, and misclassified.

An example of the first type of bad data is a classification of all caps that may contain lower-case characters or be misspelled. It may present both flaws.

Alternatively, and more insidious, the classification may be incorrect even though correctly formatted and spelled. Fortunately, modern operational software is increasingly configured to prevent omission, misformatting and misspelling. Unfortunately, it is not always possible to prevent incorrect choices at the time classifications are assigned to a record.

The cleansing process is staged for categorical variables. First, we must rectify the misformatted and misspelled classifications. The method, done with translation tables, was presented in section 3.3.2 of the chapter to build super tables.

However, there may still be misclassified records. Section 6.3 will introduce the two most practical ML-AI models to seek them out. The full explanations and demonstrations of the models will be the subject of Chapter 14 to explain how to recover classifications such as failure modes and codes.

6.2. Strategies for Numeric Data

The elephant in the room is that there may be bad data in our super tables. The only obvious bad data are the missing cases revealed when we inspect our super table. At the level of subtables and single variables, even missing data can drop from sight. The process to locate them in our super table was explained and demonstrated in Chapters 3 and 5.

Spotting bad numeric data is difficult as compared to obviously misformatted and misspelled categorical data. Even then, incorrectly classified categorical data can easily slip past us; especially past a data scientist that is not a subject matter expert (SME) in the subject operations. In other words, part of the solution to bad data is that at least a critical mass of SMEs in our maintenance and reliability operations know their data science.

We will approach the search and rectification along two dimensions; numeric and categorical variables. This section will speak to numeric; the next categorical.

6.2.1. Spotting Outliers and Influencers

For numeric data, we are essentially searching for outliers, and leveragers and influencers. When we spot them, our attention shifts to determining if they are bad data or interesting. Depending upon our conclusions we must decide what our rectification strategy

matter experts in maintenance and reliability to select the predictor variables that give us the best fit compared to the null model.

Once we arrive at that best fit, we must go through vetting steps to evaluate the accuracy of the model. The build, fit and evaluation sequence is the subject of Chapter 7. Finally, the variables of the bad case are fed to the regression model to estimate a score to substitute in for the bad one.

6.3. Strategies for Categorical Data

The levels to categorical variables are sometimes called classifications. Maintenance and reliability data have many levels or classifications. Examples are the codes we select for variables such as maintenance type, craft type, priority, asset type, failure and more.

Categorical variables may be more vulnerable to being bad data than are numeric variables. They can be misclassified. Correct classifications may be corrupt.

Our CMMS and other systems may not enforce coding as a condition for an order to pass each step of the maintenance and reliability processes. We are additionally vulnerable if the paths we can follow through our maintenance systems are not gated by the code we select along the way.

Correct classification can be corrupted when entries are allowed to be free style rather than selected from menus. In these cases, our ability to accurately subset records can be lost by misspelling and misformatting.

6.3.1. Finding and Rectifying Misclassifications

As already explained and demonstrated in section 3.3.2, we can easily find and rectify misspelled and misformatted classifications with translation tables. In contrast, it is much more difficult to determine if classifications that are correctly formatted and spelled are also correct. And when not, we must determine what they should be.

One strategy is to employ the method that was explained and demonstrated in section 3.5. It was used to build in a super table a classification variable that was not normal to the CMMS. The example classified craft lead to orders based upon discerning the craft type with the greatest hours to each order. The methodology can also be employed to spot suspicious classifications in parallel with existing variables. It can also be used to help determine what they should be.

Of course, the best method is for an SME to inspect each record and accept or rectify it accordingly. However, the possibly laborious nature of the approach means that much beyond working with a few dozen records may be prohibitive.

There is an alternative. First, gather by ML-AI or inspection a set of records of good data with a significant number of records to each classification. Second, train a model on the good set. Third, input an additional block of data to the model and sanity-check the returned predicted classifications for accuracy; actual versus predicted.

The naïve Bayes method allows us to get at the gold. Together, both methods offer a two-fisted strategy to recovering bad classifications.

The idea of a model built on naïve Bayes is to determine the probability of a classification (e.g., failure mode or code) in union with the probability of a word appearing in the classification out of all classified cases in which the word occurred. Rather than a single word, the probability is generated upon the large body of words found collectively in free-text variables.

Naïve Bayes is the most common of probability-based methods. That is especially so for classification with free text. So much so that it has become the de facto standard for working with free text.

6.4. Subsets and Multilevel Models

There is an elephant in the room. We cannot take for granted that the cases (AKA, records) to our regression models are independent. For this reason, the book gives preference to the regressions over the trees as another possibility.

What does that mean? It means that the cases might only be independent within subsets. Otherwise, they are influenced by at least one subset to the records. If we disregard the possibility, we can bring misinformation into our envisioned insight deliverable. With respect to rectifying data, the estimated numeric or revised classification will not be as accurate as we think, if at all.

The concept is shown graphically in Figure 6-3. The right graph reflects linear regression and the left reflects logistic regression. To keep it visualizable, both reflect an outcome and only one predictor variable.

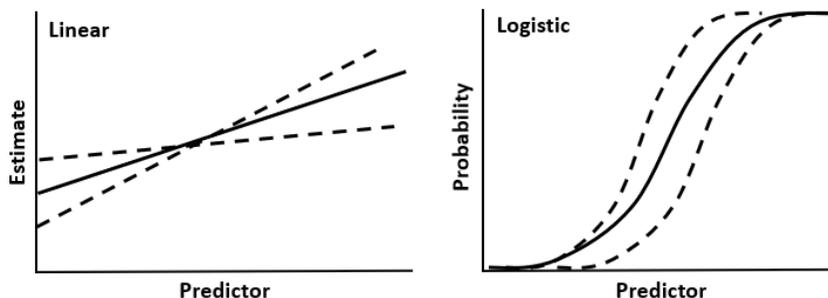


Figure 6-3: Cases requiring subsetting of models.

The solid line to both graphs represents the model fit to the entire body of records. As we subset models for categorical variables to the dataset, we may find that the models may be significantly different as depicted by the dashed line plots.

As an example, let's fall back to the miles per gallon data (mpg_ktd) that was used in Chapter 5 and the linear regression case to explain and demonstrate how to explore for

interpretation would be based on our sensitivities as a practitioner in reliability and maintenance operations.

We would accordingly subset our models on our sensitivities. In other words, we would ultimately build a collection of regular regression models rather than a single MLM. With them, we explore our data through the eyes of SMEs in the domain the data represents.

6.5. The Models in Other Purposes

Because the subject of the chapter has been to find and rectify bad data, four types of models have been singled out as the most practical of models to determine bad data and impute good. However, imputing is far from the mainstream purpose of the introduced regression.

Chapters 7 and 14, respectively, will explain and demonstrate linear and logistic regression models. It will be apparent that regression models allow us to explore the most important of all questions. Which asset, process and KPI variables are most strongly related to an outcome or a classifications of interest? The inverse is often also an important insight.

The reality is that we will more frequently use regressions models to explore the strength of relationship of predictors to outcomes than to predict outcomes. We are also often interested in the comparative strength of relationships to an outcome. This is because the most basic of all insight to any operation is what matters most to its success; especially if what we find that matters most is controllable.

For the tree models as a possibility for imputation, a recognition may only jump out at us upon reading the Lantz book. One was just mentioned. Search for which variables matter most. Trees automatically searches for them, whereas regression places the burden on the shoulders of the modeler.

Trees can also be a powerful tool in the work to audit and improve processes. The reason is that trees reveal operational rules that lie along the branches from decision nodes until reaching each leaf. Furthermore, as compared to human and social behavior, we would not expect much departure from the set rules. Consequently, it should get our attention when there are departures from the mandated rules.

It follows that we can use tree models to audit processes. With the rules as a learned model, we can seek those that are occurring but are not as mandated or expected. Where there is smoke, there may be fire.

In other words, a tree model reveals how the process is actually being worked. When there is a departure from policy, we can ask ourselves two question. Is there non-compliance that must cease and desist? Or have we discovered a refinement that should be made policy to the process?

Finally, the rules of the tree can be used as input to other models, such as a regression. This may be especially helpful to recover missing and misclassifications. We would add

the rules as a record variable in our super table. Thence, we would pull them into a logistic regression and, thus, improve its ability to estimate probabilities to the respective classifications.

Bibliography

Field, Andy; Miles, Jeremy; Field, Zoe. *Discovering Statistics Using R*. Sage Publications 2012.

Finch, Holmes; Bolin, Jocelyn; Kelley, Ken. *Multilevel Modeling Using R*. CRC Press 2014.

Lantz, Brett. *Machine Learning with R*. Second Edition. Packt Publishing 2015.

Chapter 7

Relate Operational Variables to Outcomes

At the mention of linear regression most of us think of predicting outcomes in engineering design and economics. In the previous chapter, we recognized linear regression as a means to estimate a replacement to bad data; assuming omission is not an acceptable strategy.

However, the big point in enterprise operations is how to predict an outcome. Which variables from our management systems, and metrics upon variables, are meaningful predictors to an outcome? Do any of the variables and metrics need to be transformed in order to sharpen predictability? Are there subdivisions within the variables that must be recognized? Does the model tell the same story universally or only for our circumstances?

In other words, if we can predict, it does not automatically follow that we will predict. More often than not, we instead use our ability to predict as a platform to discover how to plan, organize and control our operations for optimal performance.

I remember the golf pro who tried to fix my vicious slice. Instead of correcting my slice, she tried to teach me how to intentionally make a hook. Her logic was that if I can learn to cause a hook, I would learn to avoid both slices and hooks. The variables in our regression are the element of being able to consciously slice, hook and everything in between.

When we hear the term “regression,” we typically think of “linear” regression. However, there are three regressions within what is called the “general linear model” (GLM). They are distinctive by the type of outcome they predict upon a common structure of “linear coefficients.”

Linear regression of the GLM relates variables to continuous numeric outcomes. Logistic regression relates variables to classifications through probability of occurrence in the classification. Poisson regression relates variables to counts and counts per unit.

The body of predictor variables in the three regressions look the same. However, the linking functions between the body and the solution equations are different. In all three, we ultimately step through a trial-and-error process until we arrive at the ability to “predict.” And only then we will we see much more clearly what matters most to our operation and, in turn, our enterprise. Furthermore, senior management wants us to know and act upon what matters most.

This chapter will begin by introducing the common and distinguishing characteristics of the three regressions. It will then explain and demonstrate the issues, procedures and interpretation of the linear regression. Finally, it will apply the model to predict outcomes.

Most of the principles and methodology for linear regression apply in the other two with some nuances. Chapter 14 will extend upon this chapter to explain and demonstrate logistic regression. Poisson will not be addressed beyond this chapter. If interested, the reader can explore the regression type through the reference in the Bibliography to an example provided by the UCLA, Institute for Digital Research and Education.

7.1. The General Linear Model

The general linear model (GLM) is so called because the regression coefficients (β s) are to the order or power of one as can be seen in Figure 7-1. Furthermore, the systematic component or body of a linear model are similarly constructed for the three regressions. How we know to work the systematic body to build a model of maximal predictability is largely the same for the three.

Building linear models is not technically difficult. Insert variables, run and interpret the returned output. However, good models require thought. Because more input is only better to a point, it is necessary to carefully select which variables to include in the model. Then, because no model is perfect, it is necessary to determine how the model is flawed and how to counter the flaws. In fact, the work of decisions for inclusion and flaws may be the greatest of all values of modeling to our operations. There is learning in doing.

In this section we will summarize the components of models and the procedure to build them. The subsequent sections will explain by demonstration what is introduced.

7.1.1. Systematic Component

We can say that gaining the ability to predict starts with a simple question pertaining to the variables we can bring to the model. Which to leave in, which to leave out?

The action is to identify one or more predictor variables that are seemingly related to the outcome of interest. Is there a significant relationship? Is there a relationship, but somehow distorted? What is the direction of the relationship? Are any of the variables proxies to others that should, instead, be in the model?

The measure of how well a model predicts is how much of the variance to the mean of the null model (a model without predictors) is explained by the included variables. Too few, and too much of the variance is left unexplained. Too many and it is prohibitively difficult to interpret the model for insight.

The variables in a model are variously structured as shown in Figure 7-1. In the first systematic component, the three variables (X_1 , X_2 and X_3) are included as single terms (called main effect variable). In the second systematic component, one variable is included as a polynomial term (X_2) and one as a log (X_3). In the bottom systematic component, all three variables are included as main effects and as two- and three-way orders of interactions. Although not shown, we could include all main effects, transformations and interactions in a single model.

Once again, stay calm. We are not taking the watch apart to explain how to tell time. The simple takeaway of the figure is that the systematic component will ultimately be variables as a body of terms. The body can range from only one main effect to engaging some or all main effects, transformations and interactions that best tell the story. Additionally, the variables can be numeric and categorical.

With main effects

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$$

With polynomial and transformed variables

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_4 X_2^2 + \beta_3 \text{Log}(X_3)$$

↙ Transformed variable

↗ Variable as polynomial

With two-way and three-way interactions

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \underbrace{\beta_4 X_1 X_2 + \beta_5 X_1 X_3 + \beta_6 X_2 X_3}_{2\text{-way}} + \underbrace{\beta_7 X_1 X_2 X_3}_{3\text{-way}}$$

Figure 7-1: Model bodies—systematic component—of various configurations.

The in-or-out question is answered as the model is iteratively built and run until the analyst converges on the model that best depicts all relationships that will be significant to the sought insight. The choices and trials are made upon the expertise of the people who know the operation and parent enterprise. In other words, the best models are built by domain experts who know modeling rather than modeling experts who do not know the domain.

In real life, we will likely make choices for variables by any order or sequence. However, let's explain the process from top down. Iterations to evaluate the significance of each term begin with eliminating any insignificant interaction terms, from the highest order of interaction downward.

Thence, the significance of the main effects, if not included in a remaining significant interactive term, are evaluated for significance and pruning. For example, if a two-way interaction for X_1 and X_2 proved to be significant, the coefficients of both variables as main

The third case of Figure 7-2, Poisson regression, is concerned with counts and rates of occurrences. The latter ties count to some unit such as time, area and per capita. The linking function is the log of the identity function.

With the Poisson, we are still solving for a numeric outcome but as a count rather than continuous numeric. But in contrast, like the logistic regression, its outcome can only be zero or greater. And just as for logistic regression, the outcome will be multiplied by the exponential of one unit or level of change to the predictor variable. If we seek a rate, the difference is that the calculation is over unit.

7.2. Process of Machine Learning

Textbooks typically follow a sequence to explain the procedure of linear regression as machine learning and artificial intelligence (AI). First, we explore our data as explained in Chapter 5 to know our data. As part of the step, we inspect the strength of correlation between numeric variables.

Next, the objective is to select the variables from our dataset that best explain the total variances of the null model; mean of the outcome variable. The selection will be arrived at by trial and error and, thus, typically very insightful.

However, we do not stop with selecting variables. Instead, we determine which of the selected variables must or can be transformed in some way to arrive at a model that explains an even greater part of the total variance. We will also likely create and add variables to the dataset.

To this point our attention has been the predictor variables. The next stage is to seek the cases that are outliers to expectation or have an inappropriately large effect on expectation.

It is unacceptable to improve the model by eliminating them out of hand. Instead, we investigate them as either interesting cases or bad data. If interesting, they remain in the dataset and we explore for subsets and seek new variables that cause them to make sense. We rarely remove cases for any reason but bad data.

Finally, we check the model by charting and statistically testing the residuals to the prediction. Our interest is if they are normally distributed and if they disperse evenly along the fitted predictions.

If the final test of residuals shows a normal distribution and consistent spread, we can generalize the model. That means that although we built the model upon one plant's data, we can use the model to reasonably predict another plant's performance with its data. This is the essential definition of AI.

However, generalization is often a secondary or minor goal when exploring relationships. We are typically building to discover how to better plan, organize and control our operation rather than predict outcomes to other operations.

Furthermore, as will be apparent as the chapter unfolds, building models for individual operations and plants is not a prohibitive exercise necessitating that we generalize the evaluation of our own operation. We get our best insight from an operation-specific model even if we could have instead made inferences about our operation with a generalized model. However, a model structured for a common operation may be the same we would find to be the case of our equivalent operation.

To explain linear regression, for any operation, we will follow an alternative to the typical sequence. This is because it best suites the search for ways to advance operational performance. The chapter will unfold to select variables, seek necessary transformations, evaluate the residuals to predictions and finally seek the outliers and overly influencing cases. The last two stages are typically switched in order.

The reason for the switched sequence is that in operations the money is often found in the unusual. In the outliers and overly influencing cases, we often newly discover or confirm what matters most. We may also discover functional subsets in our operations. And when we find bad data, we likely discover to search for the root weaknesses that systemically allow them.

From there, the chapter will flow to techniques to cleansing data. For the sake of explanation by demonstration, we will pretend as if all outliers and influencers were found to be bad data. We will also pretend we would cleanse the cases rather than delete them. Generating imputes to the pretended bad cases will allow us to demonstrate the use of a machine learned model in its AI mode. We will subsequently introduce the method to estimate one or more new cases and their prediction interval, also AI.

The chapter's data is, of course, not data from a maintenance or reliability operation. It need not be because the teaching objective is to engage the types of variables of the dataset in a procedure, we would follow for asset management or any type of operation. The data is available from the worksheet, "insurance," in the Excel file "SkillsForCareerSecurity_Datasets.xlsx" that is available for download at the webpage provided in the preface. The R script to the chapter, `RelateOperVariablesToOutcome.R`, is also available at the same webpage with an `.txt` extension.

The dataset, using demographic data from the US Census Bureau, is a simulated dataset of hypothetical medical expenses for patients in the United States. It was created for the book, "Machine Learning with R," by Brett Lantz.

rather than using the final model to predict; engage it for AI. In other words, what we discover will likely lead to improving the asset management processes and roles in ways that matter most for advancing enterprise effectiveness and efficiency.

7.3. Extreme Cases Cleansed by AI

We know from the many previous graphics there are cases in our data that the model does not well fit. Their residuals to predictions are too large to accept without questioning.

Our first action would have been to seek additional variables to our dataset during the previous two stages. For example, we may have suspected that we need variables with respect to chronic diseases.

Now we turn our attention from gathering and evaluating variables to the model to evaluate the cases that reside outside of the mainstream. Accordingly, we will inspect the cases from two perspectives. First, which cases overly decide the model’s learned parameters, intercept and regression coefficients. Second, which cases are outliers to the mainstream but do not significantly change the parameters.

The contrast is that the model’s overall ability to predict is undermined by the first type of cases. The model poorly predicts the second type of cases but without significantly effecting the model.

The contrast suggests that we should begin by inspections for the first type. We will decide to eliminate or impute estimates to them. Then we will search out the cases of the second type. This order prevents the first type from causing or hiding the second.

Upon the two-staged scheme, Figure 7-4 charts the procedure we will follow through this section.

In the process, the Type I cases are removed from the dataset. To find the Type II cases, we run the model with the Type I cases removed and use it to identify the Type II cases. Both types are combined in a single table of dirty cases. A final clean dataset is created when the located Type II cases are removed from the dataset.

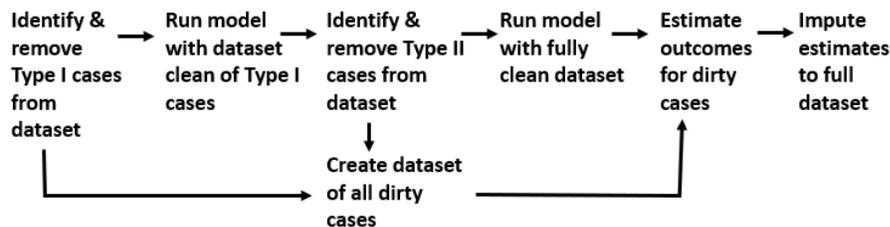


Figure 7-4: Procedure to find the cases that are out of the mainstream and cleanse bad cases.

To conduct AI for the purpose of imputing estimated good data, the model is run with the clean dataset. The dirty set is presented to the model to return an estimate for each dirty case. Finally, the clean and imputed data are combined as the working dataset to the model or for whatever purpose we have in mind for the cleansed dataset.

7.3.1. Flag Cases that Change Parameters

The cases we now seek were depicted by Figure 7-2. What they have in common is that they significantly change the parameters of the model, intercept and regression coefficients.

Our data, as presented in Output 7-17 and before, suggest that the long tail of Output 7-19 is mostly comprised of outlier and leveraging cases rather than influencing cases.

The data points are far north but within the horizontal range of the range of the x axis points. Upon inspection of scatter plots such as Output 7-17, we cannot spot points that are simultaneously extreme to the vertical axis and well outside the range of the points on the x axis.

As mentioned previously, we will first search for Type I cases. We will also distinguish them as leveraging and influencing cases in the dataset. We will call upon the hat test for the former and the Cooks distance test for the latter.

Both tests work the same way. They run the model with and without each case to compute their respective hat and Cooks statistics. We then filter for leveraging and influencing cases with the criterion we set for vis-a-vis.

Let's begin with the hat or leverage cases. The first block of code below will test each case for leverage and send the score to the `insure.3` dataset as the variable `HatLever`. Notice that the `hatvalues` function makes the computation for each case with respect to `model.6`.

```
#Create variables to leverages cases
insure.3$HatLever<- hatvalues(model.6)

##Compute Hat test (leverage) criterion
k<- 6
g<- 3
h<- g*(k + 1)/length(insure.3$HatLever)

#Form table of leverage cases
hatTbl<- insure.3[insure.3$HatLever > h, ]
length(hatTbl$HatLever)
```

The second block of code computes the criterion for deciding which cases are leveraging the model. The term “k” is the number variables in the subject model; six in our model. The term “g” is a choice for multiples of the $(k + 1)/n$ calculation. We will use $g = 3$.

The third block of the code filters the `insure.3` data for cases greater than the criterion computed by the second block. The cases revealed by the criterion are placed in a new dataset named `hatTbl`. The `length` function is used to tell us the number of cases. Although the output is not shown, there are 13.

Next to look for cases that are influencers. Our inspection of the charts to the chapter would suggest that none exist. However, we will test to be assured the inference is correct.

The code below will use the Cooks distance test to seek out influencers. Using the `cooks.distance` function with the model as its argument, the first block of code measures each case and places the measure as the variable, `CookInflu`, in the `insure.3` dataset.

```
#variable to candidate influencers of expectations
insure.3$CookInflu<- cooks.distance(model.6)

##Compute Cooks distant test for influencers
cookTbl<- insure.3[insure.3$CookInflu > 1, ]
length(cookTbl$CookInflu)
```

We are using the criterion of greater than one to return cases that are influencing cases we would audit as candidate bad data. Finally, the count of cases is returned with the `length` function. The count is zero as we would expect from our visual inspection.

Leveraging and influencing cases are usually located away from the mainstream of fitted predictions. However, they may not appear as extremes in the same sense of outlier cases. Instead, the “holes in the cheese line up” such that the intercept and coefficients of the model are torqued to some significant degree even though they may not be blatant among outliers.

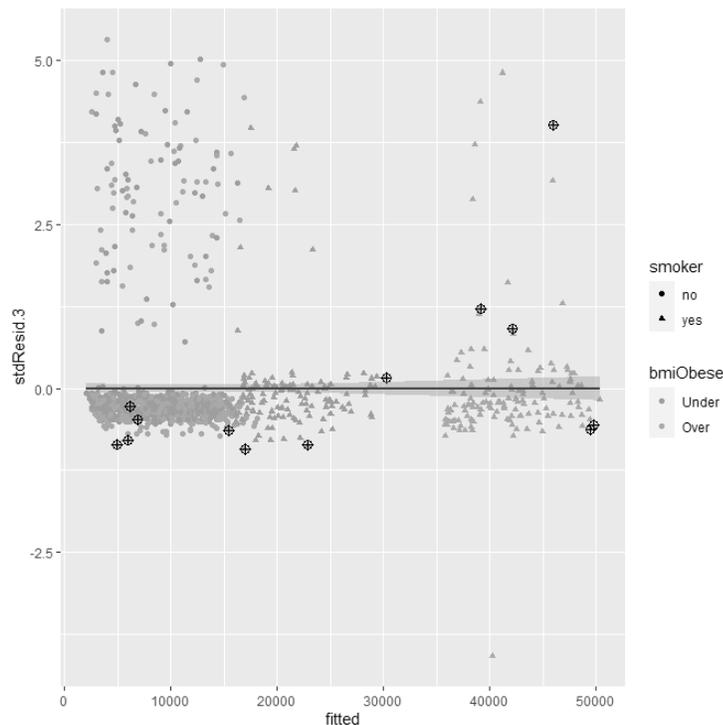
Therefore, it is good to get a visual sighting of the cases. We will do so with a residuals-to-predictions plot. It is done with the `ggplot2` ability for layered charting. The code below returns the chart shown in Output 7-21.

```
#View hatTbl and cookTbl points among all points.
HatCookPts<- ggplot(insure.3, aes(fitted,
  stdResid.3, )) +
  geom_point(aes(color = bmiObese, shape = smoker)) +
  geom_point(data = hatTbl, aes(fitted,
  stdResid.3, ),
  shape = 10, size = 3) +
```

210 | Chapter 7

```
geom_point(data = cookTbl, aes(fitted,
stdResid.3, ),
shape = 5, size = 3) +
geom_smooth(method = "lm", color = "Blue")
HatCookPts
```

The code is an opportunity to recall and demonstrate an important power of ggplot2. We are not using a single dataset as we must in Excel charts and most charting software. Three are engaged in the graph. The first is the dataset of all points; `insure.3`. The remaining two pull the `hatTbl` and `cookTbl` datasets into the chart. This possible because axis of all is the same.



Output 7-21: Leveraging and influencing cases shown among all points.

We can see the causal code if we inspect the `geom_point` geometrics to the chart. The first will run with the dataset, `insure.3`, upon which the graph is based. In the next two occasions of the geometric, we see the argument for dataset. Each respectively calls for the two datasets we created for leveraging and influencing cases.

We previously found that there were no influencing cases in our dataset. However, for completeness the code for influencing cases has been included. In this way, the reader will have a go-by for the situations in which there are influencers in their own datasets.

In the output, notice that the leveraging cases are shown with a different shape and size. Not apparent is that if there had been influencing cases, there would have been a unique shape and size to distinguish them from leveraging cases.

Now, we want to remove the leveraging and influencing cases from the dataset. Of course, this assumes they have been inspected and found to be misinformation. Otherwise, we would be very thoughtful about removing them from the `hatTbl` and `cookTbl` datasets as bad data.

The code below removes the cases from the `insure.3` dataset and creates a new dataset named `clean.1`.

```
#Create Dataset with levers and influencers removed
clean.1<- insure.3[insure.3$HatLever <= h &
  insure.3$CookInflu <= 1, ]
str(clean.1)

#Check total numbers
length(insure.3$expenses)
length(clean.1$expenses)
length(hatTbl$HatLever) + length(cookTbl$CookInflu)
```

Notice that the dataset `clean.1` is created by reversing the respective criteria to the leveraging and influencing cases. The `str` function allows the reader to inspect what was wrought.

The purpose of the second block of code is to check the balance between the original, cleansed and the sum of the `hatTbl` and `cookTbl` datasets. The latter two should sum to match the first. It is left to the reader to check the balance with the code.

7.3.2. Flag Outliers Cases to the Mainstream

We wanted to remove the cases that distort the model's parameters before looking for outliers to the mainstream of predictions. Recall that outlier cases do not tamper with the parameters. Instead, we would depict them as cases the model does not reasonably predict.

It follows that, if the cases that overly effect parameters were not previously removed, the outliers may be hidden in a model of distorted parameters. The code below will identify

212 | Chapter 7

the outliers and remove them from the dataset that was previously cleansed of leveraging and influencing cases.

```
#Run model with data absent leverage and influence
model.clean.1<- lm(expenses ~ age + age2 + bmi + children +
  bmiObese*smoker, data = clean.1)
summary(model.clean.1)

clean.1$stdResid.4<- rstandard(model.clean.1)
str(clean.1)

#Criteria for candidate outliers to expectations
#Use 2.58 for 1 percent cases
UpLmt<- 2.58
LWLmt<- -UpLmt

residTbl<- clean.1[clean.1$stdResid.4 < LWLmt |
  clean.1$stdResid.4 > UpLmt , ]
length(residTbl$stdResid.4)
```

To test for outliers against a model that has not been distorted by other leveraging and influencing cases, the first block of code runs the model of variables determined earlier (Section 7.2) but with the data cleansed of all distortive cases. The second block extracts the standardized residuals from the model and places them in the dataset `clean.1`.

Next, we identify in the dataset the outliers per a criterion reflective of our preferred confidence limit. As coded in the third block, we will filter for outliers outside of the 99 percent range to a normal distribution. The two-tailed equivalent of a standardized Z score is 2.58. The code is designed to allow us to set the limits at one time for all occasions in code that subsequently draw upon the score.

The fourth block filters for cases that fall above and below the 99 percent limits. With the results, it creates the dataset `residTbl`. With it in hand, the analyst can audit the cases for being interesting or bad data. The `length` function returns the number of cases; 77 if we ran the code. That is compare to 14 cases we would expect upon a 99 percent confidence.

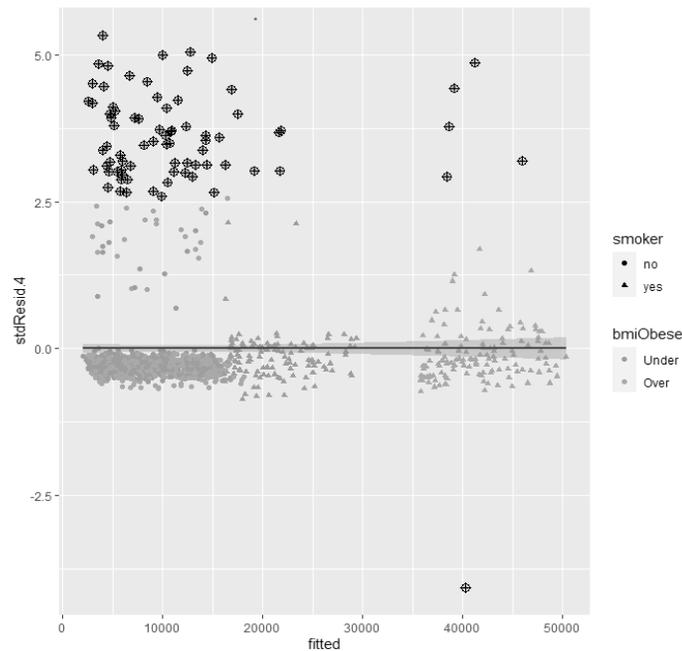
Again, it is a good idea to view the cases relative to a residuals-to-predictions plot. This is even though we could have merely inspected the plot upon `clean.1` for the cases above and below 2.58 on the standardized residuals axis.

The code below is like the previous code to overlay leveraging and influencing cases on the chart of all points. Output 7-22 returns the results.

```
#View outliers in clean.1 data
outlierPts<- ggplot(clean.1, aes(fitted,
```

```
stdResid.4, )) +
  geom_point(aes(color = bmiObese, shape = smoker)) +
  geom_point(data = residTbl, aes(fitted,
    stdResid.4, ),
    , shape = 10, size = 3) +
  geom_smooth(method = "lm", color = "Blue")
outlierPts
```

We can see that all but one outlier is to the north side of the plot. We can see that the greatest number of cases appear among nonsmokers. In previous sections we used grid functionality to subset the outliers with the `facet_grid` function. It is left to the readers to make their own subsets.



Output 7-22: Outliers cases to the mainstream with 99 percent confidence.

7.3.3. Create Datasets for Imputation

The previous section demonstrated how to seek and pull out the three types of cases to be investigated as either interesting or bad data. We can expect that we would form a list of cases to investigate and have audited them before continuing.

In a world of operations, the list may be our biggest reward. This is because operations excel upon doing the right thing and doing them right. The odd balls may reveal

that some right things are not being done, that some wrong things are being done right, and that some right things are not being done right.

We inspect the cases for two possibilities. The first is that the cases are interesting rather than bad data. Interesting cases are either left in the model's dataset or we discover that we need to subset the model. It is also possible that a case is beyond the realm of recurrence or reality.

The cases may also lead us to recognize that we need to pull additional variables into the model. Those included may still do not explain as much of the variance to the null model as we sense that they should. These may be from our systems or teased out of the dataset.

The second possibility is to find that there is misinformation in the variables to the cases. The correction may be obvious such as a slipped decimal, incorrect selection of levels, etc. We may revise the variables after audit or use our good data to estimate and replace the misinformation. As already mentioned, the latter entails AI. Sound glorious but is actually pedestrian as will be apparent in this section.

This section will assume that the full list of cases will be cleansed and imputed to our data. This is not realistic. However, we are not insurance experts, nor do we have the systems to investigate each case as candidate bad data.

Upon the "pretended" all bad, we will begin with building dirty and clean data tables. With the dirty table, we will demonstrate the action of AI to estimate replacements to each case of bad data.

The demonstration is the procedure to cleanse data by AI. However, the typical purpose of AI is to predict outcomes with the model running on good data.

The code below will create the datasets we will need. The final products are two datasets: one clean and one dirty. The clean will be used in the next section to estimate expenses for the dirty dataset.

```
#Create Dataset with dirty data only
dirty.1<- clean.1[clean.1$stdResid.4 < LwLmt |
  clean.1$stdResid.4 > UpLmt |
  insure.3$HatLever > h |
  insure.3$CookInflu > 1, ]
str(dirty.1)

#Remove duplicates from dirty cases - if any
dirty.2<- distinct(dirty.1)
str(dirty.2)
```

212 | Chapter 7

the outliers and remove them from the dataset that was previously cleansed of leveraging and influencing cases.

```
#Run model with data absent leverage and influence
model.clean.1<- lm(expenses ~ age + age2 + bmi + children +
  bmiObese*smoker, data = clean.1)
summary(model.clean.1)

clean.1$stdResid.4<- rstandard(model.clean.1)
str(clean.1)

#Criteria for candidate outliers to expectations
#Use 2.58 for 1 percent cases
UpLmt<- 2.58
LWLmt<- -UpLmt

residTbl<- clean.1[clean.1$stdResid.4 < LWLmt |
  clean.1$stdResid.4 > UpLmt , ]
length(residTbl$stdResid.4)
```

To test for outliers against a model that has not been distorted by other leveraging and influencing cases, the first block of code runs the model of variables determined earlier (Section 7.2) but with the data cleansed of all distortive cases. The second block extracts the standardized residuals from the model and places them in the dataset `clean.1`.

Next, we identify in the dataset the outliers per a criterion reflective of our preferred confidence limit. As coded in the third block, we will filter for outliers outside of the 99 percent range to a normal distribution. The two-tailed equivalent of a standardized Z score is 2.58. The code is designed to allow us to set the limits at one time for all occasions in code that subsequently draw upon the score.

The fourth block filters for cases that fall above and below the 99 percent limits. With the results, it creates the dataset `residTbl`. With it in hand, the analyst can audit the cases for being interesting or bad data. The `length` function returns the number of cases; 77 if we ran the code. That is compare to 14 cases we would expect upon a 99 percent confidence.

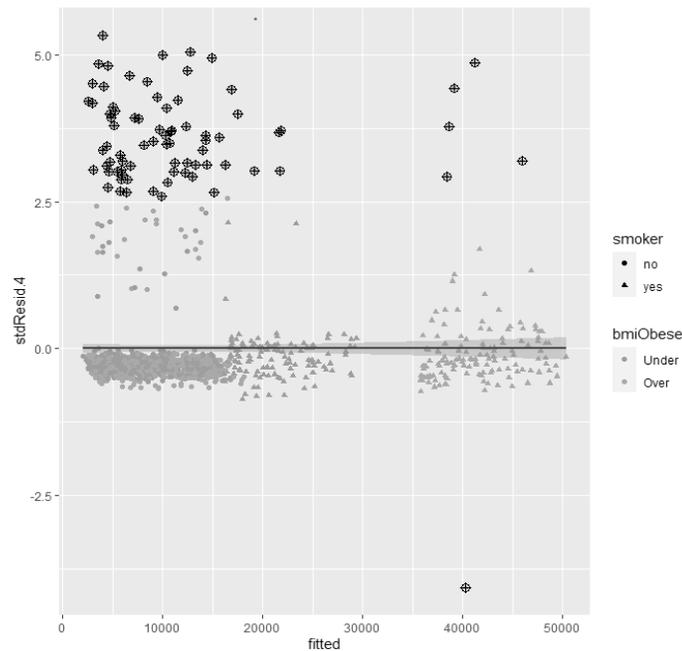
Again, it is a good idea to view the cases relative to a residuals-to-predictions plot. This is even though we could have merely inspected the plot upon `clean.1` for the cases above and below 2.58 on the standardized residuals axis.

The code below is like the previous code to overlay leveraging and influencing cases on the chart of all points. Output 7-22 returns the results.

```
#View outliers in clean.1 data
outlierPts<- ggplot(clean.1, aes(fitted,
```

```
stdResid.4, )) +
  geom_point(aes(color = bmiObese, shape = smoker)) +
  geom_point(data = residTbl, aes(fitted,
    stdResid.4, ),
    , shape = 10, size = 3) +
  geom_smooth(method = "lm", color = "Blue")
outlierPts
```

We can see that all but one outlier is to the north side of the plot. We can see that the greatest number of cases appear among nonsmokers. In previous sections we used grid functionality to subset the outliers with the `facet_grid` function. It is left to the readers to make their own subsets.



Output 7-22: Outliers cases to the mainstream with 99 percent confidence.

7.3.3. Create Datasets for Imputation

The previous section demonstrated how to seek and pull out the three types of cases to be investigated as either interesting or bad data. We can expect that we would form a list of cases to investigate and have audited them before continuing.

In a world of operations, the list may be our biggest reward. This is because operations excel upon doing the right thing and doing them right. The odd balls may reveal

as it is for the `clean.3` dataset. It then included variables seven through nine in the dataset. The result is a ten-column dataset assigned to `append.1`.

Just as for the `append.1` dataset, by omission, the second block of the code removed all test variables from the `clean.2` dataset. The result is assigned to `clean.3`.

The third block appends the dataset with imputed expenses to the clean dataset. We use the `rbind` function and assign the result to `clean.4`. If we ran the `str(clean.4)` code, we can confirm the variables and count of 1,338 cases.

We could use the full dataset, `clean.4`, to run our model. After doing so, it has been assigned to the model object `model.Impute`. Upon running the code, we would find that it explains almost 98 percent of the variance to the null model.

```
model.Impute<- lm(expenses ~ age + age2 + bmi +
  children + bmiObese*smoker, data = clean.4)
summary(model.Impute)
```

In real life, we would expect and accept as good AI something a bit better than the 85 percent explanation of total variance. That was the case before removing all of the three types of questionable cases.

Of course, this is unrealistic. We are only demonstrating the cleansing method. It is unlikely that every, if even many, of the cases for investigation would be found to be misinformation. However, we will use the model in the next subsection to demonstrate how to estimate (conduct AI) individual cases.

This is a good place for an important lesson to the hyperbole of AI. Models in the upper 80 percent are about as good as it gets. Above that, we must be suspicious that the model builder is tailoring the dataset rather than tailoring the model's ability to predict.

7.4. Individual Predictions and Intervals

As said earlier, we most typically strive to be able to predict rather than to actually predict. The previous section did predict to impute a good estimate to bad data.

There are other reasons to predict. For example, we may want to predict for an individual their expected medical expenses given their personal circumstances. It follows that if we had arbitrarily removed the oddballs from the dataset, our estimate may be misinformation to the individual.

This final section will provide a go-by for prediction of individual cases. The code below is designed to allow for one or more cases. The output is shown in Output 7-23.

```

#Inputs to the prediction
age<- c(25,45)
age2<- age^2
children<- c(2, 2)
bmi<- c(23, 38)
bmiObese<- c("Under","Over")
smoker<- c("no","no")
EstExpIndiv<- data.frame(age, age2, children, bmi, bmiObese, smoker)
EstExpIndiv

#Predict and add to data.frame EstExp
EstExpIndiv$expenses<- predict(model.Imput, EstExpIndiv, interval =
  "prediction", level = .95)
EstExpIndiv

```

The first block of the code allows us to enter the unique characteristics to the prediction. Code of the format `c(x, y)` allows for multiple entries. Otherwise, we would insert a single argument in the `c` functions. Thence, with the `data.frame` function, we create the dataset for estimate. We assign it to an object `EstExpIndiv`.

Once again, we see the `predict` function in action, but with two new arguments. Upon our prediction, we would want to know the upper and lower intervals to the estimate. Furthermore, we need to make the distinction between prediction and confidence interval.

The prediction interval is the range in which an individual estimate will fall some percent of time. In contrast, confidence interval shows the likely range of values associated with some statistical parameter of the data or model such as the population mean, coefficients, etc. Until now we have always spoken to the latter of the two.

Accordingly, we can spot two additional arguments in the `predict` function: `interval` and `level`. For the first, we specify that we want the prediction intervals, and for the second, we want a 95 percent probability of occurrence within the interval.

Once again, the vector of predicted expenses and intervals are placed in the data frame as the `expenses.fit`, `expenses.lwr` and `expenses.upr` variables. Output 7-23 is the returned output of the `predict` function that is assigned to the `EstExpIndiv` data frame.

```

> EstExpIndiv
  age age2 children bmi bmiObese smoker expenses.fit expenses.lwr expenses.upr
1  25  625         2  23   Under    no    3938.2432    509.3956    7367.0909
2  45 2025         2  38    Over    no    9075.5355    5647.2847   12503.7863

```

Output 7-23: Prediction of medical expenses and intervals to individual cases.

Bibliography

Field, Andy; Miles, Jeremy; Field, Zoe. *Discovering Statistics Using R*. Sage Publications 2012.

Lantz, Brett. *Machine Learning with R*. Second Edition. Packt Publishing 2015.

Ott, Lyman and Longnecker, Michael. *Introduction to Statistical Methods and Data Analysis*. Seventh Edition. Brooks/Cole, Cengage Learning 2016.

Institute for Digital Research and Education. UCLA. Poisson Regression Example.
<https://stats.idre.ucla.edu/r/dae/poisson-regression/>

Chapter 8

Achieve Entirety of Data

Operating systems and spreadsheet software emerged in the 1990s along with the reengineering craze around what the systems made possible. Reengineering has passed on but most operations, including maintenance and reliability, are now almost completely supported by systems. Where there are not systems for tasks along an operation, they are frequently supported with homegrown Excel spreadsheets.

Think about the ramifications. Between them, the technologies capture every type and piece of data that is natural to an operation. Consequently, every act of planning, organizing, execution and control leaves a data trail. In turn, for every aspect of achievable data-driven asset management, what is captured along the trail is the feedstock data.

To this place in the explanation of data-driven asset management, all has been on the presumption of entirety of data. Entirety is defined as all naturally generated data are layered, current to capture and accessible on demand. The fact is that few operations have fully achieved the standard of entirety.

Data in a layered structure, as the first defining characteristic of entirety, recognizes that some data follows trails that are outside any operating system. The cases are visible as Excel spreadsheets. This is compared to system standard reports.

Spreadsheets are frequently engaged by role holders in the conduct of the processes they are responsible for. It follows that a data-driven enterprise would want to capitalize on the data trail within them. Unfortunately, the data in the background of the information is often lost to us because it is not captured within the discipline of a system. Instead, data and report are fused into a single object.

Current to capture operational data, the second defining characteristic of entirety, recognizes that the data along the trail must be available timely to all data-driven activities. Otherwise, our activities will be limited to those still allowed by the time the system's data are made available through the system.

An example is craft hours distributed to work orders. Ideally, hours would be recorded directly into a CMMS or hours tracking system and accessible close to real time. Otherwise, it may be days before the hours become available data accessible to us as a system standard report. Our data is continually short of entirety.

The third defining characteristic of entirety is that operational data must be accessible upon demand. The data in the system may be timely. However, the system may have not been configured to allow us to extract some operational data as system standard reports.

An example is work status history from a CMMS. For some inexplicable reason many maintenance systems still do not allow the history of a group of orders to be extracted as a system standard report. For these plants, their data is not entire to its role holders, although entire to its systems. Consequently, the plants do not have the functional entirety to be fully data driven.

The bottom line is that any plant will have data following trails through the discipline of operating systems. The problem is that this describes only a part of the total data flowing through the operation. Accordingly, a plant cannot rest upon a satisfaction of what it has but must, instead, achieve entirety. This chapter will fully explain the introduced three characteristics of entirety and demonstrate each, in the context of practices for maintenance excellence, how full entirety is achieved.

The data to the chapter are available for download from the webpage provided by the preface. In the Excel file, SkillsForCareerSecurity_Datasets.xlsx, the datasets are as the sheets tblTimeSheet and tblStatusHistory.

8.1. Layered Structure

All system standard reports, dashboards and analytics sit on top of one thing. Data formatted as tables. The tasks along an operation's processes both contribute and extract the data captured and stored as tables in a relational database.

What about the tasks along the operation that are not subjected to the relational-database discipline of an operating system and are, instead, conducted with software such as Excel? Are they being conducted while observing the two-layered structure that is inherent to the principle of a system and its relational database? We need them to be.

Let's establish what is meant by a layered structure in contrast to a spreadsheet structure. Layered is the correct practice but spreadsheet is the most frequently observed. The important point is that layered structure is mandatory to having available to us the entirety of the data thrown off by an operation.

To begin the explanation, we must establish precisely what is a table in contrast to what is a spreadsheet. This has become clear from the previous chapters, but the contrast tells the story.

A table is defined by its format because the following rules hold:

- Each column is a variable of numeric, categorical or free-text information.
- Column headers are single level, no header spans multiple variables.

Under a two-layer regime, the spreadsheet of Figure 8-1 is never built. Instead, the spreadsheet is generated with Excel Pivot or an equivalent software. As a working second layer, Excel Pivot is the most ubiquitous of all offerings.

A layered structure is a gift that keeps giving. First, multiple perspectives to the same operational issue, can be readily created in a pivot at the speed of click and drag. Second, rather than the static spreadsheet of Figure 8-1, we can interactively slice-dice, drill down and roll up in search of insight. Third, any table can as an independent object can be reached into from whatever operational task and for whatever super table to which its data are contributory.

How do we get past the fused spreadsheets that prevent entirety of data? First is to locate them and identify the variables to them. Second, and we have an option, we can work a two layered structure from this day on. Alternatively, we can extract the variables some distance back in the past and begin a two-layered structure from the earliest extracted history to the present. The choice is decided by whether we can accept the accumulation of data without recovering it from the fused past.

8.2. Current to Capture

Entirety is lost when some data are only available once processed to some final state. By then, enterprise performance is lost because entirety is never timely to short cycle planning, organizing and controlling. Furthermore, this is a persistent rather than sporadic loss because longer-term effectiveness and efficiency are dependent upon very short-term effectiveness and efficiency.

The day's craft hours are the most noteworthy of possibilities. Therefore, let's frame the problem and treatment with hours as an example of "current at capture."

There are two situations. One is that hours are not record directly into a CMMS or a labor tracking system. The other is the reverse but still problematic. What is captured no later than shortly after the close of the recorded day goes into the database but does not return to us as a system standard report until fully processed days later.

An example of the first situation is represented in Figure 8-2. Notice that we have two timesheets that do not meet the standard of a layered structure. One timesheet records the hours for the firm's direct employees. The other is the timesheet for a contractor's crafts which are serving the same unit. If there had been more than a single contractor, there would be a timesheet for each.

Notice in the timesheets that there may be multiple crafts to an order task. We can also see in both the firm and contractor crafts may work a task as a single crew. However, the crewmembers to work are recorded in separate timesheets with respect to source.

Consequently, to prepare the day’s hours to respective payrolls, the supervisor must think four-dimensionally across multiple crafts, tasks, overhead events and organizations. It is also possible that multiple supervisors and leaders will be recording hours with respect to the sources of labor.

Given all of this, it is necessary for the firm to pull all details together in an accurate, error free unified picture of hours to jobs and individuals, and confirm that the hours are what has been granted by schedule and job plan for the shift’s workload. Because this is so difficult, we are being divided and conquered by our data.

Employee Daily Time Sheet												
Date:		8/17/2020										
Personnel Number	Name	Occupation	Work	OP	Shift 1		Days		Shift 3		Notes (reason)	REM
					Straight Time Hrs	Overtime Hours	Straight Time Hrs	Overtime Hours	Straight Time Hrs	Overtime Hours		
10	Jones	MA - L1						9				SICK
12	Smith	MA - L1	785717	10				2				
				30				7				
19	Roberts	MA - L1	785717	10				2				
				30				7				

Contractors Daily Time Sheet								Date:	08/17/20	
Vendor Name: Ace Maintenance										
Contractor's Employee Name	SAP Personnel Number	Work Order Number	WO OP	WO Sub Op	Offsite Area	S/O/D	Work Hours	Comments/Activity Work Description		
Stewart	101	785717	10			S	6.00			
		976254	40			S	2.00			
			60			S	1.00			
Stevens	103	976254	40			S	7.00			
			60			S	2.00			

Figure 8-2: Timesheets of multiple crafts from multiple sources serving a single unit and recorded in fused spreadsheet form.

What happened in this actual case is the data was being irrecoverably destroyed. The mental rigor and tedium of working four-dimensionally with the fused timesheets tempted shortcuts. The supervisors routinely assigned hours to the day’s larger tasks, leaving goose eggs for all else. On average, the day’s total hours were assigned to 40 percent of the day’s orders. Even once it was mandated that hours were to be accurately recorded to work, the data was still untimely for short-cycle control of the operation.

The solution is to capture the hours as a table such as Figure 8-3. Notice that the collective timesheets of Figure 8-2 entail 11 variables. Each record of hours would include all or most of the variables. In one case, a craft’s hours accrued to sick time, thus, order

and task are not populated. The table is available for download at the webpage provided by the preface. It is contained in the Excel file SkillsForCareerSecurity_Datasets.xlsx as the sheet tblTimeSheet.

As an Excel table, the records can be sorted and filtered by its variables for any purpose. The supervisor may wish to monitor hours on the fly. Once all hours have been recorded for the day, the supervisor may sort and filter the table as necessary to make a final confirmation of accuracy. To monitor total maintenance cost for her unit, she may also wish to inspect all hours regardless of which are the direct crafts and contractor crafts to her unit. Furthermore, pivots as the second layer allow the immediate conduct of more advanced audits against the day’s schedule and job plans (Chapter 12).

Date	Unit	Provider	PerNo	Name	Grade	Order	Task	Shift	HrType	Hours	Exception
8/17/2020	Crude	Firm	10	Jones	MA1			Day	S		9 Sick
8/17/2020	Crude	Firm	12	Smith	MA1	785717		10 Day	S		2
8/17/2020	Crude	Firm	12	Smith	MA1	785717		30 Day	S		7
8/17/2020	Crude	Firm	19	Roberts	MA1	785717		10 Day	S		2
8/17/2020	Crude	Firm	19	Roberts	MA1	785717		30 Day	S		7
8/17/2020	Crude	Ace	101	Stewart	MA1	785717		10 Day	S		6
8/17/2020	Crude	Ace	101	Stewart	MA1	976254		40 Day	S		2
8/17/2020	Crude	Ace	101	Stewart	MA1	976254		50 Day	S		1
8/17/2020	Crude	Ace	103	Stevens	ELEC1	976254		40 Day	S		7
8/17/2020	Crude	Ace	103	Stevens	ELEC1	976254		50 Day	S		2

Figure 8-3: All timesheets record as a layer to for the timesheet and any report requiring the hours.

Obviously, a two-layered structure allows the operation to expedite the payroll and distribution processes. This will greatly reduce the associated substantial clerical work in plants with large craft bodies engaged in the job-shop type work that maintenance is. However, most importantly is that the hours are immediately available to us at capture. Once the day’s hours are recorded, our operational dataset is entire with respect to expected hours.

Hopefully, but not always the case, at sometime soon the day’s data will be fully processed and made available as a standard report of the CMMS. However, a data-driven operation does not allow its data to languish until what we want to know from it becomes “never mind.” When not allowed to languish, the data is available for close to real-time purposes that are more valuable than waiting for the final perfect data; especially since the initially entered and final data may only occasionally need to be reconciled.

Of course, it is conceivable that hours may be revised for an individual at some time between the day of activity and the final paycheck; possibly post paycheck. And when so, the records in the system will be revised. The revised data can be pulled into the short-cycle processes that append the activity of today to the recorded activity of the past.

As our data is entire, more of our operational tasks can be data driven. Outputs 8-1 and 8-2 demonstrate a scenario of ramifications of the table of hours.

Sum of Hours								
Provider	Name	PerNo	Exception	Order	Task	Shift	HrType	Total
Ace	Stevens	103	(blank)	976254		40	Day S	7
						50	Day S	2
	Stevens Total							9
	Stewart	101	(blank)	785717		10	Day S	6
						40	Day S	2
				976254		40	Day S	2
						50	Day S	1
	Stewart Total							9
	Ace Total							18
Firm	Jones	10	Sick	(blank)	(blank)		Day S	9
								Jones Total
	Roberts	19	(blank)	785717		10	Day S	2
						30	Day S	7
						10	Day S	2
						30	Day S	7
	Roberts Total							9
	Smith	12	(blank)	785717		10	Day S	2
						30	Day S	7
					Smith Total			
	Firm Total							27
	Grand Total							45

Output 8-1: Report to confirm the hours for each craft involved or not available to the day's work total to expectations.

A supervisor can use the table layer of Figure 8-3 to generate a pivot report layer shown in Output 8-1. With it, the supervisor can review and approve the hours of each direct and contractor craftsperson to assure that total direct and overhead hours for each craft matches expected hours. In the pivot table for a nine-hour day with no authorized overtime, we can see that each craft was engaged for nine hours across task work and overhead events. If there had been overtime we would see it clearly flagged by hour type and included in the craftsperson's total.

Sum of Hours							
Name	PerNo	Order	Task	Shift	HrType	Total	
Jones	10	(blank)	(blank)	Day		9	
Jones Total						9	
Roberts	19	785717		10	Day	2	
				30	Day	7	
Roberts Total						9	
Smith	12	785717		10	Day	2	
				30	Day	7	
Smith Total						9	
Grand Total						27	

Sum of Hours							
Name	PerNo	Order	Task	Shift	HrType	Total	
Stevens	103	976254		40	Day S	7	
				50	Day S	2	
Stevens Total						9	
Stewart	101	785717		10	Day S	6	
		976254		40	Day S	2	
				50	Day S	1	
Stewart Total						9	
Grand Total						18	

Output 8-2: Data transformed to reports to their respective payroll processes: firm and contractor.

Thence, the supervisor can send the hours, as reviewed and approved, down an appropriate paths for processing into systems. An example pair of submittals are shown in Output 8-2. In it we can see organized the details that a payroll and hours distribution clerk

would enter in the system. However, now they can do so without a significant burden of thought, an open invitation for errors the fused timesheet would demand of them.

The alternative to the timesheet hours presentation is to sort the table of Figure 8-3. Clerks can easily do so in the order of their preference.

Better yet would be that the system can import the hours table as-is after being reviewed and approved by the supervisor. This would make the payroll process a very short, rather than the lengthy procedure that it is in some organizations.

We think of the systems of payroll and distribution of hours as one. They are not. The payroll system does not care that hours are posted to work activity. Just that the direct and overhead (e.g., vacation, sick, etc.) hours and resulting paychecks are correct. In contrast, they are not bothered by 100 percent of the day’s hours recorded to only 40 percent of the day’s tasks.

Output 8-3 demonstrates the ramifications for “current at capture.” The pivot of the figure allows the supervisor to confirm that all work tasks show hours and that the hours pass her sanity-check for accuracy. This is an important upstream action to avoid downstream data cleansing.

Sum of Hours							
Order	Provider	Name	PerNo	Task	Shift	HrType	Total
785717	Ace	Stewart	101	10	Day	S	6
				10 Total			6
	Firm	Roberts	19	10	Day	S	2
				10 Total			2
				30	Day	S	7
				30 Total			7
		Smith	12	10	Day	S	2
				10 Total			2
				30	Day	S	7
				30 Total			7
785717 Total							24
976254	Ace	Stevens	103	40	Day	S	7
				40 Total			7
				50	Day	S	2
				50 Total			2
		Stewart	101	40	Day	S	2
				40 Total			2
				50	Day	S	1
				50 Total			1
976254 Total							12

Output 8-3: Pivot to inspect that hours are assigned to all work done and accurate.

The supervisor’s review, revision and approval processes for payroll and distribution can be enhanced by what will be demonstrated in Chapter 11. The hours table can be subjected to aggregation (Chapter 3) upon the order task, and craft type and counts so that

the timesheet data matches the lines of the day's schedule and job plans. When the aggregated timesheet data are joined to the day's schedule and job plans data, we get a super table of huge import.

The super table allows us to compare planned and actual crafts and hours. With it we can flag and measure all the unscheduled and unplanned work to the day vis-a-vis scheduled and planned work. With it we measure schedule compliance across global and local entities and any time period. Finally, this is the same data we would use to get indicators of productivity with respect to job plans that are too loose or too tight or that maintenance capacity is excessive or inadequate.

It is a beautiful thing that some plants may enter craft hours directly into a craft hours tracking system or the CMMS. This may be at the hand of the supervisor, an administrative or the individual craftsperson. In the same system, supervisors will be able to review and approve hours for pay.

However, as criterion for entirety, we must have immediate access to the data. One reason is that system standard reports are not configured to check for goose eggs and accuracy. Nor are they configured to return, as a standard report, the previously mentioned super table of schedule, job plan and timesheet.

In the background of craft hours systems there is always a table equivalent to Figure 8-3. Therefore, we can have real time access rather than being required to wait until the dust settles days later. This is "accessibility upon demand" and the third characteristic of entirety of data. It is the topic of the next section.

8.3. Accessible on Demand

All systems sit on top of a relational database. Between them the two-layer principle is at play. A database is called relational because the total of the data is parsed and stored in subtables. In turn, the total can be reconstructed upon common unique identifiers. The same principle of reconstruction is in play to build super tables as shown in Chapter 3.

"Access on demand" means that we can immediately, upon demand, extract the data of any one or more of the relational tables in the database. In fact, that is what is happening when we call up a standard report from a CMMS or any system.

Our data is "entire" if we have ready access to the tables of any of the total data as a system standard report; assuming we have permission. Many maintenance operations cannot pass the test.

Even more notable is that the test is often failed with respect to one of the most fundamental details to any operation, second only to craft hours to orders. Where are we

Let’s assume we want to extract the history for all orders for which statuses passed directly down the mainstream. These orders did not have to step out of the mainstream; say pass through engineering or an extended approval process. Let’s also assume we want to measure retention times in each stage.

Figure 8-5: A CMMS transaction to extract status history by query.

First, but not shown here, we would use our ingenuity and skills of Chapter 3 to filter by query in Access the orders that meet the criteria identifying them as mainstreamers. Figure 8-4 depicts the outcome.

With the table of mainstream orders in an Access session, we work with the crosstab query. We did not dwell on the crosstab query in Chapter 3, but we will now.

We next import into MS Access the status history of Figure 8-4. In real life, we would have extracted the history from the CMMS with a query interface such as Figure 8-5. Next, we select crosstab query from the create section of the Access ribbon and then pull the table into the upper section of the query view.

Next, we code the design grid as shown in Table 8-1. We will title the query ctqStatusHisoryWide. The ctq element to the title identifies the query as a crosstab query named ctqStatusHistoryWide. Recall that qry identifies our select queries.

Code 8-1: ctqStatusHistoryWide.

Join	None: tblStatusHistory is sole source.			
Field	Order	State	Status	Date
Table	tblStatusHistoryTable	tblStatusHistoryTable	tblStatusHistoryTable	tblStatusHistoryTable
Total	Group By	Group By	Group By	Last

Crosstab	Row Heading	Row Heading	Column Heading	Value
Criterion		“Enter”		

We create groups with the row headings code to the Crosstab sector of the grid. We are grouping by order and state. We have set the criterion to return only the Enter state for each status.

The third column is the meat of the grid. It converts the table from a long or tall format to a wide or flat format.

We can see what the conversion from tall to wide means in Output 8-4 showing what is returned by the code. Notice that with the code, column heading, to the crosstab sector of the grid, each status is now a column rather than a category to the variable Status. Previously all statuses would have presented as a single column as can be seen in Figure 8-4. This is most easily seen by visibly comparing the Status variable of Figure 8-4 to the statuses as variables in Output 8-4.

The last column in the design grid returns the values to appear under the status columns. We have seen our options as sums, averages and so on. In this case we just want our dates. Using last, will call up the last date should there be multiples. For example, work may have been completed and then revisited for some reason; causing a sequence of enter, exit, enter, exit.

Order	State	CLOSE	COMP	INPRG	NAPR	NAWA	PLIP	SCHD	WPLN	WSCH
785717	Enter	10/24/2020	9/30/2020	9/30/2020	8/20/2020	8/19/2020	9/13/2020	9/28/2020	8/21/2020	9/17/2020
976254	Enter	10/24/2020	9/30/2020	9/30/2020	8/27/2020	8/26/2020	9/21/2020	9/29/2020	8/29/2020	9/23/2020

Output 8-4: Output of cross tab presenting date in wide form.

Let’s insert an aside here. One red flag is the multiple occurrence of a state that should only occur once for a given order. We can use the history table to flag such cases. An example would be work that was completed more than once before the final closing status. Now to get back on topic.

Output 8-4 shows that what is returned are the enter dates for each status. If we want to expand the detail of the super table in Chapter 3 to include status history, we only need to join this data upon order numbers in the super table. Because we have that option, there are almost endless possible ways we can extend the power of the super table for planning, organizing and controlling the maintenance operation.

Let's imagine an analysis with our status history crosstab table as its feedstock. Say that we want to extract the total days from the first request through to completion. Additionally, we want to extract the elapsed time spent to plan each order. The code of Code 8-2 for qryStageElapsed returns the output shown in Output 8-5.

Code 8-2: qryStageElapsed.

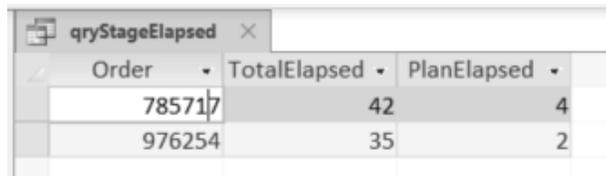
Join	None: ctqStatusHistoryWide is sole source.		
Field	Order	TotalElapsed: [COMP]- [NAWA]	PlanElapsed: [WSCH]- [PLIP]
Table	ctqStatusHistoryWide	ctqStatusHistoryWide	ctqStatusHistoryWide
Show	Yes	Yes	Yes

Recall that a characteristic of a table, compared to a spreadsheet, is that only horizontal calculations are allowed. With ctqStatusHistoryWide, we can go horizontal. Accordingly, it is now an easy step to calculate elapsed days.

The first column of the grid calls in the orders. The second column computes the total elapsed days to completion. The third column computes the elapsed days in the planning stage. Both horizontal calculations are the enter date of the terminal stage minus the enter date of the initial stage.

There are always different roads to Rome. If we had not specified Enter as the criterion, we would have been provided with columns for all states. Thence, we could have calculated elapsed day in different ways, for example, exit minus enter of the focal statuses.

Output 8-5 is the returned table. And once again we can join the findings, as a variable, to any other table with order number as the common identifying variable.



Order	TotalElapsed	PlanElapsed
785717	42	4
976254	35	2

Output 8-5: Elapsed days for two intervals of interest.

Within the context of other tables, we may want statistics such as average, variance, sum, count, min-max and first-last for elapsed time. We would form them with the aggregation functions as explained by demonstration in Chapter 3.

We could also conduct aggregation analyses for our example set of orders. We would remove order from the grid and build aggregation variables in our table. It is left to the reader to do so.

Chapter 12 to optimize the risk whenever an asset is sitting dead in the water until returned to readiness will demonstrate the exploration of retention in much more meaningful ways than average, spread, extreme, age, etc. We need to know and visualize the probability an order will remain in a stage for some length of time. Then we need to know and visualize the probability of exiting the stage just after having been in the stage for the length of time.

Chapter 2 introduced this as duration analytics. With respect to status history, the feedstock data is a composite of enter, exit and specific event. Only with a status history table can we have our composite variable. By specific event we mean that the request or order passed to one or alternate next stages.

Bibliography

Alexander, Michael and Kusleika, Richard. Access 2016 Bible. John Wiley & Son, Inc. 2016. Parts 3 and 4.
Oesko, Suljan. Pivot Tables In-Depth for MS Excel 2016. Suljan Oesko. 2017.

Chapter 9

Workload-Based Budget and Variance

Recall from Chapter 1 the definition of availability. It is the probability a plant, subsystem or item will be in a state to perform a required function at specified standards of performance under given conditions when called for; assuming cost-effective support with respect to working conditions, processes and resources.

The proposition is simple in principle and action. Plant or system availability is sustained by recognizing and cost effectively delivering the maintenance workload to sustain plant performance and condition.

The problem is that the most observed practices for maintenance budget and variance are inconsistent with the proposition. Plants budget and control dollars for labor, materials, etc., This contrasts with budget and control workload and the labor, materials, etc. to execute it.

To succeed at sustained availability and condition, and optimal maintenance capacity, all roads must pass through dual-dimensional budget and variance. There are three objectives. First is to confirm that the determined sustaining workload is being accomplished on monthly pace. Second is to confirm that execution is consistent with budgeted productivity. Third is to use the plant as a model to progressively align the craft force, as maintenance capacity, to workload.

This chapter will explain, by demonstration, dual dimensional budget and variance. It will begin with explaining the principles and calculations to build the budget and to measure the actuals against it. What is explained and demonstrated only requires skills with MS Access and Excel Pivot. In other words, budget and variance requires the skills of working with data as taught in Chapter 3 rather than skills of a data scientist.

The methods to forecasting workload and resources can range from basic to sophisticated. The former, is largely averages combined with expertise in the plant. The latter also draw upon familiarity but engage time series analytics, the subject of Chapter 10. Because basic approaches can usually get us where we need to be, the chapter will stick to the averaging rather than step into advanced analytics.

Three demonstration cases will be presented. They are to report month and year to date variance from budget, evaluate for systemic variance while seeking outlier orders, and align craft force capacity. The data to the demonstrations are available for download from the webpage provided by the preface. It is contained in the Excel File,

DataBookAsssetMgtXlsx.xlsx, as the sheets tblHistory2019, tblActual2020, tblCraftBaseline and tblCraftSample.

All queries to budget and variance are built in MS Access and then reported out in Excel pivots. The code to forming the Access objects are provided throughout the chapter.

9.1. Framework for Budget and Variance

To prove and assure business-optimal maintenance spending, plants need to break away from the typical and largely ineffective approach that is essentially a spending allowance. This section will describe two-dimensional budget and variance control as the break away. In contrast, spending allowance is one-dimensional.

9.1.1. Mission, Structure and Derivation

To begin with, we need to define the mission, structure and derivation of the budget and variance system.

Mission: Just as maintenance is a sub-budget in the plant's grand budget, there is a sub-budget for plant aggregate production. Consequently, the mission of the maintenance budget and variance system is to allow the plant to connect maintenance work to the readiness of production assets that deliver the aggregate plan. It will concurrently allow the plant to connect work to staying abreast of plant deterioration, thus, protecting against being overridden by work to sustain readiness. Along with doing the work to assure sustained readiness and counter deterioration, the third aspect of the mission is to cost effectively do the work.

Structure: There are two levels to any budget and variance system: structure and derivation. The system must have a structure to which the details of budgeted and actual spending are attached and from which variance can be clearly seen. Figure 9-1 depicts the choices for structure.

The left-most structure shows only one dimension of information: total cost or cost with respect to expended and engaged resources. The right-most structure shows the two dimensions of information: activity and cost per activity.

With one dimension, there can only be a single net variance. With two dimensions, there is variance for each of the dimensions: variance due to activity and variance due to cost per activity.

It is apparent that the contrast in insight is akin to navigating to a destination with latitude and longitude or tracking the number of miles allowed to get there. With miles as control, if we arrive at all, we will almost surely travel excess miles.

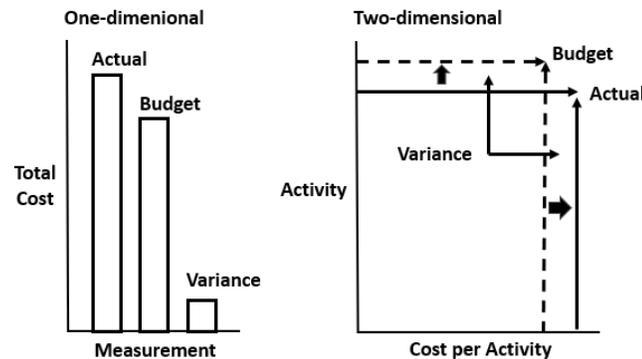


Figure 9-1: The contrast between one- and two-dimensional structures is insight.

Derivation: The second level of a budget and variance system is how we formulate the details of the budget. The question is asked and answered at each intersection of cost center, work type, cost type, craft type, asset type and components, etc. The intersections are the “groups” on which the system will budget and monitor variances.

The choices for derivation range from zero-based budgeting to predictive-based budgeting. The term “range” is appropriate because every budget group will be at the extreme of the range or somewhere in between. Invariably along the range, we build up from activities to the direct and indirect resources to conduct them. It is important to note that one-dimensional budgeting depicted in Figure 9-1 does not exist on the continuum because workload is not its basis.

At one extreme, zero-based budgeting establishes named tasks for named equipment and their components. To budget we must identify every failure mode, every maintenance strategy to each mode, every task to each strategy and every resource to every task. We would also establish when the tasks will be called for. Obviously, this is perfection, prescient and impossible except for very limited special cases.

At the other extreme, predictive-based budgeting uses statistics-based methods and models to establish the expected number of tasks, and the resources and cost to do them. In contrast to zero-based, it does not attempt to name in advance the failure modes and the tasks for each item of equipment and its components.

The choice is not chiseled in stone. The approach taken will reflect the best strategy for each of the many budget groups across the plant.

An example is any type of maintenance driven by timing. In a perfectly developed situation, we can query the appropriate system to form a budget for these work categories. If not so perfect, we can query status history to quantify the incurrence of work.

Partial or fully predictive-based derivation will be the most common case. This is because plants rarely have the perfection of computer systems, roles and data to make full zero-based derivation remotely feasible. Furthermore, systems for scheduled maintenance lack perfection and, thus, make a predictive derivation the best choice.

Predictive-based derivation also allows the plant to get around limitations and weaknesses in its historical data. Consequently, the plant can begin immediately to tap the benefits of two-dimensional budgeting and variance. The operation will not need to first improve or enforce operational rules-of-conduct as needed to eliminate the limitations and weaknesses, and then wait for good data to amass. Time waiting is big money, forever lost.

As already said, the approach to derivation falls outside the range between zero- and predictive-based derivation. The budget is essentially a “spending allowance” rather than spending derived up from activity. The budget for each of the many budget groups is the sum of its spending allowances for labor, parts and materials, services, etc.

So where do the one-dimensional spending allowances come from? Most importantly, the answer disqualifies the approach as meaningful.

The allowances for the categories of resource are typically the past year projected into the budget year after adjustments such as inflation. Too often they are the outcome of negotiations or some challenge to spend less while doing more but without the budget analysis to know what is feasible as “less and more.” Sometimes they are based on industry indexes such as percent of plant replacement cost. Most cases are the interplay of adjusted history, replacement value, negotiation and challenge.

The Chapter will keep derivation basic in order to demonstrate dual-dimension budget and variance. Workload will be based on requested and approved work divided by 12 months. Hours, materials, etc., will be averages to each budget group. The chapter will also speak to cases in which the averages to some resources are not trustworthy data.

Otherwise, there are very slick ways to look at history and build forecasts with time series analytics. Time series for forecasting budgets, among many purposes, is the subject Chapter 10.

9.1.2. The Two Dimensions

In contrast to a one-dimensional structure, a two-dimensional structure complies with the core-most rule-of-gravity in cost accounting. All costs have two dimensions. Figure 9-2 shows the dimensions to be activity (job count) and factors of cost. Let’s look closer at each.

Activity Dimension: The anchor dimension is activity (AKA, workload) because everything including indirect work, starts at the direct work that needs to be done. For a

maintenance operation, activity is the number of jobs for each budget group. As mentioned previously, all budget groups are bounded with respect to categories such as cost center and accountability, work type, order craft type, craft types to the order, asset type, etc.

For a host of profound reasons, it is not practical to attempt zero-based budgeting to set the activity budget. We would, instead, reach back into the historical data for incurred and occurred work orders. Time series analytics could also be used to arrive at the activity dimension of the budget, but statistical averaging is usually practical. From whatever method taken, we will most likely set the budget on smoothed levels of activity.

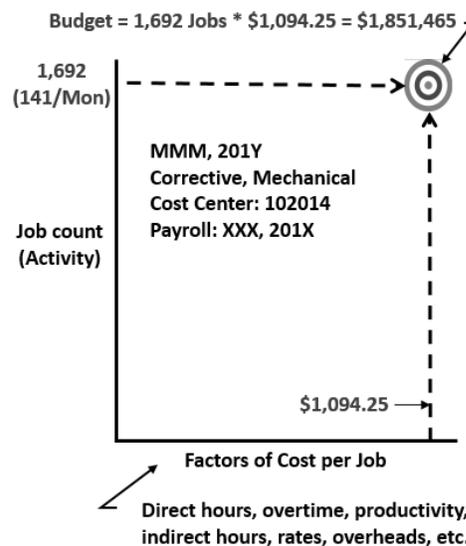


Figure 9-2: To be meaningful, budgets must be the intersection of two dimensions.

Factors-of-Cost Dimension: The other shoe to the activity dimension is to quantify the resources that will be engaged and consumed to execute the budgeted activity, and then the direct and possibly indirect costs of the resources. As shown in Figure 9-2, the dimension can be called “factors of cost.” The factors or units of cost, as an extension of activity, are derived according to the nature of each budget group.

Whereas, the activity dimension ties back to the plant’s aggregate production plan and staying abreast of its deterioration, the factors-of-cost dimension has its own connections to profit and enterprise ROI. First is the indirect connection to revenue through the adequacy of resources available to deliver the budgeted activity. Second is the connection between enterprise expense and the direct and indirect resources that are pulled in by the activity budget.

9.1.3. Budget

Let’s step from the concept of dual-dimensional budget to an actual working Excel spreadsheet. Figure 9-3 compares two- and one-dimensional budgets. The comparison dramatically highlights the distinction.

Both structures have two characteristics in common. Both are the same three budget groups. Both present a total spending budget for three costs: hours and payroll, materials and services.

However, the difference in transparency looms large for insight and decision-making. First, the activity dimension is added for each budget group. Second, average labor hours per job and total hours are added. Finally, the averages per job for payroll, materials and services per job and their totals are added.

Two-Dimensional Budget

Year 201X		Two-Dimensional Budget >> Corrective Maintenance >> Cost Center 102014						
		Work Orders		Labor		Material & Services		Total
		Month	Year	Hours	Payroll	Material	Services	Expense
Mechanical	Total	141	1,692	33,663	\$ 1,851,465	\$ 960,000	\$ 210,000	\$ 3,021,465
	Per Order			19.9	\$ 1,094	\$ 567	\$ 124	\$ 1,786
Electrical	Total	67	804	6,151	\$ 338,305	\$ 40,200	\$ 8,040	\$ 386,545
	Per Order			7.7	\$ 421	\$ 50	\$ 10	\$ 481
Instrument	Total	13	156	1,147	\$ 63,085	\$ 41,100	\$ 1,500	\$ 105,685
	Per Order			7.4	\$ 404	\$ 263	\$ 10	\$ 677
		221	2,652	40,989	\$ 2,252,855	\$ 1,041,300	\$ 219,540	\$ 3,513,695

One-Dimensional Budget

	Payroll	Material	Services	Total
Mechanical	\$ 1,851,465	\$ 960,000	\$ 210,000	\$ 3,021,465
Electrical	\$ 338,305	\$ 40,200	\$ 8,040	\$ 386,545
Instrument	\$ 63,085	\$ 41,100	\$ 1,500	\$ 105,685
	\$ 2,252,855	\$ 1,041,300	\$ 219,540	\$ 3,513,695

Figure 9-3: The comparative guiding value of the budget structures is immediately apparent.

In contrast to one-dimensional budgets, we are presented with every natural detail of planned activity and spending. But wait, there is more! Behind the curtain of the shown detail are the body of records, datasets and models by which the terms were developed. It follows that the budget team and management can drill down to them as needed for insight along the way to arriving at consensus.

The difference in insight will no doubt change immensely the discussion of the budget. For activity, the discussion may be why the expectations are what they are. It may be whether select budget groups can be reduced and earmarked for catchup when current business conditions improve. For the factors of cost, the discussion may be if the resources and associated costs are reasonable and where and how is it possible to better align and reduce them.

9.1.4. Variance

With a one-dimensional structure, the plant has very little power to steer and control its monthly activity and cost. And, of course, the whole purpose of budgeting and variance. The immensity of what is lost is best demonstrated by what is gained when we operate with two-dimensional budgets and variance.

Figure 9-4 is a conceptual depiction of what cost accounting calls “flexible” budgeting as compared to “static” budgeting.

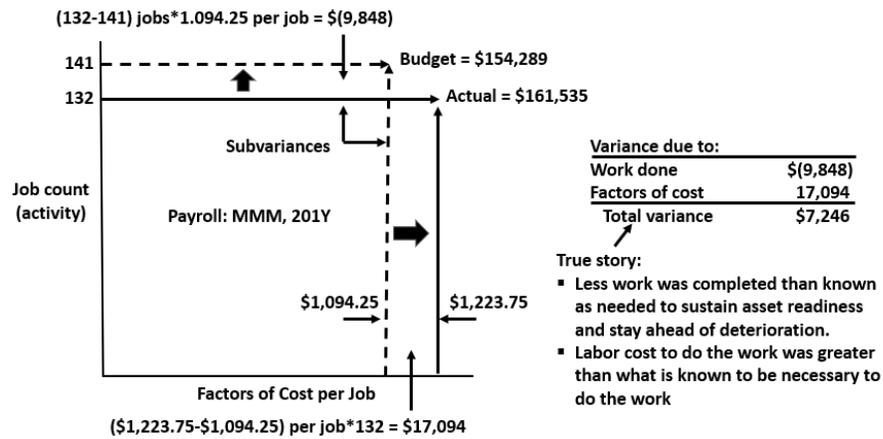


Figure 9-4: The true story of the month can only be known through dimensional subvariances.

Conceptually, flexible budgeting is the overlay of two rectangles: budget and actual. In a world of certainty, they would match each month throughout the budget period. Of course, they never do which is why there is variance reporting.

The cross-hatched areas are the variances to each dimension of the subject budget group. The calculations of the variances are straightforward and, as shown Figure 9-4, are as follows:

$$\text{Variance due to count} = (AJC - BJC) * BHJ \tag{9-1}$$

$$\text{Variance due to hours} = (AHJ - BHJ) * AJC \tag{9-2}$$

Where:

BJC = Budget job count.

AJC = Actual job count.

BHJ = Budget hours per job.

AHJ = Actual hours per job.

that we can trace from the trailhead of each subvariance in the variance report to the outlier orders and systemic forces that caused the variances.

Two-Dimensional Variance Report

Month MMM, 201Y							
Two-dimensional Control Month >> Corrective Maintenance >> Mechanical >> Cost Center 102014							
		Activity		Cost			
		Orders	Dir Hrs	Payroll	Material	Services	Total
Budget	Total	141	2,805.25	\$ 154,289	\$ 80,000	\$ 17,500	\$251,789
	Per WO		19.90	\$ 1,094	\$ 567	\$ 124	\$ 1,786
Actual	Total	132	2937.00	\$ 161,535	\$ 117,876	\$ 14,916	\$294,327
	Per WO		22.25	\$ 1,224	\$ 893	\$ 113	\$ 2,230
Total Variance		(9)	131.75	\$ 7,246	\$ 37,876	\$ (2,584)	\$ 42,538
Due to work done			(179.06)	\$ (9,848)	\$ (5,106)	\$ (1,117)	\$ (16,072)
Due to factors of cost			310.81	\$ 17,094	\$ 42,982	\$ (1,467)	\$ 58,610

One-Dimensional Variance Report

		Cost			
		Payroll	Material	Services	Total
Budget		\$ 154,289	\$ 80,000	\$ 17,500	\$251,789
Actual		\$ 161,535	\$ 117,876	\$ 14,916	\$294,327
Variance		\$ 7,246	\$ 37,876	\$ (2,584)	\$ 42,538
Actual per WO		\$ 1,224	\$ 893	\$ 113	\$ 2,230

Figure 9-5: The comparative ability of the variance structures to tell the true story.

The ability to trace to the roots of variations gives us the deep insight on which to decide upon appropriate adjustments to planned spending as the year unfolds. It also sharpens the plant’s ability to plan and control the next budget year. Just as exciting, reliability and maintenance professionals can find in the answers exactly where business profits and returns can most be advanced through surgically redesigned maintenance and reliability practices and procedures.

Finally, we can summarize the contrast between two- and one-dimensional structures as it was well verbalized by a senior executive. She admitted, “Each month we have big questions about maintenance, but know we can’t have an answer. And if we do get an answer, we know it is not a good one. Now we can get good answers.”

9.2. From History to Budget to Variance

The previous section explained the principles and calculations of dual dimensional budget and variance. Figure 9-3 presented a detailed budget and Figure 9-5 a detailed variance report.

The Excel-based reports allowed us to see the principles in full form. However, we can show the details in interactive pivot charts and tables rather than the somewhat

impractical presented form of a spreadsheet. This section will demonstrate how to build the elements of the figures in MS Access and present them in Excel pivot charts.

The Figures 9-3 and 9-5 were also full scope. They included all factors of cost to the plant the example was taken from. However, the demonstration will focus on hours per job. Because it is operationally pragmatic to narrow budget and variance to hours, a plant may decide to operate as demonstrated in this section rather than extend to a full-scope system.

This is pragmatic because hours are both the greatest and most controllable of all cost factors. They also most directly decide sustained readiness, condition and maintenance expense. As a variable in maintenance, craft hours are the knob to most adjust maintenance capacity, budget and variance to the workload. In other words, a plant may decide to budget and variance on what matters most.

Repair parts are a good counter example. Although a significant proportion of the maintenance expense, they follow work rather than are directly controlled. This is equally so for most cost groups in maintenance expense.

Having said that, the value of a full-scope budget is that we can present to management a full budget built up from the workload to cost effectively sustain plant performance and stay abreast of deterioration. Stated differently, we can present to management the defensible detail of having put pencil to paper.

Another perspective is that the enterprise will need a bottom line for the maintenance operation within its overall total. Although not described in the chapter, all indirect and overhead costs would be layered on the full scope of direct maintenance expenses.

It is always possible that what is found as required in the proposed budget cannot be funded due to circumstances such as business cycle. When so, the maintenance operation has a factual basis upon which to collaborate with plant management to revise the workload by design: putting pencil to paper. Along the way management can know and stay on top of the ramifications. Furthermore, the revisions can be reflected in the next budget cycle.

The methods to be demonstrated in the chapter for direct hours would be emulated for all cost groups in the maintenance expense. As the chapter unfolds, focused on direct hours, it will still describe how to build a full scope system. The readers will be able to work the described alternative with the degree of instruction provided.

9.2.1. The Case

The case is simulated data of the typical variables to a maintenance operation. They recognize budget groups as permutation of the plant's variables which will at the least be cost center, maintenance type and order craft type. Further granularity could include the

categories of crafts as disciplines engaged in each order, crew and asset type. The big point is that all groups are pinned to workload.

The demonstration will present as two cost centers although a plant may be tens of centers. It will also include only two of the five maintenance types defined in RCM: condition based and corrective. Finally, the case will group orders by two lead craft types: mechanical and electrical. We could, but will not, include craft disciplines such a mechanic and electrician. The extension will be demonstrated in Section 9.4.

Craft hours can be direct employees and contractors. The case assumes both sources are engaged in the same workload and, because we report on hours rather than dollars, we have no need to distinguish them as budget groups. Alternatively, a plant may choose to add the distinction of direct employees and contractors to the budget grouping scheme.

The number of variables in the budget groups are limited in the case because of the work to simulate a typical dataset of all real-world budget and variance groups is prohibitive. However, the effort to build budgets and report variance for the many groups we would expect in real life is no different or greater than what is demonstrated.

The case simulates work history for 12 months and actuals for the first three months of the variance year. In real life we may, and easily can, reach back farther than 12 months. We may do so if we suspect there are longer-term changing patterns in the workload we should consider in our budget.

The case will follow the progression of input tables, queries and report out pivots as shown in Figure 9-6. Note that once the system of input tables, queries and pivots are built, updating `tblActual2020` for the latest month is the only action needed to update the set of Excel pivot reports. The sausage is automatic once the meat is dropped into the grinder.

Because of this automation, the monthly variance report is available almost as soon as the data are complete for the month. Furthermore, because of being scoped on hours, our data in the CMMS and by entirety (Chapter 8) is complete after the last timesheet is completed at the front line (Section 8.2). Thence, we can have our report for the month within an hour with respect to this most actionable dimension of plant performance, state and operating expense.

The history of activity, 960 orders, is what would be queried from the CMMS. We have titled it as `tblHistory2019`. To form the data into a budget, the history will be grouped by the budget grouping. That happens in `qryBudget2020`.

Beginning with the first month of the variance year, we bring a table of extracted data, `tblActual2020`, of orders completed in the variance year. Within the `qryActualGrp2020`, the actuals data are grouped to match line by line the groups of `qryBudget2020`.

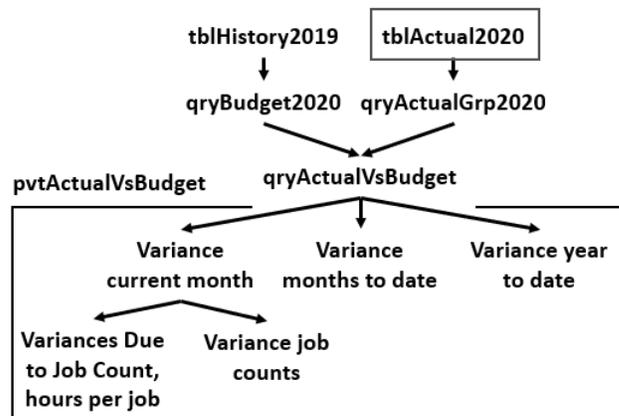


Figure 9-6: Schematic of the budget and variance process.

Because the budget and variance queries consist of mirror rows for budget groups and variables, we can join them to form the query on which variance reporting is done. In `qryActualVsBudget` we set up the variance equations 9-1 and 9-2, and total variance. The result of the calculations is reported visually as an interactive Excel pivot chart.

Of the almost infinite possible charts and tables possible by pivot on top of `qryActualVsBudget`, four will be presented to generally demonstrate just a few of the many thought processes that dual-dimension budget and variance make possible. As charted in Figure 9-6, they are as follows:

- Variance of all groups in the current month: March.
- Variance in job counts in the current month.
- Variance for each month to date: January, February and March.
- Total variance for year to date: aggregation of the months to date.

Of course, a maintenance analyst will search through the returned variances from many directions, limited only by their forensic senses. The forensics will reflect the circumstances and nuances of each plant and variables to their budget groups. The query `qryActualVsBudget` allows tremendous flexibility. Pivots allow us to interact with the flexibility with no more effort than click and drag.

It was previously promised that, along the way, the full-scope case would be described to the degree the reader will need to be able to build it. At this stage, the only difference is that we would have included labor cost, parts and materials, and other costs as variables in `tblHistory2019` and `tblActual2020`. In real life, we would typically do that on general principle, regardless of our chosen focus.

9.3.1. The Case

To create statistical mass, we will choose to conduct the analytic cumulatively. At each month, all orders up to the month will be included. This sharpens the search as the year unfolds. It also causes past orders to be retested as there are more observations.

The schematic for the search is shown in Figure 9-8.

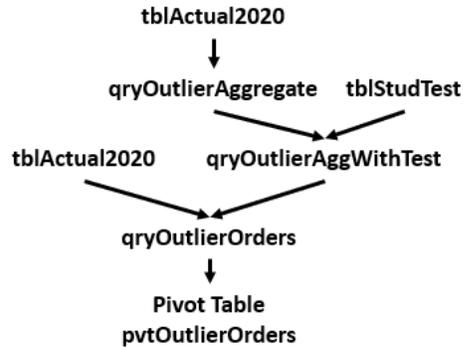


Figure 9-8: Schematic to return table of outlier orders for year to date.

We start with the actuals for the months to date. In qryOutlierAggregate, the count, mean and standard deviation to each budget group are created with aggregation variables. In qryOutlierAggWithTest, qryOutlierAggregate is extended to include the Student T for each group by joining with tblStudTest. In qryOutlierOrders, the group statistics are joined with the table of actuals and each order is tested as an outlier. Finally, the findings are interactively inspected in pvtOutlierOrders.

9.3.2. Year to Date Aggregation

For the budget groups, with qryOutlierAggregate, we are transforming the actuals to date to the aggregation variables we need to seek outliers. Mean, standard deviations and count are returned for each group. Code 9-4 is the code to the query.

Code 9-4: qryOutlierAggregate.

Join	None: tblActual2020 is sole source		
Field	CostCenterSum: CostCenterA	MntcTypeSum: MntcTypeA	CraftTypeSum: CraftTypeA
Table	tblActual2020	tblActual2020	tblActual2020
Total	GroupBy	GroupBy	GroupBy
Show	Yes	Yes	Yes
Field	MeanHours: HoursA	StdDevHours: HoursA	CountOrders: HoursA

Table	tblActual2020	tblActual2020	tblActual2020
Total	Avg	StDev	Count
Show	Yes	Yes	Yes
Field	IDGrpAct		
Table	tblActual2020		
Total	Group By		
Show	Yes		

Notice the absence of date as a grouping variable. The absence causes the aggregation to be year to date. If our choice had been to test individual months rather than cumulatively, we would have included pulled the variable YrMonDateCompA from tblActual2020 into the query as a grouping variable.

There is another point of note in the code. Notice that we have renamed the variables for cost center, maintenance type and craft types. For example, CostCenterA has been given the alias of CostCenterSum. The reason is to facilitate joins further down the trail when it is good to avoid identically named variables in the joined tables. A bit of proactive coding.

Output 9-10 is returned by the code of Code 9-4.

IDGrpAct	CostCenterSum	MntcTypeSum	CraftTypeSum	MeanHours	StdDevHours	CountOrders
7001CondBasedElect	7001	CondBased	Elect			
7001CondBasedMech	7001	CondBased	Mech			
7001CorrectiveElect	7001	Corrective	Elect			
7001CorrectiveMech						
7002CondBasedElect				2.09	0.87	34
7002CondBasedMech				2.05	0.88	64
7002CorrectiveElect				6.50	2.25	16
7002CorrectiveMech				12.74	3.77	50
				2.06	0.79	33
				1.74	0.65	19

Output 9-10: qryOutlierAggregate returns mean and standard deviations of hours and count to each budget group.

We still need to extend the table to include a Student T score for our chosen confidence level and the count of orders to each budget group. However, there is no such function in Access. There is in Excel.

Therefore, we will build a Student T table in Excel. Figure 9-9 shows what is constructed in Excel as tblStudTest. It is a table of 1,000 scores for counts from 2 through 1001. It is available from the collection of tables in file SkillsForCareerSecurity_Datasets.xlsx.

and `qryActualGroup2020`. In the inserted query we somehow eliminate the outliers. From there we would continue the remaining scheme without the outliers.

However, with the scheme we can only evaluate systemic variance with respect to average hours per job. This is because the outliers have also been removed from the variances due to count. This may be fine if hours are the focus.

Alternatively, if we want to explore both variances as systemic, we can get more sophisticated. Most directly we can estimate hours by the average of non-outliers to the associated budget groups. We would modify our initial scheme as necessary to replace the reported dependent variable of the outliers with the estimated averages for each group and then continue to the final variance table.

A deeper approach would be to estimate outcome upon linear regression. The method was explained in Chapter 7.

Recall, we would create a super table, but with the non-outlier orders. We would conduct the regression to create a model of variables that are most strongly related to hours. In turn, we would feed the independent variables of the outlier orders to the model and receive an estimate for each. With the estimates we would insert them in the scheme built for receiving the estimates.

9.4. Maintenance Craft Capacity

It is easy to fall into thinking that planning and scheduling, and executing work decide craft force productivity. The reality is that, if we cannot align our craft force as maintenance capacity to the capacity required to fulfill workload, we will be either unable to maximize productivity or to deliver sustaining workload.

Over capacity is the greatest cause of maintenance expense overrun. Alternatively, under-capacity may present as high craft force productivity, but the plant will eventually fail to meet its production plans and its value as an asset will decline.

We saw in the variance analysis of Section 9.2 the possible red flags of potential mismatches of craft force capacity and workload. In the case of variance, we seek cause and find that some are driven by capacity. We would also want to determine if we were head faked while seeking appropriate capacity because job planning has systemically prepared job plans that are too loose or too tight.

Therefore, an organizational goal is to arrive at a work force that is a reasonable optimal match to workload with respect to effective person years. Because we have budgeted workload, we have the primary elements at hand with which to tease the optimal craft force from our data.

However, we cannot simply add up the total hours expected for each craft category to execute the year's workload and then divide by hours per craft per year. This is because maintenance is a "job shop" operation of varying arrival counts, varying jobs, varying resources to each and varying time to complete.

One trail for seeking the optimal is to marry the process and methods of budget and variance and the processes of planning, organizing and scheduling each day's and week's work execution. We can join, in super tables, the variables of the schedule, basic job plan details for craft category and headcount, and timesheets. With the super tables we can work with analytical approaches to seek out an optimal craft force.

Another commonsense method is to simulate the number and mixture of crafts to execute a collection of generalized single-day workloads. They would be templated on the budgeted smoothed workload.

Simulation can be done with the functionality of scheduling software such as MS Project. The software will assign craft disciplines and grades from a pool of crafts to the day's tasks until the proper compliment of resources are not available to do the next task. By trial and error, the mixture and count of the craft pool will converge on the optimal craft force.

The strategies for designing a craft force by analytics are the subject of Chapter 11. However, the budget and variance process also has a role in finding the optimal capacity.

The budget quantifies the annual workload. When execution is by smoothed workload or other patterns, we can spot indicators of misaligned capacity. As we saw in Section 9.2.5, an indicator of over capacity is finding that we exceeded smoothed workload. Another indicator of misaligned capacity is when our reports show over and underrun for hours.

Ultimately, we create a loop of budget and variance to scheduled and unscheduled work, job plans and timesheets to gravitate to craft force. As just mentioned, the interplay of methods will be a subject of Chapter 11. This section will explain the methods of budget and variance with respect to seeking indicators of poor capacity match.

9.4.1. The Case

The differences in the case and earlier sections are the aggregation of craft disciplines. The previous sections aggregated work by lead craft without regard for disciplines. In this case, we will set up the dual-dimensional structure for the craft disciplines to the work order.

For example, we have electrical and mechanical orders. For each there can be electricians, mechanics and more. We want our dual-dimensional structure upon them

rather than the aggregation on lead craft as we have demonstrated throughout the previous two sections.

There is another distinction. In the previous section we created a budget from the last 12 months and then compared the data to its subsequent variance year. In this case, we take one or more years back from now to set the baseline. In turn, we contrast the actuals of the last 12 months. Recall that the baseline is work requested and approved rather than work completed as it is for actuals.

For the case, a new set of tables has been created. The only reason is to dodge the drudge of extending the 960 simulated orders to include multiple craft disciplines. The tables to the case are of an adequate number of records to demonstrate the processes that would be scaled up for a full set of real-life records. It is left to readers with their own data to work the full analysis as it will be demonstrated.

The tables are available as sheets `tblCraftBaseline` and `tblCraftSample` in the previously mentioned Excel file, `SkillsForCareerSecurity_Datasets.xlsx`.

Figure 9-11 shows the scheme from data tables to queries to Excel pivot chart. Once constructed, the process through the queries is automatic. Whenever the input tables are updated, the data will flow through to `qrySampleVsBaseline` and to Excel `pvtCraftCapacity`.

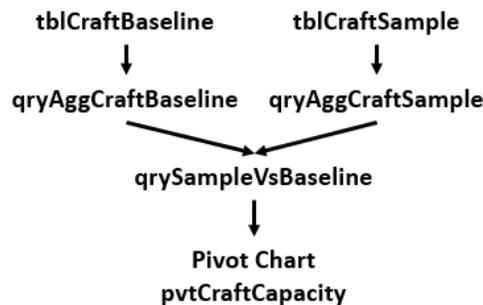


Figure 9-11: Tables, queries and pivot to analyze craft force maintenance capacity.

The explanation will begin at the end. It will present representative pivot charts and describe some of the thought processes they suggest possible. After that we will see how we got there with the queries in Access.

9.4.2. Capacity Report and Analyses

Output 9-14 shows two pivot charts from `pvtCraftCapacity`. The left most chart distinguishes between order types, electrical and mechanical, and then reports the respective variances for electricians and mechanics.

Chapter 10

Through the Lens of Time Series

At the mention of time series, we think of a line chart of some variable as its vertical axis and unit of time as its horizontal axis. It is a form of insight that may be one of the most pervasive of all insight deliverables. It is hard to imagine managing without our many routinely disseminated line charts.

However, what is customary, since invented by William Playfair in 1786, is marginal insight vis-à-vis modern possibility in which analytics are so accessible. This is because what is typically presented with line charts has buried within it a trove of essential insights. At most, with the customary, we can see movement and spot outliers and bad data.

In contrast, we use time series to understand the past and forecast the future upon the understanding. The big idea is to quantify the main features and randomness of the series.

The main features are trend and seasonal or cyclical variation that can be modeled deterministically. A third feature of time series is the degree that observations close together in time are correlated or serially dependent.

A systemic change in the time series that does not appear as periodic is usually called trend although naming, as we shall see, is not fixed. A repeating pattern within the fixed period such as year is a seasonal or cyclic feature. There can be cycles that do not match the fixed period. We would recognize them in the trend feature.

The methodology of time series analytics is to distinguish and explain the main features and correlation using dissection and statistical models. Once a good model is fitted to the features and correlation, we have the means to better understand what happened and if desired, forecast the future on what happened.

The variables of interest to time series are likely to be the predictor variables to a dependent outcome variable. We may start with the time series to an outcome variable to look for undesirable patterns and then investigate the time series to the predictor variables. We may have determined the predictors of greatest interest by other types of analytics such as regression and ANOVA.

As we ponder a time series, we must also be aware that the series may be random, thus, unpredictable. It makes no sense to forecast on randomness. We should also be aware that there is a possibility that a series can appear deterministic when it is actually random. These are called random walks and can fake us into misguided actions.

And then there are forecasts. Upon a deterministic series, a forecast relies on the assumption that the present circumstances will continue. If we are forecasting beyond a year, it would be better to regard the insight as a scenario.

The introduction of a time series points to a body of questions to ask of a series. As expressed by the opening paragraphs, a traditional line chart does not allow questioning beyond shallow.

This chapter will explain by demonstration how to answer the deeper questions. Are there seasons or cycles across the fixed periods? After removing the cycles, what does the adjusted series (trend) look like? Are there cycles in the trend spanning multiple fixed periods? Is a current period influenced by one or more previous periods? Can the series, adjusted for cycle, be modeled or is it a random walk? Is there drift in the random walk? Are there lead-lag patterns between variables? Can we comfortably form a forecast upon the series? How far out can we safely forecast?

Inspecting the predictor variable to an outcome was mentioned. The previous chapter calculated averages upon workload history for the budget period and groups. We could have used the practices of this chapter to arrive at workload upon deeper understanding of the history and nuances of the workload. We could have also explored for trends in hours per types of orders and other influencing circumstances.

Maintenance practitioners wax poetic about key performance indicators (KPI). It is hugely insightful to explore patterns in the KPIs with time series analytics. Furthermore, we can seek and validate the strength of lead-lag patterns. This takes us to an additional question. Are our intuitive assumptions for lead-lag true and, if true, significant?

The sections to follow will introduce what is called a time series object and the R functions to work with series and to answer the many enumerated questions. All will be shown as R in action.

Rather than maintenance data, the engaged data have been selected to demonstrate principles and practices. It is better to use demonstration data instead of data from a CMMS. This is because with CMMS data, we would be evaluating for the characteristics that may or may not be present in the subject plant. In this chapter we are demonstrating how to evaluate data so that the readers can, in turn, evaluate the characteristics of any of their own datasets.

The data to the chapter is either internal to the R software or available for download at the webpage given in the preface. All are contained in the Excel file `SkillsForCareerSecurity_Datasets.xlsx` as the worksheets `tblMaineEmploy`, `tblUsEmploy`, `tblChocBeerElect`, `tblApprvBuild`, `tblWkCycle` and `tblStochSeries`. The R dataset is `AirPassengers`.

The script to the chapter is the R file named `ThruLensOfTimeSeries.R` is available at the same webpage. Of course, while following along, the reader can write and explore the code as an R session as it is presented and explained, one occasion at a time. However, the R file allows the reader to paste the entire script of code to an R session.

The reader will need to load the R libraries of `library(xlsx)`; `library(ggfortify)`; `library(forecast)` and `library(lubridate)`. Recall that if not already installed on a reader's computer, it is done with the `install.packages` function.

The chapter will stick with base graphic functions to time series. Rather than be `ggplot2`-centric, it is good for the reader to gain experience with the base graphic functionality of R.

However, there are graphic packages associated with `ggplot2` to produce the same graphics. If the reader wishes to apply the `ggplot2` graphics for time series they are explained and demonstrated at https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_ts.html. Readers are recommended to work with the `ggplot2` alternatives once they have absorbed this chapter. At each occasion of graphics in the code, create the `ggplot2` alternative.

The chapter will limit itself to what the reader needs to know of the background principles and mathematics of time series analytics. If the reader wants greater depth, she is referred to the first four chapters of the text, *Introductory Time Series with R*, by Cowperwait and Metcalfe. The book is written for laypeople rather than specialists in time series analytics.

Section 10.3.2 of the chapter introduces the ARIMA model for forecasting. If the reader wishes to get into the workings, they can read Chapters 6 and 7 of Cowperwait and Metcalfe.

10.1. Working with Time Series

As already mentioned, a time series is much more than a variable plotted over time. This section will begin by establishing what a time series object looks like and its characteristics. Then the attention will be how standard data such as what we can extract from systems, such as a CMMS, are transformed to a time series object. Finally, the section will explain how to work with series, whereas the next section will explain how to conduct analytics with time series.

10.1.1. Time Series as an R Object

We think of time series as an Excel table of two columns. If charted, the variable of interest is the vertical axis and time period is the horizontal axis. In R, the data are what is

```

> AP
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 194 180 170 185 206 206 190 166 146 174
1953 196 196 233 213 200 215 236 236 220 194 174 201
1954 204 188 233 213 200 215 236 236 220 194 174 229
1955 242 233 269 249 236 251 272 272 256 230 204 278
1956 284 277 314 294 281 296 317 317 291 265 239 306
1957 315 301 338 318 305 320 341 341 325 299 273 336
1958 340 318 355 338 325 340 361 361 345 319 293 337
1959 360 342 400 383 370 385 406 406 390 364 338 405
1960 417 391 428 411 398 413 434 434 418 392 366 432

```

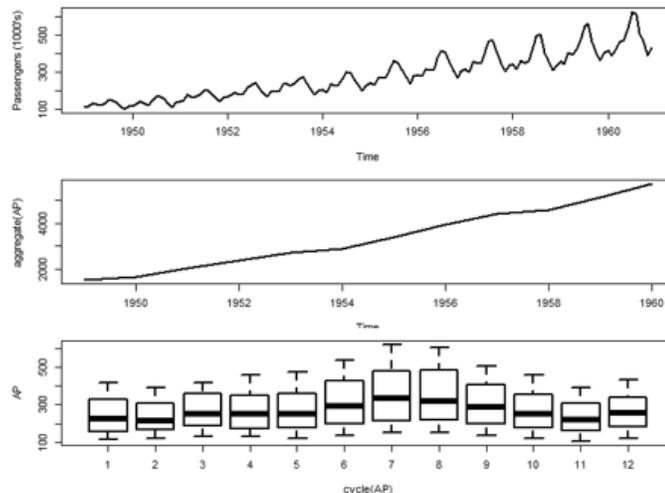
```

> #Characteristics of the object
> class(AP)
[1] "ts"
> start(AP)
[1] 1949  1
> end(AP)
[1] 1960 12
> frequency(AP)
[1] 12

```

Output 10-1. The ts object has labeled rows and intervals as columns.

The top frame is the observed series. It is equivalent to what we typically see with Excel charts. However, notice in the middle panel that we have removed the annual cycle by aggregating the months to each year. The boxplot chart in the bottom panel shows that there is a definitive cycle.



Output 10-2. Observation, trend and annual cycle of the AirPassenger dataset.

Now it is clear, by virtue of the middle panel, what the underlying growth has been. Plateaus emerge that would have been otherwise hidden from us, if in Excel, we had fit a straight line to the plot of the first panel.

We can also see clearly the pattern of the annual cycle. The range of the plot follows the fixed periods, whereas the heavy line is the collective median of the cycles. If we overlaid the cycles with the `lines` function, we would see the annualized strata.

10.1.2. Transforming Data to a TS object

Data from a CMMS and other operating systems do not come to us as a time series object. More typically, we will get our data in format of a data table. The `ts` function is used to convert the standard format to a time series format with the code below.

```
#Import tblMainEmploy and name MainMon.
MaineMon<-read.xlsx(
  "C:\\<path>\\SkillsForCareerSecurity_Datasets.xlsx",
  sheetName="tblMaineEmploy", header=TRUE)

#Format MainMon to time series object
MaineMonTs<- ts(MaineMon$unemploy, start=c(1996,1),freq=12)
MaineMonTs
```

The first block of code imports the data `tblMaineEmploy` and assigns it to the object `MainMon`. If we checked with the `class` function, we would find that our data is a data frame object.

The expression, `ts(MaineMon$unemploy, start=c(1996,1), freq=12)`, in the second block of code makes the transformation. We must specify enough arguments to define the time series.

In this case, we specify that the first record of data was observed at January 1996. The `freq=` argument identifies the breakdown of the year as months. If our data had been quarterly, we would have set frequency as 4. We could have coded an `end=` argument but there is no need because of the frequency argument.

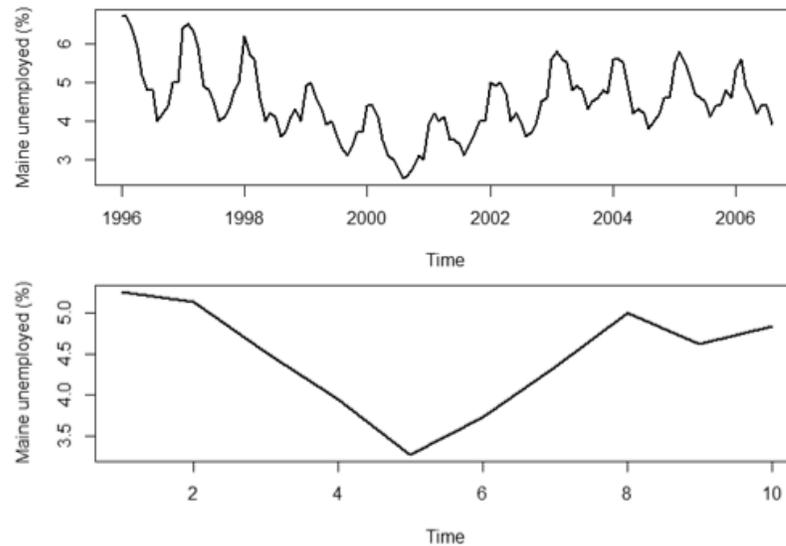
As seen with the AP dataset, we get our series in months. However, what if we want the year as average month? We would use the aggregate function as shown in the code below.

```
#Create aggregate series as annual
MaineAnnTs<- ts(aggregate(MaineMonTs), FUN=mean)
MaineAnnTs

#Chart monthly and annual contrast
layout(1:2)
plot(MaineMonTs, ylab="Maine unemployed (%)")
plot(MaineAnnTs, ylab="Maine unemployed (%)")
```

The expression `ts(aggregate(MaineMonTs))` would total the monthly observations for each year. However, if we want the average month to each year. We make that happen with

the argument `FUN = mean`. Alternatively, we could have divided `ts(aggregate(Mainemonts))` by 12. Output 10-3 presents the results.



Output 10-3. Maine unemployment as monthly observations and average month.

10.1.3. Windows to Time Series

A valuable R function is `window`. It allows us to surgically pull data from the series. The code below is an example.

```
#Computation with time series
Maine.Feb<- window(Mainemonts, start=c(1996,2), freq=T)
Maine.Aug<- window(Mainemonts, start=c(1996,8), freq=T)
(Feb.ratio<- mean(Mainemonts[Maine.Feb])/mean(Mainemonts) )
(Aug.ratio<- mean(Mainemonts[Maine.Aug])/mean(Mainemonts) )
```

What we see in the code for February and August is that starting with 1996 we are extracting the unemployment for the months designated as 2 (Feb) and 8 (Aug). `freq` as `true` causes isolation on the designated months. For example, if we compared what is returned as `Mainemonts[Maine.Feb]` with the `Mainemonts` object, we would find that the former has extracted the column for February from the `Mainemonts` object.

It is possible that we want one or more months or years rather than the slice of across years. We would use the `windows` function to do so with `start` and `end` arguments.

10.2.3. Autocorrelation and Correlograms

An important question of a time series is if a period is influenced by one or more previous periods and by how much. That is an issue because we need to know if, for example, a KPI, is somewhat reflective of previous periods rather than purely reflective of the reported period. This is called autocorrelation.

Chapter 4 explained correlation. Correlation is a measure of how similarly two variables move from their mean. The same general formula applies for autocorrelation. However, for time series, the variables are taken from the single series. Each set of variables is representative of a lag between observations. Accordingly, the respective two variables to the computation are:

$$\begin{aligned} \text{Variable1} &= x_i \dots x_n \text{ and,} \\ \text{Variable2} &= x_{i-1} \dots x_{n-1} \end{aligned}$$

For this case, the pair are subjected to the modified correlation calculation. Variable1 and Variable2 are the pair evaluated for correlation. If the correlation is high, we have autocorrelation.

Whereas the above set of pairs depicted a lag of one, pairs of variables are formed for progressively more distant lag periods. For the two-period lag case, the previous notation becomes:

$$\begin{aligned} \text{Variable1} &= x_i \dots x_n \text{ and} \\ \text{Variable2} &= x_{i-2} \dots x_{n-2} \end{aligned}$$

The correlation calculation is made for sets of two lag periods and then for all possible sets of lag.

The graphic to inspect for autocorrelation is called a correlogram. The code to follow returns the graphic to assess for autocorrelation. It begins with loading the R dataset, AirPassengers.

```
#Load dataset AirPassengers
data(AirPassengers)
AP<- AirPassengers

#Correlogram of observations
acf(AP, lwd=2)
```

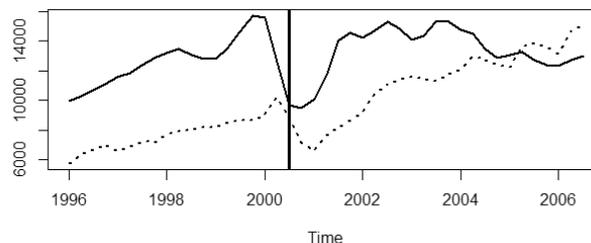
be reported back as a tighter variation than true and the opposite for negative autocorrelation. At times, spread looms large in conclusions and decisions, thus, the unrecognized autocorrelation can be harmful.

10.2.4. Lead-Lag Variables

With times series objects we can search for lead and lag relationships between variables. An example may be the lead and lag relationships between the many KPIs to a maintenance operation.

Of course, we must create a table of the KPI as periodically calculated to be a series. With the methods of Chapter 3, we can build an appropriate super table and then convert them to time series objects in R. As we saw in Section 10.1, the KPIs can be aligned even if the datasets are of periods that do not match end-to-end.

Output 10-14 depicts a quarterly series for which the solid line is construction approvals and the broken line is construction activity. The chart spans 11 years. As we would think, it appears by inspection that approvals are the lead indicator to activity. They have similar patterns, just not at the same time. We can use the insight to be suitably positioned for changing outlooks.



Output 10-14. Time series for approvals and activity by quarters.

We will now go through the code to create the figure and then on to the analytics for lead-lag. We begin by pulling in the `tblApprvBuild` dataset with the `read.xlsx` function, assign it as object `Build` and inspect the first some rows of the object with the `head` function.

```
#Load data
Build<- read.xlsx(
  "C:\\<path>\\SkillsForCareerSecurity_Datasets.xlsx",
  sheetName="tblApprvBuild", header=TRUE)
head(Build)
```

The code below shows a useful technique. It is to place the two series in the same chart. In the code, we first convert the variables, `Approvals` and `Activity`, in the dataset to

300 | Chapter 10

be time series objects. The data is in quarters, therefore, `freq=4`. The `abline` function places the vertical comparison line in the chart.

```
#Plot ts objects in same chart
layout(1:1)
App.ts<- ts(Build$Approvals, start=c(1996,1), freq=4)
Act.ts<- ts(Build$Activity, start=c(1996,1), freq=4)
ts.plot(App.ts, Act.ts, lty=c(1,3), lwd=2)
abline(v=2000.5, lwd=3)
```

The `ts.plot` function combines them in a single chart as shown in Output 10-14. Although there are only two series in the example, we could have included additional ones and set up our code to seek lead-lag combinations among them.

The example is conveniently of two series with the same axis units and practicably close in range. What if we had series such that the KPIs entail different measures such as hours per job and count? That would be weird. It is also a problem if one variable is vastly different in magnitude than the other.

The solution is to convert the multiple series to an index such that all measure against an axis from 0 to 1. The index can be created in Access and included in the imported datasets. In Access, we develop a calculated variable with equation 10-9.

$$\text{Index} = (\text{Min} + \text{Observation}) / (\text{Max} - \text{Min}) \quad (10-9)$$

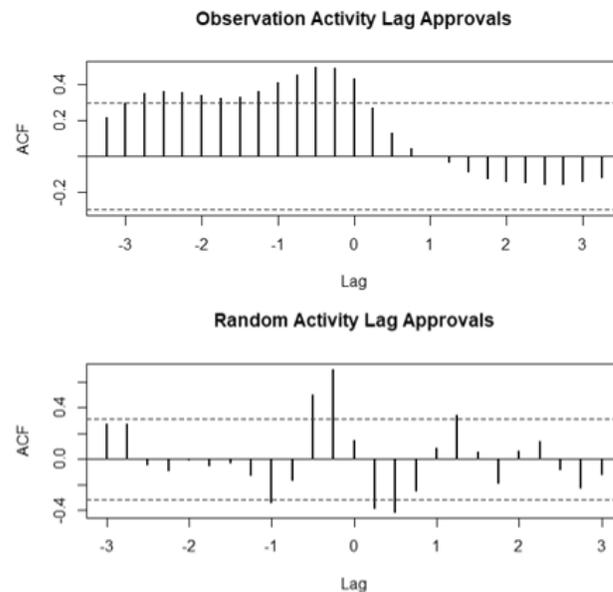
It is left to the reader to use the skills presented in Chapters 3 and 9 to create the variable. Hint: recall that `Max` and `Min` are options to aggregation.

Another conceivable possibility is that the variables have different frequency such as year, month and quarter. Once again, we can prepare the dataset in Access before pulling it into the R session. We need to aggregate the involved time series on a unit of time.

We would use the aggregation functionality of Access to make all time series match with respect to units of period. That too is left to the skills of the readers by virtue of what was demonstrated in Chapters 3 and 9.

The above paragraphs suggest the use of Access. Of course, the tasks can be done in R. Using Access, rather than R, is only suggested as the option most of us will be most comfortable and fluent with.

We seek lead-lag on the random component of the series rather than the series. This is because trend and cycle could dominate the perspective and distort the truth of relationship. The code below returns the random series for approvals and activities.



Output 10-15. Cross correlation between approvals and applications.

10.2.5. Seven-Day Cycles

The inventors of the annual calendar did so without much thought for our needs. The months are not exactly four weeks and the days of a month and year are not divisible by seven. Therefore, working with time series is problematic when we wish to show patterns on weeks within months and days within weeks. At least all weeks start on the same day.

We can work around the issue of irregularity for weeks. We may want to get a quantified sense of a seven-day pattern for some operational variable. The code shows how, beginning with downloading a dataset to the variable of interest to us. The representative dataset is for only three weeks, but the method can apply to any number of weeks.

```
#Import dataset
SevenDay<- read.xlsx(
  "C:\\<path>\\SkillsForCareerSecurity_Datasets.xlsx",
  sheetName="tblwkCycle", header=TRUE)
head(SevenDay)
```

The next block of data transforms the dataset to a time series. Notice the differences from what has been presented in previous sections. Rather than start with a date year, we start with "1" as the first week. Then we specify frequency as the days in a week.

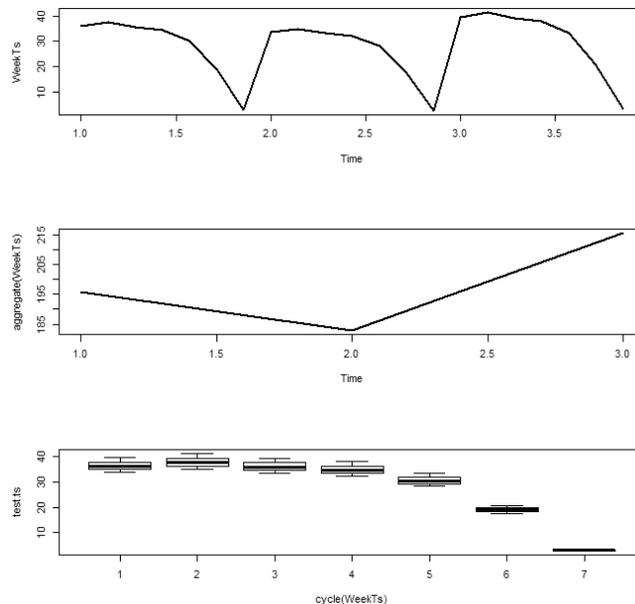
```
#Transform series variable to ts object
WeekTs<- ts(SevenDay$Series, start=c(1,1), freq=7)
```

Now to see what the weeks look like. The code below allows the three insights shown in Output 10-16.

```
#Pattern for week
layout(1:3)
plot(weekTs, lwd=2)
plot(aggregate(weekTs), lwd=2)
boxplot(test.ts~cycle(weekTs))
```

In the top panel of the figure are the daily observations for the three weeks. If for example, the variable was number of requests for new work, there is a pattern. The middle panel shows the total (aggregate) new requests for each week.

The third panel shows a weekly pattern. The peak day, Tuesday, is day two. As we may expect, new requests drop sharply in the final two days which are Saturday and Sunday. Of course, we can set up for any day to be the first day. If data were for less than seven days, we can also set up accordingly.



Output 10-16. Observation, week aggregation and seven-day cycle.

If we did not do that, we would get an output with the same seven factors repeated for each week. The code also converts the data as a time series to a numeric variable with the `as.numeric` function.

The final block of code creates the data frame we want and was previewed in Output 10-17. It combines the factorial patterns of both decompositions and gives a day number and name to each factor.

10.3. Forecasting into the Future

So far, we have talked about what is happening within the range of our data. However, we may want to forecast into the future upon the assumption that current conditions will continue. An example is to assume an economy that will place the same level of productive stress on the plant.

We will demonstrate two methods. Holt Winters and ARIMA. Until now we have worked with the `decompose` function. However, it does not allow us to forecast. This is because the function works upon the computation of the earlier explained moving average. It is not a fitting function. Consequently, it does not return parameters and smoothers on which the past can be projected into the future.

Finally, the section will explain the need and means to confirm that what we want to project can be projected. We cannot if a data series is random or random walk. Furthermore, a random walk can look like a series that can be forecasted. Therefore, the section will demonstrate the test for assuring the time series is deterministic as it must be for forecasting.

10.3.1. Holt-Winters

The Holt-Winters model fits parameters for level, slope and cycle to the data series. Its fit can be additive or multiplicative.

To understand the model, we will start with explaining what is called the exponentially weighted moving average (EWMA). The equation is:

$$a_t = \alpha x_t + (1-\alpha)a_{t-1} \quad (10-10)$$

Where:

x_t = Observation at time t .

a_t = Exponentially weighted moving average at time t as an estimate of x_t .

α = Smoothing factor.

In the smoothing process to compute a_t , α decides how much weight will be placed on the current and the previous period. Therefore, let's now fully extend the concept as the Holt-Winters method. In the model a_t as "level" is computed upon "slope." Slope is the change from one period to the next after accounting for and removing cycle. As a point of reference, "level" in the Holt-Winters model is equivalent in definition to "trend" to the decompose method.

The EWMA model for a_t is generalized beyond α for level. It further seeks smoothing fits for slope (β) and cycle (γ). This can be seen in Equation set 10-11 which is the additive version of a set of three interrelated equations for a series with period p .

$$a_t = \alpha(x_t - s_{t-p}) + (1 - \alpha)(a_{t-1} + b_{t-1}) \quad (10-11)$$

$$b_t = \beta(a_t - a_{t-1}) + (1 - \beta)b_{t-1}$$

$$s_t = \gamma(x_t - a_t) + (1 - \gamma)s_{t-p}$$

Where:

a_t = EWMA for level at time t .

b_t = EWMA for slope at time t .

s_t = EWMA for cycle or season at time t .

p = Periods of the cycle within the fixed period.

α = Smoothing factor for level.

β = Smoothing factor for slope

γ = Smoothing factor for cycle or season.

As mentioned, the model is a set of three equations computed in order from a_t through to s_t . The Holt-Winters function fits the model to the data. It returns α , β and γ as fitting parameters and factor for each period of the cycle.

The associated predictive equation is:

$$\hat{x}_{n+k|n} = a_n + kb_{n+k-p} \quad k \leq p \quad (10-12)$$

Where:

n = Period from which the forecasts is made.

k = Number of time periods forward from n .

The multiplicative case of Holt-Winters is the rearrangement of the equation set for additive. The rearrangements are equation 10-12.

$$a_n = \alpha(x_n/s_{n-p}) + (1 - \alpha)(a_{n-1} + b_{n-1}) \quad (10-13)$$

10.3.3. Test as Deterministic or Stochastic

To this point in the chapter and especially for forecasting, we have assumed that the series, after removing cycle, is deterministic. In other words, we assume the time series can be modeled upon conditions up to current and then projected upon an equation of learned parameters and coefficients.

Some series cannot be fitted because they are random. We need to be able to recognize random series because some can fool us. We may find ourselves forecasting what cannot be forecasted and, worse yet, acting on an imaginary view of the future.

Truly random cases are obvious to us as can be seen in the top panel of Output 10-25. But beware, there is the random walk case in which the series looks as if it is deterministic. In contrast to the random series in Output 10-25, the series in Output 10-26 seems to be going somewhere.

The point is that we need to confirm that our apparently deterministic series is not actually a random walk. This section will explain and demonstrate how to make the determination with tests for autocorrelation.

The code below has become familiar by now. The first block downloads and inspects the dataset. The head function reveals there are two variables; Random and RandomWalk

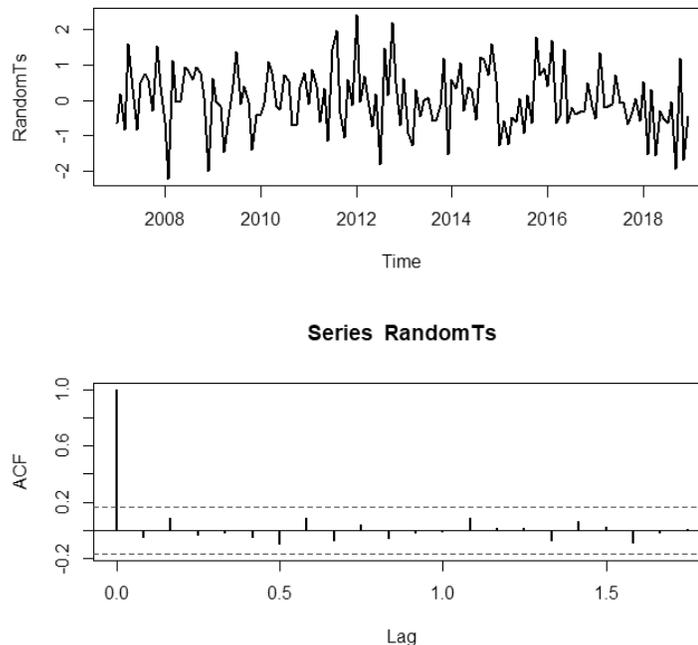
The next two blocks test the two variables. A plot for each series is shown and then its correlogram. For the random walk case we are introducing an additional test, for it will have failed the first test.

```
#Import dataset
StochSeries<- read.xlsx(
  "C:\\<path>\\SkillsForCareerSecurity_Datasets.xlsx",
  sheetName="tblStochSeries", header=TRUE)
head(StochSeries)

#Random series
layout(1:2)
RandomTs<- ts(StochSeries$Random, start = c(2007,1), freq=12)
plot(RandomTs, type="l", lwd=2)
acf(RandomTs, lwd=2)

#Random walk series
layout(1:3)
RandomwalkTs<- ts(StochSeries$Randomwalk, start = c(2007,1), freq=12)
plot(RandomwalkTs, type="l", lwd=2)
acf(RandomwalkTs, lwd=2)
acf(diff(RandomwalkTs), lwd=2)
```

Output 10-25 shows the plot of the random series and the autocorrelation test. In the upper panel we see a series that is obviously random. In the bottom panel we see the autocorrelation test that confirms our grasp of the obvious. At none of the lags does calculated autocorrelation exceed the boundaries.



Output 10-25. Random series and correlogram to test for autocorrelation.

In contrast, as we will see in Output 10-26 that a random walk series fails the test. On inspection of the series, it appears as a long-term increase with some possible cyclicity. More recently, the series plateaued for some time and since then has been on the decline.

We can imagine an organization taking some action on the series. For the protracted decline, the organization may form a team to determine how to intervene to bring about a turnaround.

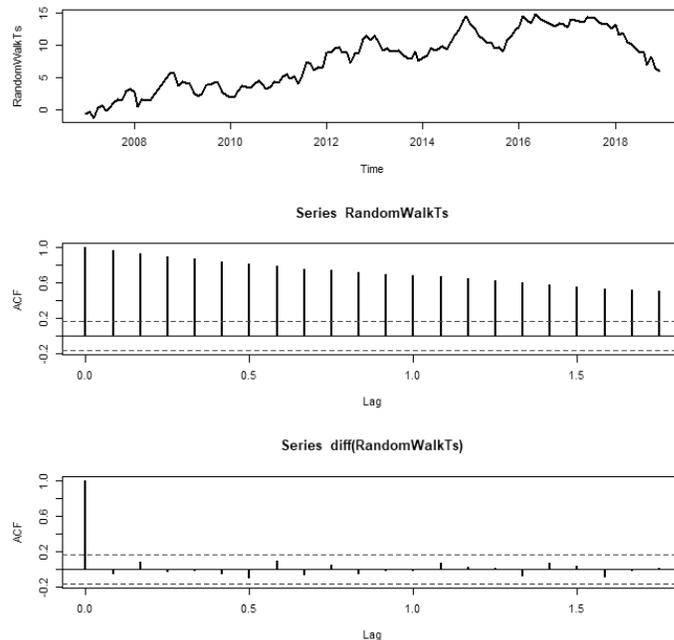
The rub is that a random walk series can fake us into reacting to something we cannot influence or change. Although a series may be a random walk, it takes on the appearance of deterministic. This happens when each period is influenced by the one or more preceding periods.

In the second panel we see that the series shows strong autocorrelation. Because of the shape of the series, we see the pattern in the correlogram. This is the pattern of a

316 | Chapter 10

deterministic series. Recall that a correlogram of the air passenger series showed the same pattern but with season showing as waves in the annual cycle.

What is noteworthy of a random walk is that each period is random. Therefore, if we tested for the difference in autocorrelation between each period, we would reveal the randomness.



Output 10-26. Random walk series failing the autocorrelation test but passing the difference test.

The test is in the third block of the code. In the expression `acf(diff(RandomWalkTs), lwd=2)` we see the `diff` function. The code applies the `acf` function but to the difference between each period.

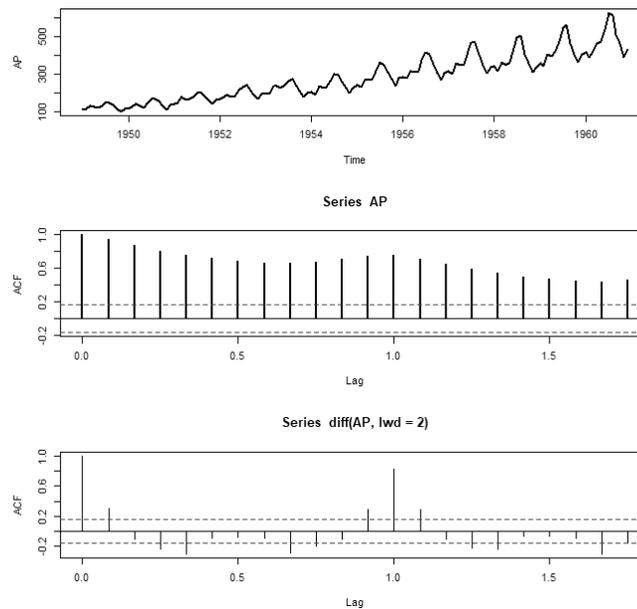
The bottom panel shows the footprint of a random walk. There is no autocorrelation with respect to the difference between periods.

With autocorrelation, we have contrasted the random case with the random walk case. In turn, using the air passenger data, let's contrast between random walk and deterministic. The code below returns the tests shown in the Output 10-27.

```
#Import air passenger data
Data(AirPassengers)
AP<-AirPassengers
```

```
#Test a known deterministic series
layout(1:3)
plot(AP, lwd=2)
acf(AP, lwd=2)
acf(diff(AP, lwd=2))
```

In the top panel of the figure, we see a series that is increasing along with progressively wider cycles. It obviously is not pure random, but is it a random walk?



Output 10-27. Plot and test of air passengers as deterministic series.

The correlogram of the middle panel looks much like the random walk of Output 10-26. The difference is the presence of seasons, but both show the pattern of an increasing series.

However, when we construct the correlogram on the difference between periods, the bottom panels of Outputs 10-26 and 10-27 are markedly different. For the air passenger data, what is returned by `acf(diff(AP))` shows autocorrelation, whereas the same test for the random walk series shows none.

There is a final check to make of a random walk. Does it have a drift? For example, a stock's price at each day's closing may be a random walk. However, we hope that it has some drift in the right direction buried in its walk. The code below applies the test for drift to the random walk series that was previously validated as such.

318 | Chapter 10

```
#Check for drift in Random walk
RandomwalkTsDrft<- diff(RandomwalkTs)
mean(diff(RandomwalkTsDrft))
mean(diff(RandomwalkTsDrft))+
  c(-1.96, 1.96)*sd(diff(RandomwalkTsDrft))/
  sqrt(length(RandomwalkTsDrft))
```

The first line of code assigns the `diff` function of the series to the object `RandomwalkTsDrft`. The second computes a mean for the object: `-0.00456`. Here the mean indicates that there is a slight negative drift to the series. The final line computes the upper and lower 95 percent confidence intervals: `-0.214` and `0.205`.

We can conclude from the confidence limits that there is no significant drift. This is because the signs change, meaning zero drift is possible. If they had not switched, we could have concluded that there is a drift equal to the mean of the differences.

Bibliography

Cowpertwait, Paul and Metcalfe, Andrew. *Introductory Time Series with R*, Springer. 2009.

Cran.r, Vignette for `ggplot2` in time series. https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_ts.html.

Chapter 11

Budget-Based Schedule and Craft Capacity

Of all maintenance operation procedures, planning and scheduling get the most attention and are the most widely adopted of all “best practices.” The rationale for planning and scheduling is indisputable. They are additionally enticing as practices to adopt because the methods are explicit, and they look good and feel good.

However, the procedures have historically been flawed by disregarding a fundamental principle of industrial engineering. They have not been practiced as top down from a workload-based budget that quantified how much work must be done each month to sustain the plant’s productive capacity and asset value. Consequently, the weekly schedule is not consciously loaded with a sustaining workload. It can only follow that craft capacity has not been fitted to deliver a sustaining workload.

Chapter 9 explained and demonstrated how to build workload-based budgets and conduct dual dimensional variance analysis upon them. For reasons explained, workload-based budgeting is a new development to the landscape of asset management.

We should now map onto the landscape the planning and scheduling procedures as they should be downstream from a workload-based budget. That is the purpose of this chapter.

11.1. Scheduling Framed Top-Down

In a production enterprise, here is what happens upstream to maintenance planning and scheduling. The firm determines the market in the coming year and their share of it. They convert the forecast to an aggregate production plan for the year and each month. Upon the aggregate and monthly production plans, the organization determines and prepares every direct and indirect enterprise activity and associated resources that must be in place to profitably win and defend their market share.

This is where the maintenance operation enters the picture. It must determine and prepare its activities, processes and resources to deliver the maintenance workload necessary to sustain the plant’s ability to meet the production plans and to sustain the plant’s value as a production asset. Furthermore, rather than victory at any cost, the maintenance operation carries the burden of delivering the workload with optimally sized craft capacity.

With respect to the goals, the parallel to the aggregate production plan is the workload-based maintenance budget. The budget process was explained and demonstrated in Chapter 9. From the workload-based budget, the plant will quantify a baseline for a weekly mixture and count of jobs. Throughout the year, each weekly schedule is molded in accordance with the baseline.

Each week, planned ready work is assigned to the schedule. For each mixture group, jobs are assigned according to sustaining count and in accordance with plant-specific policies for priority and in-out (e.g. first in, first out), etc.

There has been a standing philosophical debate for loading the week's schedule. Some advocate that greater or less than true expected workload be loaded. Others advocate loading upon 100 percent of the expected achievable workload.

Just as the budget quantifies annual and monthly sustaining workload, so must scheduling quantify the weekly sustaining workload. The purpose settles the argument. Schedule is loaded at 100 percent because it is the baseline of how much work must be done each week as the cost of winning and defending market share and protecting asset value.

When there are day breakers, each takes a slot on the week's schedule, bumping out a scheduled job. Whether the jobs were scheduled or breakers is secondary to management. They are not purist.

To them, the true overarching measure of a successful week is to accomplish a baseline mixture and count of work. That would be senior management's definition of success and, thus, the ultimate value to them of the maintenance operation.

The baseline weekly workload, as smoothed mixture and count, are the flywheel with backlog as the wildcard, growing and shrinking on any given day. Furthermore, scheduling in response to backlog makes operations reactive. The issues, analytics and design of backlog are the subject of Chapters 12.

Now let's contrast scheduling pulled along by the sustaining workload to the currently predominate practices in maintenance operations. Rather than loaded on smoothed sustaining workload, the week's schedule is largely loaded according to craft hours available to do work. The loading is also influenced by the number of jobs ready in the backlog. Which jobs are scheduled from the backlog reflect policy upon priority without regard for each mixture group in the schedule. In total, whether scheduling from week to week is on beam of sustaining workload will be largely left to happenstance rather than intention.

However, the consequential contrasts to senior management are the outcomes they do not want the maintenance operation to leave to happenstance. There is the link from

market conditions and share to the workload to assure share is not lost. There is the link from asset value to workload to assure asset value does not become overvalued in the balance sheet. And there is the link from cost leadership that wins market share in many industries to craft capacity fitted to the plant’s sustaining workload.

11.2. Budget-Based Schedule Cycle

To repeat, the goals for budget-based scheduling and craft capacity are twofold. First, execute daily a workload to sustain productive capacity to obtainable market share and asset value. Second, execute the workload with a craft capacity fitted to the workload.

The budget process will have established with senior management the sustaining workload. The budget has shown them proof of the cost of maintenance for market share. They have accepted it as a true cost of doing business and have agreed to pay it. Inherent to the cost is the promise to execute the work with a craft force fitted to the workload management has agreed to “purchase.”

Consequently, to senior management, scheduling procedures are the current-time proof that the agreement and promises will be realized. They must see in action the three stages below and the subject of this section.

- Establish baseline schedule and craft capacity.
- Conduct the weekly scheduling and execution.
- Measure and confirm the proficiency of the weekly cycle.

11.2.1. Establish Baseline Schedule and Craft Capacity

Rather than begin here, the weekly scheduling process actually began with the annual budget. What must happen next is flowcharted in Figure 11-1. Each step will be expanded upon.

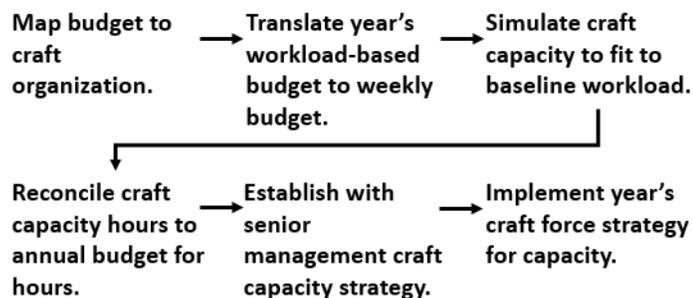


Figure 11-1: The activities to set weekly budget for sustaining workload and craft capacity to do the work.

being hardheaded or ignorant, they are reacting to some strategic condition such as business cycle. In this case, the collaborative decision-making flows from the dollar-limited craft capacity down to achievable workload as a strategic variable in enterprise success.

Just as importantly, management's departure from the proven necessary sustaining workload can be known and recorded in detail. Future budget cycles can know to pick up the deferred workload in the process of the next round of collaborative decision-analysis and decision-making.

Implement craft capacity strategy. It is entirely possible that the budget will vary from full scale in some years. This means that there may be adjustments to craft types and headcounts. The current body may be under or oversized to the final budget.

How the adjustments are made is also a senior management decision. It is a decision that must be made with an eye to many issues such as longer-term availability of crafts, flexibility of crafts, when crafts will be reengaged, etc. There is also the psychological to consider and image as a corporate neighbor.

Regardless of final strategy, it must be implemented. The final step is to form strategy with senior management and implement it.

11.2.2. Conduct Weekly Scheduling and Execution

The previous stage set the baseline to the weekly schedule and craft capacity to execute daily a sustaining workload. The two baselines are pre-measures. The weekly baseline is the pre-measure to the week's schedule. The baseline for crafts is the pre-measure for hours to be made available to the week's schedule.

The important thing to note is that the weekly process is different than traditionally taken. This is because all is based on sustaining plant market share and value with a craft force necessarily fitted to do so. This departs from the process where current available craft hours are the independent variable rather than sustaining workload.

The process is flowcharted in Figure 11-2. At the top of the figure are the three inputs to the process. Two were set in the previous stage. They are baseline sustaining weekly schedule and craft capacity fitted to it. The third is an appropriate backlog of ready jobs to allow the week's schedule to be loaded 100 percent with scheduled and planned work.

The inventory of ready jobs is a discussion and process in its own right. The inventory must hold the mixture and counts for sustaining workload such that the backlog for each is adequate to enable smooth operations but not retained overlong in the stages from initial work request to becoming ready jobs. The flow of job plans through the stages will be the subject of Chapter 12.

Decision for cumulative over and under run of executed jobs. The goal of the step is to bring each new week back on beam to sustaining market share and asset value. As there is slippage from sustaining workload, we need to be aware so that the plant can tweak the current or subsequent weeks to get back on beam.

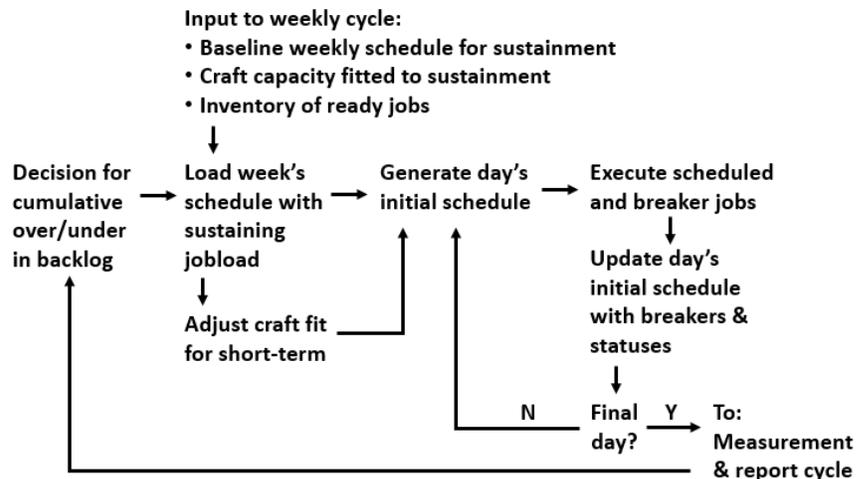


Figure 11-2: Process for week and day scheduling upon the principle of smoothed sustaining workload and fitted craft capacity.

As shown in the figure, the measurement and report stage will track the slippage balance as of the end of the previous schedule week. The measurement is an input to the current week.

The cumulative over-under to sustaining workload is reported by mixture groups with respect to the budget as of year to date. Do not confuse it with the total backlog balance which moves independently of the week's schedule.

Load week's schedule with sustaining workload. The next step in the cycle is to load the schedule. In the flowchart, this is the junction at which the baseline pre-measures for workload and capacity enter the process. At this step, the schedule is 100 percent loaded with jobs that are planned. If any loaded jobs are not planned, there is a failure in the stages that stock the inventory of jobs ready to go.

Historically, the loading practice has been to place orders on the schedule by a scheme of priorities. Only one rule of the scheme holds for loading with respect to groups of mixture and count.

It is the rule for work that must be selected first to load. These include carried over work and work for which there is a justifiable reason to break the rule of first in, first out to the inventory of ready jobs.

11.2.3. Datasets, Measures and Reports Stage

Throughout the week there is a process of updating and maintaining data as subordinate and super tables, and the measures and reports upon the data. All are disseminated through the accessibility of all role holders to the scheduling and overall maintenance operation.

Figure 11-3 is a flowchart of the stage. It begins with designing and building the tables and insight deliverables for the first time. Once built, tables and deliverables are a daily task. It is conceivable that our experience with the daily will send us back to the development tasks of the stage to upgrade standing insight deliverables and create new ones.

In the flowchart, we can see that the outcomes of the week’s schedule process flow to the task to maintain the tables to the schedule and execution stage. In turn, we can see that the task to generate the insight may send us back to the baseline stage of the scheduling function.

Formulate subordinate and super tables. When an operation is data-driven, there will be subordinate and super tables inserted into its processes. They are essential but do not exist in our operational systems and never will.

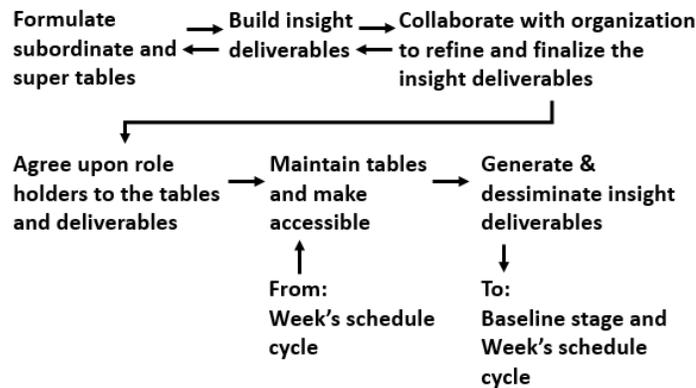


Figure 11-3: Steps to the stage to measure and report to workload scheduling and craft capacity.

The task is to design the tables and make them operational. Section 11.3 defines and scopes the collection of tables to be built and explains how to build them.

The task may be revisited. When a plant begins to schedule upon sustaining workload, for a few weeks or months the builder may be called upon to pop the hood and soup up the engine. At some time, the operation will settle in and upgrade much less frequently.

where located. Role holders across the operation that draw upon the insight do so by call-up. To them are returned the latest editions of the insight deliverables.

The tasks in the stage are a running verification of the baseline workload and craft capacity that were established by analysis in the first stage of the short-, middle-, and long-term planning and scheduling cycles. Accordingly, the depiction of return to the baseline setting stage is shown as ongoing.

11.3. Tables to the Schedule Cycle

The previous section described what must be done as process if the plant is to schedule and execute work orders in accordance with principles of conducting the workload to sustain plant productive capacity and asset value, and with fitted craft capacity. This section will identify and describe the tables upon which the vision is realized.

For each table, the section will describe how to build it in MS Access. However, what is described transfers to other software such as Tableau in which tables can also be built.

Chapters 3 and 9 have explained and demonstrated the full range of techniques for building the tables as described in Access. Therefore, this section will not provide the code for every table as it did in the two chapters. The chapters were written to bring the readers to be capable of building the described tables.

The description of the tables for scheduling are generalized because all maintenance operations are situationally specific. However, we can and will identify and describe the purpose and construction of all necessary tables.

Accordingly, the section will present representative tables. It is left to the readers to parallel what is presented here to the plants for which they are developing the tables.

The data to the chapter are available for download at the webpage given in the preface. They are contained in the Excel file, SkillsForCareerSecurity_Datasets.xlsx, as the worksheets tblDaySchedule and tblCraftHrs.

11.3.1. System of Tables to Build and Maintain

The source tables are taken from the CMMS, scheduling and hours capture systems to the maintenance operation. They are normal to the weekly and daily scheduling, and execution cycles. Between them they capture every associated record to the insight we seek.

The tables are as follows:

- Weekly schedule.
- Initial day schedule with breakers and categorizing classifications added.
- Craft hours from hours capture system.

- Craft classification translation table.
- Super table of the day's schedule and breakers joined with craft hours.

The tables are built daily and appended to the week's table. Ultimately, each week is appended to a continuous table of history. Therefore, history is all previous weeks captured as a single table.

The tables are initially built by analysts and can then be kept current and available to users by an administrator. Users in turn, engage them in their own roles.

The reality of modern systems and data is that we no longer need to look backward from pre-specified insight deliverables to what data to collect. Instead, in our tables we are free to hold every bit of data that is imaginably associated with workflow, planning, scheduling, execution and measurement. This is because there are no savings to be had from limiting the data in our tables. Nor are there practicable constraints or costs to making all data available to everyone without first questioning cost-benefit.

11.3.2. Weekly Schedule as Table

The week's initially loaded schedule is the workload budgeted to sustain the plant production plan and asset value. Typically, plants make the week's budget available as a standard output of their scheduling software. For most, the content is structured as a data table rather than some unworkable spreadsheet as explained in Section 8.1.

The variables we need as a table and typical to the standard output of scheduling software are as follows:

- Date of schedule week.
- Order number.
- Order description.
- Task number.
- Task description.
- Craft classification.
- Hours per craft classification.
- Headcount per craft classification.
- Duration of task.

There is an issue to verify. Can we extract the initial schedules from the schedule system as the history of all past weeks rather than as each week in its own table? If not, an administrator will need to append each week to the history of weeks; a simple task with the append query.

throughout the chapter so to focus on method. For their plant, readers will add them to the explanations.

Now to continue the explanation. First, notice the three classifying variables that tell us the nature and fate of each job at the end of the day. They are classifications as scheduled or not, planned or not and status at the end of the day. As it will be seen, they play variously in every imaginable measure and insight deliverable.

DateWeek	DateDayS	OrderS	TaskS
2/3/2020	2/5/2020	6001	10 Me
2/3/2020	2/5/2020	6001	11 Ele
2/3/2020	2/5/2020	6001	12 Ins
2/3/2020	2/5/2020	6003	11 Wr

CraftS	CountCrftPI	HrsPerCrftPI	DurationPI	HrsTotalPI
10 Mech		2	2	4 S
11 Elect		2	2	4 S
12 Insulate		2	1	2 S
11 Welder		0	0	0 B

SchedFlag	Status	Planned	CrftId	WrkID
4 S	3C	P	60011043866Mech	43866600110
4 S	2S	P	60011143866Elect	43866600111
2 S	1H	P	60011243866Insulate	43866600112
0 BD	3C	NP	60031143866Welder	43866600311

Figure 11-4: Table of day's scheduled work updated with day's actual activities.

The SchedFlag variable identifies the job as initially scheduled or have broken into the day. Only two classifications are shown by the example, but the developer can create any set of classifications as part of their schedule process. For example, "BU" may designate a breaker order as urgent.

The big promise of planning and scheduling is that a day's work is most productive when structured. We already have a code for scheduled or breaker work. We should want one for designating which orders are planned and unplanned. That is the why of the "Planned" categorical variable.

With an if-then expression, we can cause the scheduled jobs variable to automatically populate; assuming all scheduled jobs are planned jobs. However, we should not assume the reverse; that breaker jobs are never planned jobs. For breakers, we would look to the administrator to enter the correct classification during or at the end of day.

The variable, "Status," is the final primary classification. Notice that it designates whether the job was started (2S), completed (3C) or left untouched (2H). The numeric first character allows the statuses to be ordered by sequence; something the developer

discovered a need for while designing the report of measures. An alternative is to assign classification that have alphabetical order.

The final two variables in the figure, CrftId and WrkId, are unique identifiers with two insight deliverables in mind. At the source of the table, Excel or Access, they are made automatic to the records. If not created upstream, downstream is never too late.

The first, CrftId, is a concatenation of order number, task number, day's date and craft. The second, WrkId, is the concatenation of day's date, order number and task number. They will be mirrored in other tables.

Just previously, it was noted that our grouping variables are more extensive than demonstrated. Accordingly, in real life, the concatenation would include all grouping variables.

As mentioned, the administrator's updating actions are done daily. As implied, the final table to the day is appended to a table of history. The append action would be done by the administrator.

11.3.4. Craft Hours Table

Chapter 8 introduced the idea of data current at capture. Section 8.2 used the situation of timesheets to explain the issues. Now the ramifications are dramatically apparent with respect to the measurement of sustaining workload and craft capacity.

The point of "entirety" was that we need our hours data now, not days from now when fully vetted and distributed to work orders in the CMMS. Our objective is to have feedback as soon as the day's hours are fully recorded.

The method in Section 8.2 was to structure timesheets as a table format. Consequently, at the moment of the last recorded hour to the day's work, we can extend the day's schedule horizontally to span craft actual hours and other details to the hours.

Of course, some plants may have time entry systems rather than Excel timesheets processed by payroll clerks. In that case, the data must be made accessible on demand. If not the case, the plant's data engineer can make it so.

The variables to the craft hours table are as expected and as follows:

- Date incurred.
- Personnel number.
- Craft category and grade.
- Order number.
- Task number.
- Hours.
- Hour type or code.

- Shift.
- Unique identifiers.

Figure 11-5 show the table but without order and task descriptions. Notice the mirror identifier variables to the table of work scheduled and conducted. They are named differently but are identical concatenations.

DateT	OrderT	TaskT	CraftT	EmpNo
2/5/2020	6001	10 Mech		1001
2/5/2020	6001	10 Mech		1034
2/5/2020	6001	10 Helper		1004
2/5/2020	6001	11 Elect		1008
2/5/2020	6003	11 Welder		1019

HrsT	Shift	HrCode	CraftIDTime	WrkidCr
11	2 D	S	60011043866Mech	43866600110
14	3 D	S	60011043866Mech	43866600110
14	1 D	S	60011043866Helper	43866600110
18	2 D	S	60011143866Elect	43866600111
.9	1 D	S	60031143866Welder	43866600311

Figure 11-5: Table of detail for all individuals in the craft pool accountable for a body of work.

We want a master table of all hours as far back as we have captured daily schedules in a table of history. Therefore, the administrator will append each day's hours to the history version of the figure.

Sometimes hours are corrected after initially recorded. If so, we can easily modify the processes to generate the table. We would retroactively pull final approved hours into history; replacing the initially captured hours.

Although not shown, we would collect all timesheet hours to the crafts without regard for direct and indirect hours. This leaves in the data table the wherewithal to study the proportions and classifications of direct and indirect hours embedded in the craft force. It is also an opportunity to study the subsets to indirect hours. In fact, the variables are all that are imaginable to establish the gross craft force necessary for the net craft capacity to the sustaining workload after allowing for indirect time.

11.3.5. Craft Classification Translation Table

Section 3.3.2 explained and demonstrated the idea of “translation” tables. One frequent use is for categories to a common variable in tables from different sources that do not have exactly matching classifications or formats.

Having explained translation tables, they will not be demonstrated in the explanation to come of the super table to the weekly schedule and execution. The variables from each source table have matching variables. This section has been an opportunity to explain how to work around the problem if it were the case.

11.3.6. Super Table to Measurement

Top notch maintenance operations require a top-notch ability for analytics and insight around scheduling, planning and craft capacity. They, in turn, require a platform super table.

Let's look at a representative such super table. It is shown in Output 11-1. From there we will track back to how the previous tables and queries converge to build the super table.

Date_Week	Day_Date	Order_No	Task_No	Craft	C
2/3/2020	2/5/2020	6001		10 Mech	
2/3/2020	2/5/2020	6001		10 Mech	
2/3/2020	2/5/2020	6001		10 Helper	
2/3/2020	2/5/2020	6001		11 Elect	
2/3/2020	2/5/2020	6001		12 Insulate	
2/3/2020	2/5/2020	6003		11 Welder	

Count_Plan	HoursPerCraf	Duration_Pla	Hours_Total	
2	2	2	4	S
2	2	2	4	S
0	0	0	0	S
2	2	2	4	S
2	1	2	2	S
0	0	0	0	Bl

SchedFlag	Planned	Status	
4 S	P	3C	M
4 S	P	3C	M
0 S	P	3C	H
4 S	P	2S	El
2 S	P	1H	
0 BD	NP	3C	W

CraftType	Employ_No	HoursCharge	Shift	Hours_Type
Mech	1034	3 D	S	
Mech	1001	2 D	S	
Helper	1004	1 D	S	
Elect	1008	2 D	S	
Welder	1019	1 D	S	

Output 11-1: Super table of day's schedule, job plans and craft actual hours.

In the super table, we see the dates to the schedule week and day. Imagine a continuous super table of all weeks up to the present. The ramifications for insight are the subject section. In this case, we are viewing the super table filtered on the week of February 3, 2020 and Wednesday of the week.

The variables from Order_No through to Hours_Total come with the day schedule. However, notice the zeros occurring in the planned headcounts and labor hour details. They were generated within the query to the super table to replace empty cells.

They indicate two possibilities. The first is that an unplanned craft was engaged in the work. Helper for order 6001, task 10 is such a case. The second is that a breaker task was allowed. Order 6003, Task 11 is a breaker that engaged a welder.

Next are three flag variables. They are scheduled or other, planned or not, and status of execution at the end of the day. They represent a range of classifications. We foresee them as needed to ultimately generate the measures and insight deliverables we have in mind for the week's schedule and execution cycle.

The variables Craft_Type through Hours_Type are the standard details to payroll and the distribution of hours to orders. If direct hours are incurred, they will appear here as an extension of the job plan details.

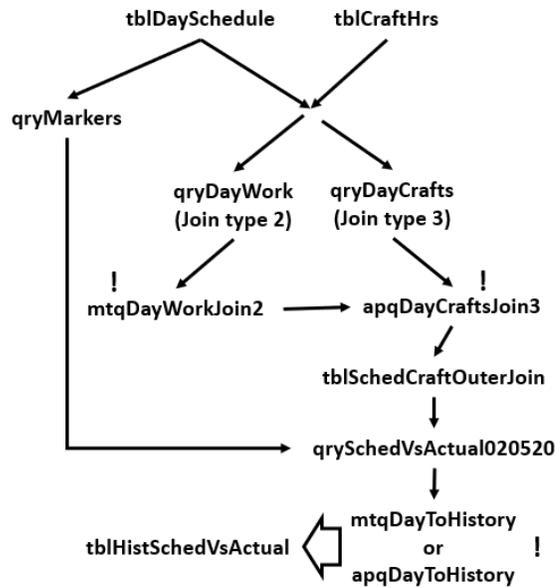
Let's make a note here. In real life the table of craft timesheet would include direct and indirect hours. To build the super table we would insert a criteria for hours type such that only direct hours appear in the super table. Not to get back on message.

The empty cells related to timesheet for order 6001, task 12 reveal a scheduled job that was not started. As already spotted from the schedule and plan side of the super table, Order 6003, Task 11 is a breaker as indicated by welder hours, goose eggs in the plan variables and schedule flag of the classification "BD."

The shown super table is an administrator's daily task to keep current. The task ends when the table is appended to the table of all previous weeks. The daily procedure is shown in Figure 11-6.

The process begins with the tables tblDaySchedule and tblCraftHrs. They were the subject of Figures 11-4 and 11-5. They converge to return qryDayWork and qryDayCrafts. In both, they are joined on the matching concatenated identifier variable to each. Actually, the variables to both queries are identical. The difference is the join type between them as noted in the flowchart.

Consequently, the query, qryDayWork, will list all work to the day; scheduled and breaker. The completed and not-completed work is also included.



Notes:

! = Query must be run rather than automatic.

Run mtqDayWorkJoin2 before apqDayCraftsJoin3

Figure 11-6: Flowchart to build the super table.

However, we still need to find any case of work done by a craft that was not included in the job plan. In the example, a helper craft took part in Order 6001, Task 10. That is why the Type 3 join, a right join, is called for in the qryDayCrafts.

Both queries upon the different join types will return records that appear in both. Therefore, we need to isolate on the differences. This is done in qryDayCrafts by setting the criteria, “Is Null,” to a variable that will be empty in a case of exceptions. In the example, the DateWeek variable was used but could have been another. The result is that the records common to both queries will be omitted from the query.

The goal is to arrive at the day’s super table, qrySchedVsActual020520. To get there, we need to combine the two queries vertically rather than horizontally.

There is a two-step dance. First, we use a Make-Table query, mtqDayWorkJoin2, to return tblSchedCraftOuterJoin. Next, we run the Append query, apqCraftJoin3, to append the data of qryDayCrafts in tblSchedCraftOuterJoin. The resulting table is the input to qrySchedVsActual020520.

The make-table and append queries are what are called “action” queries. This means that we must command that they run, whereas other types run automatically. The Access icon shown in the flowchart is clicked to command the run.

To explain it further, let's consider the flowcharts of tables and queries we have seen throughout the book that did not include run queries. In those cases, if we called up the final query in the flow, all upstream queries run automatically in the background. If we updated an input table, we would not need to open and run each query along the way to the final query.

For the flowchart, we would need to run each of the action queries. We must also run them in order that they are to take an action. In the flowchart, we would run the make-table query first and then the append query.

The flow can be made automatic with the Visual Basic language of MS Office. That is left to the readers who wish to make it possible.

The power of the super table is that it gives us the essential ability to classify the jobs by schedule type and action. However, we need a query to extract the details from tblDaySchedule and now join them to qrySchedVsActual020520. The developer's trick is to extract them by query from the day's final tblDaySchedule along with the concatenation variable for day's date, job number and task number. The extraction is named qryMarkers. We could alternatively make the flags appear for the first time in the super table, but it would require more gyrations.

The next step is to polish the query to be the super table shown in Figure 11-6. Rather than explain them, the design grid to the super table is Code 11-1. In the grid, the reader can see how each variable takes its final form.

Code 11-1: qrySchedVsActual020320

Join	Inner: tblSchedCraftOuterJoin and qryMarkers on WrkId		
Field:	Date_Week: IIf([DateWeek] Is Null,#2/3/2020#[DateWeek])	Day_Date: IIf([DateDayS] Is Null,[DateT],[DateDays])	Order_No: IIf([OrderS] Is Null,[OrderT],[OrderS])
Table:			
Sort:			Ascending
Show:	Yes	Yes	Yes
Field:	Task_No: IIf([TaskS] Is Null,[TaskT],[TaskS])	Craft: IIf([CraftS] Is Null,[CraftT],[CraftS])	Count_Plan: IIf([CountCrftPl] Is Null,0,[CountCrftPl])
Table:			
Sort:	Ascending	Descending	
Show:	Yes	Yes	Yes

340 | Chapter 11

Field:	HoursPerCraft: IIf([HrsPerCrftPl] Is Null,0,[HrsPerCrftPl])	Duration_Plan: IIf([DurationPl] Is Null,0,[DurationPl])	Hours_Total: IIf([HrsTotalPl] Is Null,0,[HrsTotalPl])
Table:			
Sort:			
Show:	Yes	Yes	Yes
Field:	SchedFlag	Planned	Status
Table:	qryMarkers	qryMarkers	qryMarkers
Sort:			
Show:	Yes	Yes	Yes
Field:	CraftType: CraftT	Employ No: EmpNo	HoursCharged: HrsT
Table:	tblSchedCraftOuterJoin	tblSchedCraftOuterJoin	tblSchedCraftOuterJoin
Sort:			
Show:	Yes	Yes	Yes
Field:	Shift	Hours Type: HrCode	
Table:	tblSchedCraftOuterJoin	tblSchedCraftOuterJoin	
Sort:			
Show:	Yes	Yes	

Although pretty much akin to what we are accustomed to with Excel, there are noteworthy actions taken for some of the variables. They deal with issues that emanate from the fact that the table is representative of creating a full outer join (Section 3.3.4).

One group of four variables entail null cells for the week and day dates. They are caused by the nature of work by crafts which were not identified by the job plan. The variables are Date_Week, Date_Day, Order_No and Task_No. For them the expressions in Field of the grid replace empty cells with the appropriate data.

Another cluster of variables are associated with the job plans. They are the variables Count_Plan, HoursPerCraft, Duration_Plan and HoursTotal. These are the cases to which a craft type was not foreseen in the plan. For them, the records need to include zero rather than be an empty cell. That is because computations with a null often return a failure in the code.

A huge amount of power resides in a plant's history tables. In this case, the history of every day is captured in a continuous dataset. That is the final step to the flowchart.

Upon the first day of building the super table, we would create a table such as tblHistSchedVsActual with a make-table query. For every report day, thereafter, we would use an append query to update the history table.

We now have our super table. From it we can do almost anything. The next section will demonstrate the super table as the platform to a range of measurements.

11.4. Measures and Methods

The previous sections presented a representative super table and the tables that feed it. The whole purpose is to arrive at the ability to measure the many aspects of planning, scheduling, capacity and execution of the plant's sustaining workload.

There are three dimensions of performance to measure.

- Sustainment of productive capacity and asset value.
- Structured execution as the degree that each day's work is planned and scheduled.
- Accuracy of job plans.

The subsections to follow will introduce measures to each dimension of performance. They will also describe the means to extract the measures from the super table qrySchedVsActual020520.

11.4.1. Confirm Sustainment

We need to assure ourselves and enterprise management that the aggregate plan for production will be met. We also want to be assured that the year will end with a balance sheet that reflects the true value of capital assets.

The assurance is based on the following two measurements:

- Actual work compared to week's initially loaded sustaining workload.
- Balance of sustaining workload in backlog.

The measures suggest the reason that the week's schedule be 100 percent loaded. We want to compare all work done against how much sustaining workload must have been done. We will want to do that each week and track and monitor cumulative variance from the balance.

Recall from Chapter 9 that one dimension of dual dimensional variance analysis reports variance due to work completed. In it, we verify that the month was on beam for sustaining the plant. The above measures are short-term verification timely to ending the month on beam.

Actual work compared to week's initially loaded work. Figure 11-7 is a flowchart of what must be built in Access and Excel for the first measure. It is also the step-off to the second measurement of sustainment.

There are two sides that converge to generate the measures. Along the left side of the flowchart is the week's initial 100 percent loaded schedule. It is the baseline for how much work must be done.

With it we want to create an aggregation query of counts for orders to every work-type mixture group. As mentioned repeatedly, the groups will reflect more than mixture.

Whatever the grouping, we need to place a concatenated identifier variable in the query that will be identical to that planted in the parallel aggregation query. We will probably have already created the concatenated identifiers in the table of week's schedule.

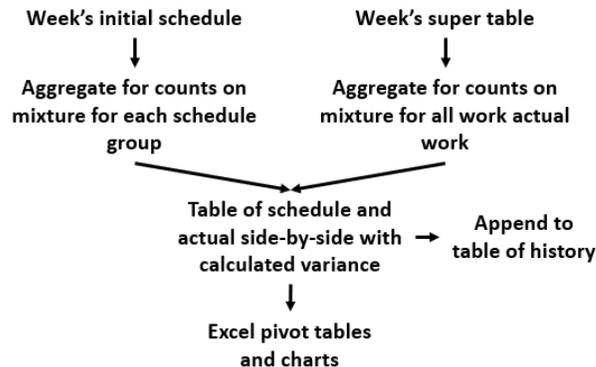


Figure 11-7: Specifications for queries and pivots to measure sustaining against actual work accomplished.

Along the right side of the flowchart, we want a mirror aggregation query of counts to the week's scheduled jobs but built upon actual work started and completed. The input table is the week's super table query.

In the query we will want to remove all scheduled jobs that were not started. To do that, we would set the criteria to the variable to remove the IH case.

Next in the query, we need to remove the jobs that started on one day and were completed in the next day or later. It is also done in the aggregation query. Across multiple days, these cases would appear as duplicate records with respect to their groups.

Recall that the available aggregations included First and Last options. With it, we can retain the duplicate record with the latest status in the sequence from hold to start to complete.

In the query we sort the status variable in ascending order. Recall the statuses have a sequencing numeric index, 1H, 2S, 3C. Then the "Last" aggregation of the status variable will return the last status in the sequence to be returned in the query.

Both trails converge. They are joined on the previously described concatenated variables that cause them to match up on groups. For each group, we now have the counts to scheduled and actual jobs for the review week. It follows in the query we will insert a calculated variable of actual minus scheduled counts. The calculation is the variance.

The variance query can be taken a step farther. A variance can be placed in the criteria row that will return only groups for which there are variances. We could also set upper and lower limits to variance. How was explained in Sections 3.4.4 and 3.4.5 and 9.2.4.

the end queries will come back as measures for periods of time. We can sort and filter them from any direction to see the current week in the context of history.

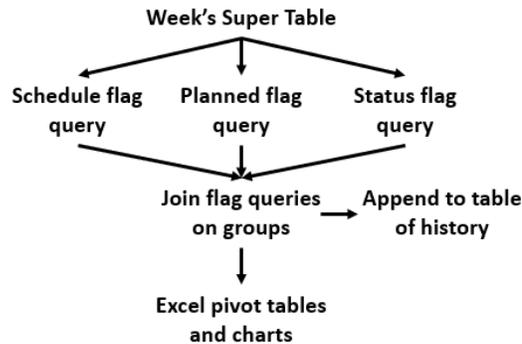


Figure 11-8: Build tables to return the six measures.

Finally, the measures can be explored, presented and disseminated in Excel pivot. We can explore them in Access, but the interactive nature of pivot tables and charts allows us to look at them in many perspectives.

11.4.3. Job Plan Accuracy and Variance

It is not usually the intent that job plans be rigorous and precise. The idea is that a reasonably accurate plan, as to scope and resources, organizes us toward doing the right things in forming the week's schedule, positioning resources and executing work.

It is good to evaluate a week's planned jobs against the actual outcome. While all is still fresh in current events, we want to spot craft omissions, variances from craft count to the scope and variances in average and total hours.

However, "measurement" of job plans is not the primary thrust. More importantly, we are seeking to sharpen the proficiency of the schedule and execution processes. Therefore, the unforeseen and variances to job plans rather than measures are the central point.

We could translate that to measures of plan accuracy. However, the better purpose is to identify the job plans that varied significantly from what happened in the field. Accordingly, we would set upper and lower boundaries upon which is returned a list of orders of interest.

The variances we seek for identifying true outliers are as follows:

- Craft types not included in scheduled plan.
- Count to a planned craft.
- Hours per craft.

- Total planned hours.

Figure 11-9 flowcharts the queries to arrive at a table of variances. The super table comes into play once again. The super table is queried by schedule and timesheet paths. After forming aggregations along each trail, all are joined in a single variance table.

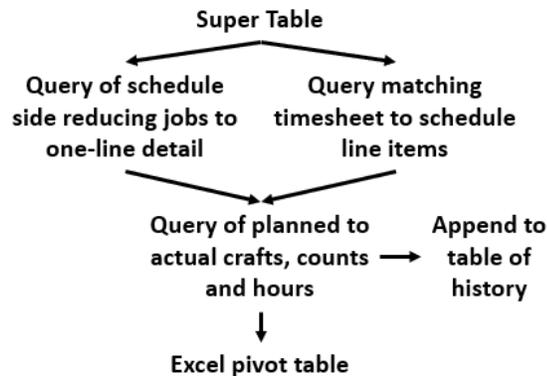


Figure 11-9: Flowchart to return variances to job plans.

The schedule side of the flowchart is simple. The variables to the plan side are pulled into the query from the super table. We also pull in the flags for scheduled, planned and completed and set the criteria for only the orders that were scheduled, planned and completed. We want a single record for each job.

To assure that each plan presents as one line, the query applies aggregations. Each variable is set to Group By in the Total row of the grid. This is a developer's trick to avoid surprises.

The final requirement of the query is to set a unique identifier upon which to horizontally join the query to the timesheet single records. In real life, this will entail the concatenation of all fundamental groups such as day date, cost center, work type as mixture, order and task numbers, etc. They may already exist in the joined queries.

Down the right trail we have the actual hours and crafts to each planned job. Code 11-2 is snipped from the query in Access to show how the aggregations were setup for each variable.

In the figure, we see the groups on date, order and task number and the flags. The crafts are grouped on the timesheet-side query. The flag variables are set with the same criteria as the plan-side query.

We want the count of each craft type to each plan as actually engaged in the field. Therefore, the aggregation type is count using the CraftType field. Alternatively, we could have chosen to count on employee number.

Date_Week	Day_Date	Order_No	Task_No
2/3/2020	2/5/2020	6001	10 He
2/3/2020	2/5/2020	6001	10 Me

Craft	Count_Plan	HoursPerCraI	Duration_Pla	Hours_Total
0 Helper	0	0	0	0.5
0 Mech	2	2	2	4.5

SchedFlag	Planned	Status
S	P	3C
S	P	3C

CountOfCraf	SumOfHoursI	AvgOfHoursC
1	1	1
2	5	2.5

VarCraftCour	VarHrsPerCra	VarTotalHou
1	1	1
5	0	0.5

Output 11-3: Table of variances between job plans and field actuals.

We may want to place boundaries to the calculated variances upon which we would regard a job as an outlier warranting investigation. We can plant them as a variable in the table. Alternatively, we can set them as criteria to each variance.

If we want variances that are sensitive to groupings we can form and join a translation table of boundaries to the variance table. We can also upgrade to boundaries using the Student T statistic. With it, recall from Section 9.3, the boundaries adjust statistically to reflect the number of records to the allowable boundaries of variance to each group.

With the joined table, we can look for variances in almost infinite ways. We may want to explore a particular craft group. We may want to explore patterns to plans and actuals for types of assets. It all comes back to how we set and soup up the queries flowcharted in Figure 11-9.

11.4.4. Measures as Time Series

The explanation of the three sets of measures were presented as measures to the completed week. As such, they are updated daily and rolled up to weekly and appended to a table of all preceding weeks. The long-term granular transparent history allows us to measure in the context of time rather than only by weekly snapshots.

Chapter 10 introduced and explained how to engage the table of history in time series analytics with the “R” software. Time series analytics present us with insights that are not possible from snapshots and traditional line charts.

A primary insight is whether planning and scheduling are on beam to established set points for performance and controls. We want to confirm that the plant is a championship team rather than a team that has an occasional championship season. For example, are the incidences of allowed break-in work as percent of all initially scheduled work within boundaries.

For most of the measures we want to confirm that seasonal or other cycles are not, or are, influencing the series. If there are cycles, they must be removed from the series so that the shape of the core series is apparent. How to inspect for and remove cycles from the series is explained in Sections 10.2.2 and 10.3.1.

There is another purpose for inspecting for cycles. The principle of smoothed workload and fitted craft capacity imply that there should not be cyclicity or trends up or down. There should only be some randomness around the smoothed workload and occasional spike weeks. If the budget is built on multiple workload mixtures and levels during the year, the constancy should hold within each distinctive period.

We should also inspect for 7-day patterns as explained in Section 10.2.5. The issue is not necessarily that there should not be patterns. Instead, is a weekly pattern driving the plant to retain slack craft capacity? This can be the case if one or several days are overly influencing craft capacity in order to cover them.

Promises have been made for the ramifications of planning and scheduling in maintenance operations. We may want to prove them rather than ask management to take them on faith. Alternatively, if management accepts them on faith, they will want confirmation that they are being well and persistently conducted in accordance with setpoints of performance.

Accordingly, to prove the ramifications, we may want to search for lead-lag relationships between the measures of planning and scheduling and plant outcomes and other promised ramifications. For example, can we see a change in plant production some periods after a change in percent of completed orders that were scheduled in the day's or week's total workload? Section 11.2.4 explained the method.

Forecasting is also potentially interesting as quarterly or annual insight. We are not so much interested in forecasting the future because, by policy, it should largely mirror the past. Instead, we may explore for two other issues.

Section 10.3 introduced the Holt-Winters and ARIMA models for forecasting. They are not just a straight line fitted to a series of points. Instead, they fit a series model to the history of core series and its cycles and, in turn, project the patterns into the future.

One point for inspection is if the forecast fitted to the past is picking up something in the history that we cannot otherwise detect. If a core series is expected to be largely

constant, a forecast that shows otherwise should be explored as a bellwether of lurking issues.

A second bellwether is the spread of the confidence limits to the forecast. Is the forecasted spread changing? The ARIMA model (Section 10.3.2) provides insight after taking into account any autocorrelation that misstates the spread in series data.

Let's now speak to methodologies of time series. As just mentioned for each method of measure and underlying super table, there was a flowcharted step to append each day to week and each week to long-term history.

It is necessary to convert the variables of measure to time series objects. The first step is to form a query with which we would extract the variables to convert to time series objects. In the query we would aggregate variables upon the date intervals of interest: year, month, week, day and weekday. For the intervals, the variables to be the series are aggregated as counts, sums, averages and calculations.

Finally, the returned table is converted to a time series object with the "R" software. How is explained in Section 10.1.2.

In all calendar-based series, time intervals of less than a month are a problem. The solution lies in understanding what happens when calendar data is converted to a time series object. In the background, the dates are converted to a decimal relative to a fixed time period.

Therefore, for measurement intervals of week and day we can work around the problem through the start, end and frequency arguments to the times series function; taking advantage of the fact that the dates are decimal numbers in the background. How is demonstrated in Section 10.2.5.

It is likely that we want to inspect the different measures in parallel along a common time interval. Section 10.1.4 explains how. The result is a stacked set of plots of the parallel series. An example is Figure 10-5.

We may also want to place multiple variables in a single chart. Figure 10-14 is an example. How was explained in Section 10.2.4.

The section also explained how to convert measures to an index between 0 and 1. It is good trick when the compared variables are of different units or the magnitude of variables of the same units are too greatly different in magnitude. To avoid problematic overlay, we could add an offset factor to the indexes.

Bibliography

Alexander, Michael and Kusleika, Richard. Access 2016 Bible. John Wiley & Son, Inc. 2016. Parts 3 and 4.

352 | Chapter 11

Cowpertwait, Paul and Metcalfe, Andrew. *Introductory Time Series with R*, Springer. 2009.

Cran.r, Vignette for ggplot2 in time series. https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_ts.html.

Nyman, Don and Levitt Joel. *Maintenance Planning Coordinating and Scheduling*. Second edition. Industrial Press, Inc. 2010.

Oesko, Suljan. *Pivot Tables In-Depth for MS Excel 2016*. Suljan Oesko. 2017.

Palmer, Richard. *Maintenance Planning and Scheduling Handbook*. Fourth edition. McGraw Hill. 2019.

Chapter 12

Elapsed Time Through Stages

Recall from Chapter 1, availability is the probability that a plant, subsystem, asset or component will be in a state to perform a required function at specified standards of performance under given conditions when called for; assuming cost-effective support with respect to working conditions, processes and resources.

Availability is generally expressed as the interplay of reliability and maintainability intervals: the interval between down (reliability) divided by the sum of interval between down and the interval until up again (maintainability). Chapter 1 introduced three subtypes of availability: inherent, achievable and operational. They are distinguished by the characteristics of their reliability and maintainability intervals.

Inherent availability is the availability to be expected when the reliability interval is the result of running all assets to failure. In the computation of availability, it is mean time to failure (MTBF). Meanwhile, the maintainability interval excludes administrative and logistic time. Therefore, the maintainability interval is only elapsed time to execute maintenance tasks. Accordingly, it is mean time to repair (MTTR).

Achievable availability brings all types of scheduled maintenance into the reliability interval, supplanting many run-to-failure tasks. Consequently, in the overall computation of availability, it is mean time between maintenance (MTBM). Administrative and logistic time is still excluded from the maintainability interval. However, achievable availability now depicts a plant in which only some failures are consequential with respect to a specified standard of performance.

Operational availability adds logistic and administrative time to the interval to conduct maintenance tasks. Therefore, operational availability reflects the resource levels and organizational effectiveness and efficiency of the overall maintenance operation. Accordingly, the maintainability interval is lengthened in the calculation of availability.

Just as MTBM signifies, not all failures and maintenance present as a down condition. We must now think in terms of mean time to maintain (MTTM). There is more than just the time to execute work in the real-life maintainability interval.

Chapter 1 explained the use for each of the categories of availability. Obviously, operational availability is the "bottom Line" availability. It rolls up to the firm's financials and return on investment. As was depicted in Equation 1-4, the reliability and maintainability intervals are mean time between maintenance and mean time to maintain.

MTBM recognizes that, of all required maintenance, not all will reduce the current level of performance. MTTM accounts for the total time that a level of performance is reduced from a standard by the policies and conduct of the maintenance operation. Furthermore, not all maintenance interferes with uptime.

However, there are always snakes lurking in the grass because there is always maintenance work outstanding. Availability, as probability, tells us that a level of performance may hold but the probability of holding it has decreased until outstanding work is complete. Therefore, we must always strive to get out of Hell before the Devil knows we are there.

This means that work in the backlog is actually a risk of lost levels of performance. Worse is that how things could play out is beyond our ability to be prescient.

In an existing plant, rather than only asset reliability and maintenance strategies, our third major defense is to optimize the patterns by which orders are retained in the pipeline. The overall maintainability interval to availability is far more pliable than reliability and maintenance strategy.

The principles and methods of the chapter are to optimize the MTTM interval in operational availability. More specifically, we want to find the stages in the pipeline of work orders at which policy and process will reduce the time we are in Hell. Fortunately, there are analytics to do that.

12.1. The Insight We Need

Chapter 9 explained by demonstration how to build a budget as a smoothed mixture of maintenance type and counts to sustain market share and asset value. Chapter 11 explained by demonstration the scheduling and measurement to assure the conduct of the sustaining workload.

There is a third and much more subtle aspect. It is the retention time of orders through the stages in the pipeline from requested work until placed on a week's schedule. Furthermore, retention in most of the stages should be largely decided by the rule of first in, first out (FIFO). Of course, there will always be some small number of orders that must jump the line for good reason.

If retention is unnecessarily overlong, the plant will likely lose its FIFO flow. The timing of orders in the process will be gamed. Alternatively, if retention is too short, the plant will lose the buffer in the system that assures there is enough work ready to schedule each week as a smoothed sustaining workload.

As some orders are undeservedly privileged, gaming the retention rules generates other problems. As the under-privileged orders experience unhealthy overlong retention,

the likelihood of losing the current level of plant performance is building. Furthermore, gaming will create a culture of distortion throughout the CMMS as the result of hiding the gamed orders.

The loss of buffer in the inventory to each stage in the pipeline, has its own implications. The ability to operate a smoothed weekly schedule will be lost. As the buffer moves toward zero, the weekly schedule would become increasingly reactive in mixture and counts of orders. The schedule would increasingly track the arrival of new work into the pipeline rather than work withdrawn smoothly from the pipeline.

One reason reactivity is to be avoided is its effect on the payroll cost of workload. As the schedule is more reactive, the plant it will require a larger craft force to cover each week's workload. Not only can this noticeably bite into net margin, but it may also bite into market share if the firm competes in cost leadership industries.

There is an analytic to measure the work pipeline. This chapter will call it retention-event analysis. However, it is normally called by names such as survival-hazard and event history. The remainder of this section will explain retention-event as a concept. The next will explain the mechanics and mathematics to the analytic. The final section will demonstrate the analytic as computer-based methods to design the patterns of retention and events for the plant.

12.1.2. Retention in the Pipeline

“Retention” is the synonym to “survival” in survival-hazard analytics. In plain English, it is the percent, portion or probability of orders remaining in the pipeline beyond some duration. Figure 12-1 depicts the general pattern for the stages in a maintenance process as the pipeline. It is depicted that only 5 percent of the work orders remain in the pipeline beyond duration t . We can also see that some much smaller portion of orders have an event after a very short duration.

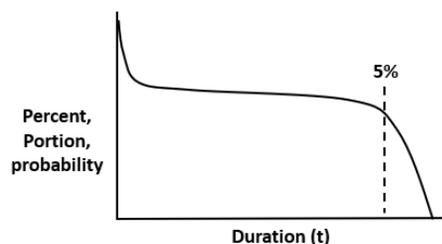


Figure 12-1: General pattern of retention as in a pipeline of work.

With the insight that is inherent to the figure, we have questions to which we can now know to seek answers. Is the shape as we would expect or wish for? Is the duration

excessive? Is the duration such that there is no buffer in the pipeline as we would want there to be?

In real life, we would want to segment the pipeline as stages from requested work through to work ready to schedule for execution. The final event, at duration t , is to be placed on a week's schedule. Any sequence of activities in the flowchart of the maintenance operation can be subjected to retention-event analysis.

Figure 12-2 shows the idea as a pipeline with three stages. We would want to explore some or all stages in the pipeline. This is because we would expect to find different patterns in the stages; some good, some bad.

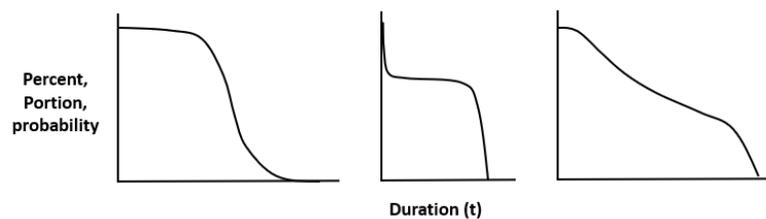


Figure 12-2: Stages to a pipeline of operational activities.

Now our questions are directed at the stages. They are also directed at patterns to confirm that there is compliance with the rules by which work is to flow through each stage of the pipeline. We are also looking for the stages that, if redesigned for planning, organizing and control, would most influence availability performance and the risk of holding a current level of performance.

For some stages we would expect a sharp drop off after an expected duration. Why is that not so? For others, is the drop off gradual rather than precipitous as we would expect? Are there indications that orders are allowed to stagnate in the pipeline? For the stages over which we have full control, why is the duration much longer than the activity of the stage? For those over which we have limited control, such as waiting for parts, should we institute practices to expedite flow through the stage?

Upon the findings of our questions, we may want to redesign the planning, organizing and control of at least some of the stages. For example, as the point of Chapter 9, corrective maintenance as smoothed mixture and count occur for a range of assets and failure modes. However, we may want to establish retention for the assets in the pipeline that represent the greatest risk of losing a current level of performance. If not possible, asset-specific rules to the pipeline can be promulgated for selectively jumping the standard FIFO rule.

We have presented the idea of sectoring the pipeline. The discussion of the idea so far has been one of the many.

It is likely we would want to subset a particular pipeline. The sections depend on what we are looking for. We may subset or group on cost center, maintenance type, asset type, craft team, etc.

Section 12.3.3 will demonstrate how to test for effects of predictor variables on the retention and event patterns. If there are effects, we may want to generate the retention analytics and plots upon them.

12.1.2. Exit Event from the Pipeline

The probability of an exit event is the other shoe to retention. It is the instantaneous chance of exiting the pipeline stage. As a point of reference, in the parlance of survival-hazard, it is called “hazard.”

The definition of hazard is that, on the condition of having lasted up to “just” before now in a stage, there is a probability of an exactly specified exit event from the stage. Some gallows humor will make the point.

When I woke up on the morning of my 73rd birthday, I was available to die. According to actuarial tables, my probability of dying during the year was 2.96 percent. Connected to that, the retention function predicts that I might last another 12 years. Should I make it to my 86th birthday, there is a 10.87 percent chance of not living out the year. The good news is that the retention function would expect that I might last until my 92nd birthday.

The same description applies to the previously defined work pipeline. Figure 12-3 depicts the exit function to the retention function of Figure 12-1. It follows that there is a similar function to each of the three stages in Figure 12-2. The reader will also recognize the formulation of the hazard plots will return one of the six life patterns (see Chapter 1) as input to formulating maintenance strategy with the methods of reliability centered maintenance.

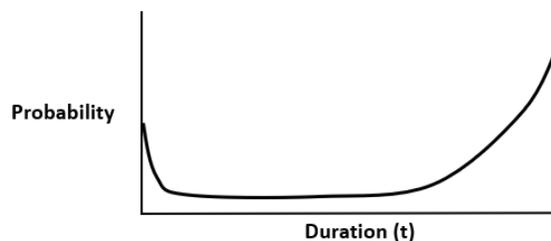


Figure 12-3: Instantaneous probability of the specified exit event.

Also notice that there is a chance of an early exit event as allowed by policies for the flow of orders through the maintenance process. Then, the chance of an exit event is

generally constant. As orders reach the point in retention at which between 10 and 5 percent are still in retention, the chance of exit increases rapidly.

Cumulative probability of an event is exactly that. At duration t , the probability of an event is the sum of probability of an exit up to duration t .

More typically, we are interested in the cumulative probability or hazard between two points in duration. Let's return to my age. As my aunt said at 93 about her chances of an event, "It's not going to get better." My chance of exiting life before my 75th birthday increases each year. The cumulative probability is 6.18 percent.

The probability of event is not as much of interest as is cumulative probability. This is so if we want to measure the probability of an event in some sector of duration after having a reached a time in retention.

The probability of an event has its meaning in understanding how it plays in the calculation of the retentions as shown in Figure 12-1 and 12-2. Section 12.2.2 will explain the construction.

12.2. How it Works

This section will introduce and explain the mechanics and mathematics of retention-event (survival-hazard) analytics. We now need to understand life data and events as a variable, study window, and construction and calculations.

12.2.1. Life Data, Event and Window

The life variable to each case in retention-event analysis is a composite rather than a numeric or a character variable. It is the composite of elapsed time and event.

We can define the life variable in two ways. One as the start and end dates for entering and exiting a stage to the pipeline or the pipeline. The other would be the calculated difference between the dates.

Just as retention has a start and end date, an event is defined very specifically. An example is the stage to prepare job plans. The specified exit event can be when the status is changed to "waiting for parts." If a case is changed to a status other than waiting for parts, it is not a specified exit event to the retention-event study.

This brings to surface the next issue. It is how orders that depart from retention to some other status than specified are treated in the analysis. They are identified as censored events. Weibull literature often calls them "suspended." Also designated as censored events, are the cases of which we have lost track.

Figure 12-4 depicts that we work with cases during a window of study based on the calendar. Notice that most cases, denoted by the symbol "+," exited as a specified event."

Among them, some cases depart the stage as something other than the specified event. As censored events, they are denoted by “c.”

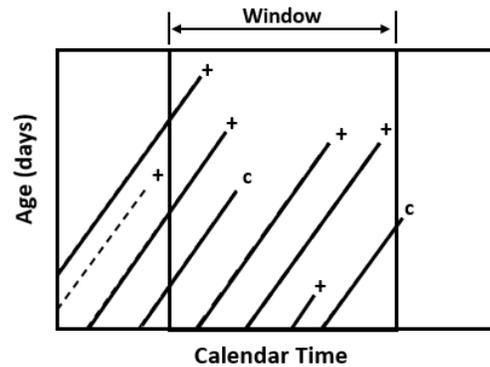


Figure 12-4: Cases as a specified or censored event during the window of the study.

There are several other points to note in the figure. The study is “left truncated.” The analysis only includes cases that still have not resolved before the left boundary of the study window.

Cases that did not resolve within the window of the study are also classified as censored. They are called right censored. The classification holds regardless of the ultimately occurring type of event. The principle is that the study has no way of knowing the ultimate resolution of the cases.

As mentioned, Figure 12-4 depicts having set the window with respect to only calendar boundaries. However, age is the second dimension to a study window. In the figure we have accepted cases of all ages if they met the criterion for date.

Figure 12-5 depicts having additionally set a criterion for age rather than include all ages. In contrast to Figure 12-4, we have defined upper and lower limits to age. As a result, we can see that two cases are no longer included in the study window. Also notice that a case entering the window, but with an age above the window frame, is treated as censored.

With respect to Figures 12-1 and 12-3, we may have set the age boundaries in order to analyze the period in retention after early exits and before rapidly increasing exits. If we had wanted to segregate the study upon early exists, we would have lowered the upper age and allowed all orders at or below the age to enter the study.

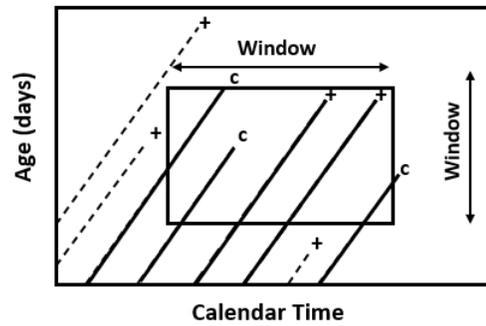


Figure 12-5: Window of a study with age as a dimension.

12.2.2. Construct of Retention and Event

Behind the curtains of the retention-event analytic there is an empirical construction. It is necessary to understand them as a construct. We will demonstrate the basic concept with what is known as the Nelson-Aalen estimator of probability of a specified exit event (hazard) and the Kaplan-Meier estimator of retention (survival).

Figure 12-6 is a plot of life data: duration and event. It demonstrates how the cases in a study window are transformed to the probability of a specified exit event after days in retention. As before, the symbols distinguish between specified and censored events.

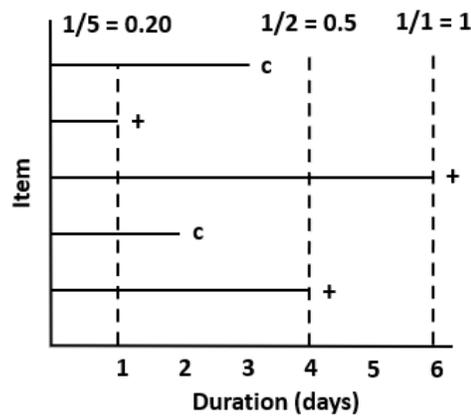


Figure 12-6: Probabilities computed upon specified exit events.

In the figure, we see that at each point in duration at which there is a specified exit event, the probability of exit is computed on the event. The demonstration of the computation includes every case still in the stage “just” before the time of the event.

Figure 12-7 depicts the transformation of Figure 12-6 to a plot of probability of a specified exit event after days in retention. In the parlance of survival-hazard analysis, it is known as the hazard function.

The probability at each exit event increases with duration of retention. The probability of a specified event on day three would be greater than 0.2 and less than 0.5.

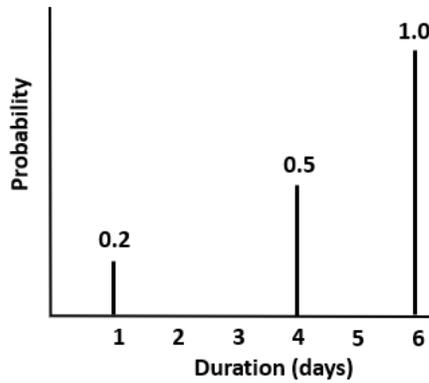


Figure 12-7: Plot of probability for specified exit events.

Cumulative probability of a specified event is a simple step from the plot of probability after days in retention. As can be seen in Figure 12-8, cumulative probability is just that. In the parlance of survival-hazard analysis, it is called the cumulative hazard function.

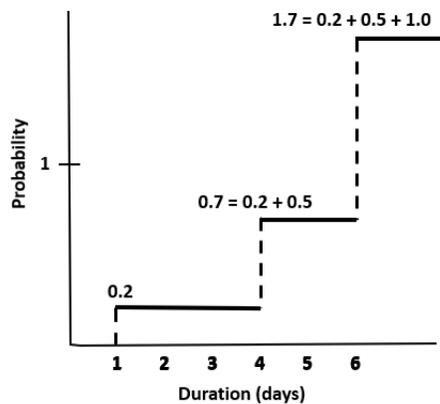


Figure 12-8: Cumulative hazard with time for an exit event.

The function tells us the cumulative chance of an exit the greater the number of days retention continues. Notice in the figure that, at some time in duration, an event is

essentially “statistically certain.” In the previous example of the actuarial tables, to live longer than expected (86th birthday) is to beat the odds and a new bet is placed (92nd birthday).

The probability of retention or portion of cases still in retention beyond a day in duration is also formulated upon the calculated probability of a specified exit event. This can be seen in Figure 12-9. Once again, in the parlance of survival-hazard analysis it is known as the survival function.

At duration zero, 100 percent of all cases will remain in retention. At day one, the portion of cases expected to remain will be reduced by the probability of exit. With each exit event, the portion remaining to exit is reduced as the product of the probabilities of exit to that time.

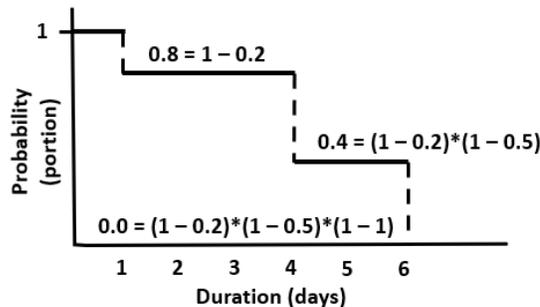


Figure 12-9: Computation of the portion of cases remaining in the stage at day in duration.

The explanation does not involve right-censored events that extend past the last specified event. The example of Section 12.3 shows a study in which some cases do. For it, the terminal point of the retention plot would be short of 100 percent of all cases. How far short of 100 percent would reflect the window of the study and the characteristics of the right-censored events.

12.2.3. Equations to Retention-Event

The previous figures depicted discrete rather than continuous plots of the functions. It is also possible that the characteristics of some censored cases in the data may cause the plots of retention to not go zero percent.

This is where the Weibull distribution enters the picture. With it, models can be fit to the data. The fitted distribution will return the parameters to engage in the equations to compute the points of interest to us with the retention-event functions.

There are two parameters to extract from the Weibull distribution. They are beta (β) and eta (η). Section 12.3 will demonstrate how to generate them with the R software.

The probability of an exit event at time t is the derivative of the failure function. The probability is expressed by the hazard function ($h(t)$).

$$h(t) = (\beta/\eta) * (t/\eta)^{(\beta-1)} \quad (12-3)$$

The computation is the continuous version of the plot in Figure 12-7. It computes the instantaneous probability of an exit event instantaneously after time t .

The cumulative hazard function ($H(t)$), as cumulative probability of a specified event, computes the sum of probability of exit up to time t .

$$H(t) = (t/\eta)^\beta \quad (12-4)$$

The computation is the continuous version of the plot in Figure 12-8.

Finally, there is the formula for mean time to a specified failure mode. In the context of retention-event analysis, it returns the mean time to specified event.

$$MTTF = \eta * \Gamma(1 + 1/\beta) \quad (12-5)$$

Conceptually, the formula is equivalent to the sum of retentions for all cases with the specified event and that divided by the number of specified events. That is if beta equals zero. Because it may not be. Mean time can always be computed as a function of beta and eta related by the gamma distribution.

12.3. Retention and Event Models

Let's now model the patterns of retention and events for a sample of work orders. The demonstration represents work orders that we will imagine to be in retention from the initial request for work through to being loaded on the week's schedule as work to be executed. The example will begin with introducing a dataset as variables suitable for retention-event analytics. Then, the R functions for the empirical and parameterized analyses of retention and events will be introduced and demonstrated. Finally, we will introduce how to determine if one or more operational variables are significantly related to the shapes of the retention-event curves.

12.3.1. Data to the Analytic

As mentioned, the dataset is representative rather than actual. In a real setting, it would have been built from the status history of work orders. In this case, imagine that we

extracted all cases that were either requested or scheduled in February. However, our study window need not be a month.

Here we reach back to the orders that entered retention during and before the month and departed during or after the month. The window includes such orders without limiting their ages.

With the code below, we would import the excel spreadsheet, `tblDuration`, from the file, `SkillsForCareerSecurity_Datasets.xlsx`. The file is available to download at the webpage given in the preface. In turn, we inspect the data set we have named “dur.” The element, `<path>`, in the code depicts that the reader is to insert the full path to the directory in which they have chosen to save the Excel file. The R script is also available at the webpage as `ElapsedTimeThruStages.R`.

Recall, that to import the data, we will need to call up the `xlsx` package with the `library` function. At the same time, the code calls for pulling up the other necessary packages, `eha` and `weibullR`.

```
#LIBRARIES TO LOAD
library(eha); library(weibullR); library(xlsx)

#IMPORT DurationDataSet.xlsx AND NAME AS DUR
dur<-read.xlsx("C:\\<PATH>\\SkillsForCareerSecurity_Datasets.xlsx",
  sheetName="tblDuration", header=TRUE)
head(dur)
str(dur)
```

In the upper portion of Output 12-1, from the `head` function, we see the Enter and Exit dates. They were extracted from the CMMS as status history to each order. The variable `Age` is the difference between the enter and exit variables. The event variable is “1” for the cases that exited retention as specified events and “0” for the cases that exited as censored events. The variable, `Planner`, identifies who planned and administered each order to exit.

In the lower portion of the figure, with the `str` function, we see the number of cases (151) and variables (5) to the dataset. Two planners were involved, Jones and Smith. Notice that the enter and exit variables became POSIXct variables upon importing. To use the variables as enter and exit dates to life data, we would convert the variables to a date with the `as.date` function. We have not because we will use the `age` variable instead.

```

> head(dur)
      Enter                Exit Event Age Planner
1 1900-01-07 00:00:43 1900-01-31 00:00:43    1  24  Smith
2 1900-01-10 00:00:43 1900-01-31 00:00:43    1  21  Jones
3 1900-01-07 00:00:43 1900-02-01 00:00:43    1  25  Smith
4 1900-01-12 00:00:43 1900-02-01 00:00:43    1  20  Smith
5 1900-01-12 00:00:43 1900-02-01 00:00:43    1  20  Smith
6 1900-01-15 00:00:43 1900-02-01 00:00:43    1  17  Smith
>
> str(dur)
'data.frame':  151 obs. of  5 variables:
 $ Enter  : POSIXct, format: "1900-01-07 00:00:43" "1900-01-10 00:00:43" .
 $ Exit   : POSIXct, format: "1900-01-31 00:00:43" "1900-01-31 00:00:43" .
 $ Event  : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Age    : num  24 21 25 20 20 17 16 19 18 16 ...
 $ Planner: Factor w/ 2 levels "Jones","Smith": 2 1 2 2 2 2 1 2 1 2 ...

```

Output 12-1: First six records and details of the dataset.

12.3.2. Empirical Modeling

The code below returns the empirical constructs of the portions in retention and probabilities of exit from retention. We use the Cox proportional hazard model, `coxph`, to return both.

```

#CREATE THE OUTPUT VARIABLE WITH AGE
fit<- coxph(Surv(Age, Event)~ 1, data=dur)
#EXTRACT IMPERICAL SURVIVAL TABLE AND PLOT
s.fit<- survfit(fit)
#SURVIVAL TABLE
summary(s.fit)
#SURVIVAL PLOT
plot(s.fit, lwd=2)

```

Notice the `surv` function as an argument in the `coxph` function. It creates the composite life variable in the model.

However, rather than the enter and exit dates in the `surv` function, we have used the Age variable as the time argument. It was computed in the source Excel table. Therefore, the variable equals the number of days between the enter and exit variables.

The expression, `surv(Age, Event)~ 1`, is the model equation. Notice “~ 1” in the model. It is because no predictor variables are included in the model. We are strictly inspecting retention and events without regard for whether there are predictor variables to their shape. This is called the baseline model.

After running the model, we would want to extract the table of retention and exit at day t . The `survfit` function, with `fit` as its argument, will do that. We have assigned its

output to the object named `s.fit`. Thence, the `summary` function with the object as its argument will return the table of Output 12-2.

```
> summary(s.fit)
Call: survfit(formula = fit)

   time  n.risk  n.event  survival  std.err  lower 95% CI  upper 95% CI
  2     150      1      0.993  0.00662   0.9805  1.000
  4     149      1      0.987  0.00933   0.9686  1.000
 14     148      5      0.953  0.01717   0.9204  0.988
 15     134      1      0.946  0.01844   0.9109  0.983
 16     127     10      0.872  0.02823   0.8186  0.929
 17     114     11      0.788  0.03504   0.7226  0.860
 18     101      5      0.750  0.03737   0.6798  0.826
 19      91      7      0.692  0.04031   0.6175  0.776
 20      80     10      0.606  0.04352   0.5267  0.698
 21      64     13      0.484  0.04609   0.4016  0.583
 22      50      6      0.427  0.04621   0.3449  0.527
 23      37      5      0.370  0.04652   0.2889  0.473
 24      29     12      0.219  0.04335   0.1489  0.323
 25      13      4      0.154  0.04090   0.0918  0.259
```

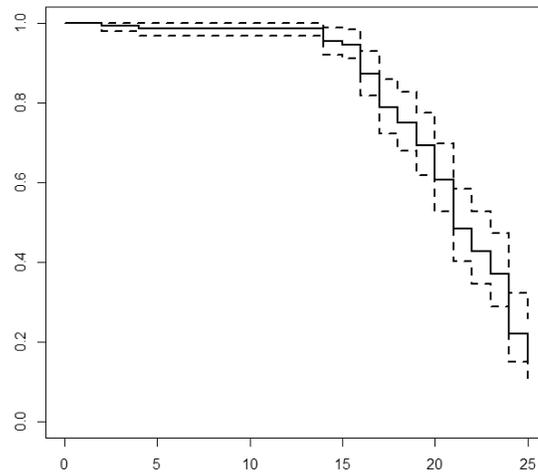
Output 12-3: Table of probability of exit and portion of retention at day t.

At day two there were 150 orders “still available to exit.” On that day one exited as a specified event. Therefore, the probability of exit at the day is $1/150$ and the portion of orders expected to remain in retention is $1 - 1/150 = 0.993$. Both are the computations depicted in Figures 12-7 and 12-9, respectively. Also, note that only the days at which there are specified events are there rows to the table.

At day four, with 149 orders available for a specified exit event, there is another exit event. Accordingly, the portion of orders expected to remain in retention at the end of the day is $(1 - 1/150) * (1 - 1/149) = 0.987$. At day 25, only 0.154 of orders are expected to remain in retention.

Let’s look at the retention plot that would be returned with the variables of Output 12-1. The bottom line of the block of code returns Output 12-3. The middle plot is the expected retention (survival). The upper and lower plots are the confidence intervals to the expectation.

We can see in the figure that there is a small probability of orders exiting early. Otherwise, we see the shape is largely level for 14 days until orders increasingly start to exit. As mentioned earlier, upon the shape, there is a host of questions we would ask of the pipeline of work orders.



Output 12-3: Plot of retention (survival) function upon the empirical calculation.

12.3.3. Parametric Modeling

We can fit parameters to the data such that the equations of Section 12.2.3 can be applied in the exploration of retention and events. The block of code below serves the purpose.

```
#PARAMETRIC FIT FOR BETA AND ETA
fit.w<- phreg(Surv(Age, Event)~ 1, data=dur)
summary(fit.w)
(beta<- exp(fit.w$coefficients[2]))
(eta<- exp(fit.w$coefficients[1]))
#PLOT UPON PARAMETERS
plot(fit.w, lwd=2)
```

Whereas, before we used the `coxph` function, we will use the proportional of hazard regression function, `phreg`. It fits the Weibull distribution to the data and returns the beta and eta parameters. Called up by the `summary` function, the fit is shown in Output 12-4.

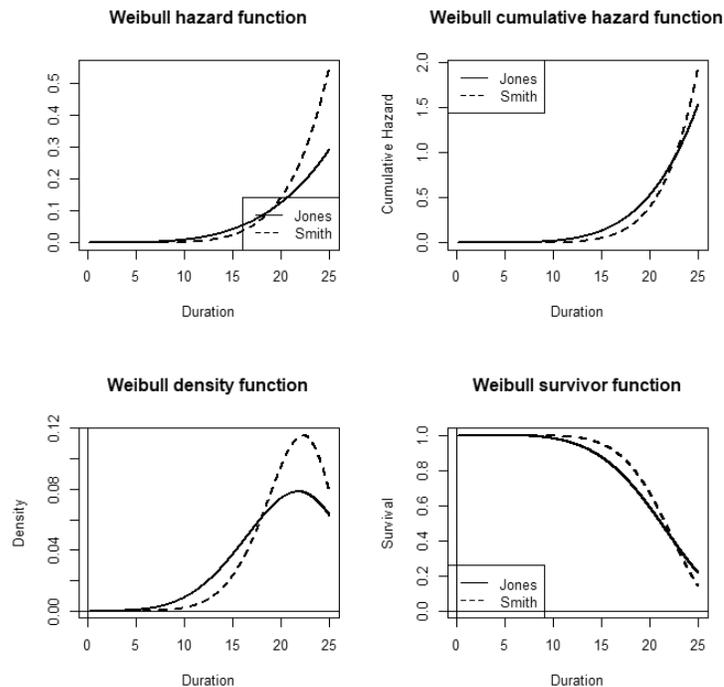
The report returned by `summary(fit.w)` returns the beta (shape) and eta (scale) as logs. The very small p-values reports that the parameters fit the data.

Before we can apply the parameters to the retention-events equations, we need to subject them to the exponential function. In the figure we see what is returned by subjecting the coefficients from the model to the `exp` function.

We need some of the code to arrive at the exponentiated beta and eta. If we subject `fit.w` to the `str` function, we would discover that one variable to the model is

```
plot(fit.w.st, lwd=2)
summary(fit.w.st)
```

The `plot` function returns the same four functions that were shown in Output 12-6. Visually, they suggest that there is difference between planners. But is it significant? Hint: the gap between the plots of the planners do not increase smoothly in the hazard and cumulative hazard plots.



Output 12-6: Plots of the retention-event functions stratified on planner.

If we subject the object `fit.w.st` to the `summary` function, we will receive the insight of Output 12-7. Recall that the covariates, 1 and 2, reflect the alphabetical order of Jones and Smith.

Just as for the previous figure, upon inspection we can see that the beta and eta parameters are different. However, once again, are they significantly different?

We can answer the question by analyzing the model with Planner as a predictor variable rather than an argument to the `strata` function. The code below does that.

372 | Chapter 12

```
> summary(fit.w.st)
Call:
phreg(formula = Surv(Age, Event) ~ 1 + strata(Planner), data = dur)

Covariate      W.mean      Coef Exp(Coef)  se(Coef)      Wald p
log(scale):1      3.129      0.032      0.000
log(shape):1      1.559      0.129      0.000
log(scale):2      3.125      0.020      0.000
log(shape):2      1.955      0.113      0.000

Events          91
Total time at risk 2919
Max. log. likelihood -298.15
```

Output12-7: Predictor variable stratified on its categories.

Just as was demonstrated in Chapter 7, models are coded as an outcome variable and predictor variables. Accordingly, notice the Planner variable is a predictor variable in the code below.

```
#REGRESSION ANALYSIS OF PLANNER TO OUTCOME
fit.w.pl<- phreg(Surv(Age, Event)~ Planner, data=dur)
```

Output 12-8 shows the report. The output is comparing Smith to Jones. The p-value is much greater than 0.10. This indicates that there is no meaningful difference.

We should note that the technique of stratification has limitations as insight. Only one strata at a time can be evaluated. The more insightful alternative lies in building a multiple variable regression model.

Also note that the model sets up like any regression. Chapter 7 explained regressions in detail. Part of the explanation, Sections 7.2.2 and 7.2.3, was how to model and interpret multiple predictor variables. All that was explained in the sections applies here.

```
> summary(fit.w.pl)
Call:
phreg(formula = Surv(Age, Event) ~ Planner, data = dur)

Covariate      W.mean      Coef Exp(Coef)  se(Coef)      Wald p
Planner
  Jones      0.456      0      1      (reference)
  Smith      0.544     -0.027    0.974    0.210    0.899

log(scale)      3.124      0.026      0.000
log(shape)      1.756      0.085      0.000

Events          91
Total time at risk 2919
Max. log. likelihood -300.85
LR test statistic 0.02
Degrees of freedom 1
Overall p-value 0.898901
```

Output12-8: Regression analysis to the survival-hazard function.

For example, we may want to evaluate retention by cost center, maintenance type, asset type and more. Once set up, we would determine which reflect significantly in the retention-event functions. This in turn, may send us to redesign aspects of operations that we discover are most significant to the maintainability interval.

12.4. Weibull Plot

Weibull plots on log-log scaled graphs allow us to manually determine the beta and eta of the retention-event (survival-hazard) functions. The previous section of the chapter introduced R functions as the means to generate them with analytics. However, as a graphic Weibull plots are still valuable insight. This section will demonstrate retention-event analytics with the Weibull plots.

The value of the Weibull plot is that it is another way to examine the characteristics of the data. The weakness is that we cannot easily see the empirical patterns to retention and events.

Weibull plots are typically set up in software as two vectors. The `wibullR` function of R is no different. One vector is for specified events and the other for censored events. The `wibullR` function takes them as two input vectors. The approach of Abernathy's text, *The New Weibull Handbook*, also sets up with as two vectors: failures and suspensions.

The first two blocks of the code below build the vectors. In the first line of code to the blocks for exits and censors, the cases for each are being extracted as a table. Then for each, the `as.vector` function extracts the `Age` variable and gives it the title of `sensor` or `exits`. If we called up the `sensor` and `exits` variables, we would see a list of the ages to each type.

```
##WEIBULL PLOT
#VECTOR OF CENSORED CASES
cen<- dur[dur$Event == 0,]
str(cen)
sensor<- as.vector(cen$Age)
str(sensor)

#VECTOR OF EXIT EVENT CASES
ex<- dur[dur$Event == 1,]
str(ex)
exits<- as.vector(ex$Age)
```

Now that the input vectors are formed, the next block of code will return a plot of the Weibull shown in Output 12-9.

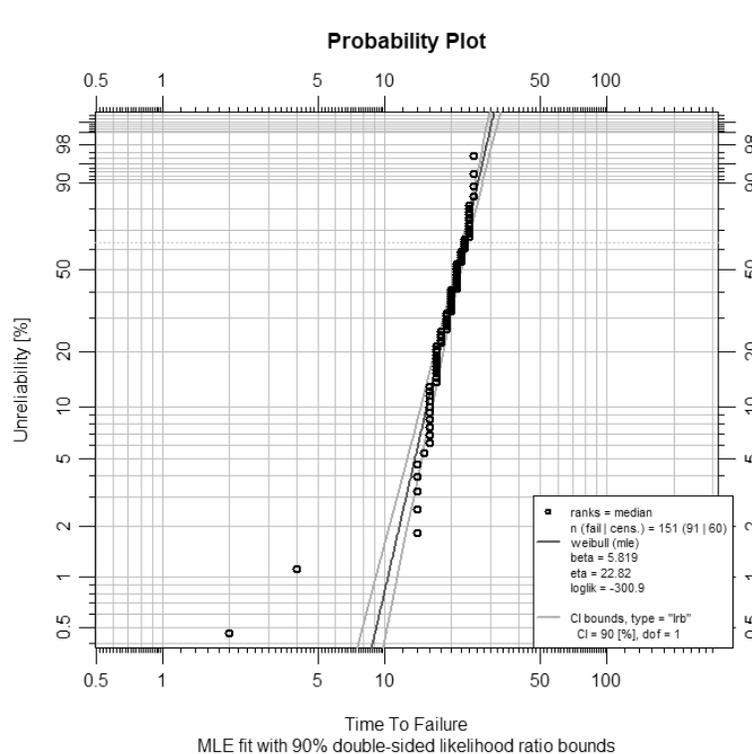
374 | Chapter 12

```
#PLOT THE WEIBULL PLOT
#BEST MODEL IS THE MLE MODEL BECAUSE OF HIGH NUMBER OF CASES
fit.dur.MLE<-MLEw2p(exits, censor, bounds = TRUE, show = TRUE)
#MRR MODEL SHOWS POOR FIT
fit.dur.MRR<-MRRw2p(exits, censor, bounds = TRUE, show = TRUE)
```

Notice that two versions of fitting the distribution are called up by the code. The first is the Most Likely Estimate method (MLE). The second is the Median Rank Regression method (MRR).

In the code to both fits, the arguments `show=` allows us to return either a plot or merely the parameters. The argument `bounds=` allows us to show confidence limits on the plot.

If we ran both, we would find that the MRR method shows a questionable fit, whereas the MLE method is the better fit. Also notice that the beta and eta of the MLE fit agree with the results shown in Output 12-4.



Output 12-9: Weibull plot of specified exit events.

The choice for MLE over MRR is due to the number of cases. Reliability engineers typically see the Weibull applied to a small number of cases. In fact, that we can draw conclusions from a few cases, is one of the virtues of the Weibull plot method. For them,

the MRR is the better of the two. However, the example entails 151 cases and, thus, the MLE method is better advised.

In the fit, notice that the fitted Weibull distribution regards the orders that exited early as outliers. If eliminated, the beta would rotate a bit counterclockwise. In contrast, the earlier empirical plot of Output 12-3 shows quick exits.

Section 12.3.3 recognized eras in the pattern and that the Weibull distribution can only plot one era at a time. The plot tells us a great deal about the era of rapid exit because it is pronounced in the pattern.

As explained in Section 12.2.1, to explore the other two eras, we would subset the study window on ages associated with the eras. When we do, we should also test which of the MLE and MRR options are the best fit. We should because exit events will likely vary considerably in count.

Finally, let's relate the plot to the previous analytics and equations. The y axis of the plot is a calculation of 100 percent minus the retention (survival) function of Output 12-5. As the complement to retention, it is reporting that at time t , a percentage of cases has experienced the specified event. If we sought to know the duration at which 10 percent of the orders are expected to remain in retention, we would read η at 90 percent on the y axis.

It goes without saying that the parameters reported in the text box would be substituted into the equations of a retention-event (survival-hazard) analysis. Also notice the text box reports that there were 151 cases with 91 specified events and 60 censored events.

Bibliography

- Abernathy, Robert. *The New Weibull Handbook*, 5th edition. Robert B. Abernathy. 2004.
- Brostrom, Goran. *Event History Analysis with R*. CRC Press. 2012.
- Moubray, John. *Reliability Centered Maintenance*. 2nd edition. Industrial Press, Inc. 1997.
- Silkworth, David and Symynck. Jurgen. Package 'WeibullR.' 2019. <https://cran.r-project.org/web/packages/WeibullR/WeibullR.pdf>

Chapter 13

Prove There is a Difference

Any operation within an enterprise entails countless practices, processes and tasks. In operations, the holy grail is to “do the right things right.” It follows that the holy grail of insight is to prove that we are.

When new operational strategies are being proposed, we must be able to answer the CEO who was known to ask of every proposition, “Why bother?” Of course, she would later want to know if to bother had proven to be worthwhile.

Figure 13-1 demonstrates the problem we face. The vertical axis is the mean of some measure of performance. Confidence limits overlay the columns. The charts present a question. Are the means between alternative strategies, practice, design, etc., significantly different?

The left half of the figure depicts a comparison of operational designs. It shows the comparison of designs for superiority based on the mean of an outcome. Most performance indicators are a mean of performance or indicators of performance. The right half depicts a before and after comparison of consequences.

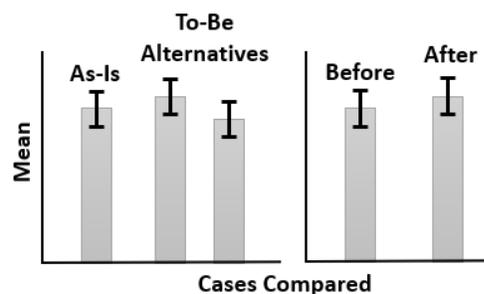


Figure 13-1: We must determine if the mean to each differ from others.

In the two graphs we see that the columns, with respect to mean, are different. However, the difference between columns may be only noise rather than “worth the bother.”

If we want management to fund our propositions for asset management skills, practices and processes, we may need proof that there will be or has been effects. The expression, “may need proof,” is because some of us may have management that takes our word for it. We heard it in a love song, can’t be wrong. This chapter and book is written

on the presumption that we, as asset management practitioners and leaders, and the management to which we report, always want proof over faith.

The chapter will first introduce the body of models we would engage to confirm performance. Rather than a single model, what it is we are trying to prove to ourselves and management decides which of seven types of models we would adopt. Following that, the remainder of the chapter will demonstrate each type of model and how to interpret them.

13.1. Effects Distinguished as Models

It is a simple question. Is there difference in performance between operational alternatives or operational performance over time? In statistical speak, we are asking if the comparative means in outcomes are significantly different between one or more alternatives or over time.

The question is simple. However, answering it depends upon the number of factors and conditions to each. It also depends upon how the tested participants as individuals, entities, assets, etc., are subjected to the alternative conditions as independent or dependent. Therefore, our first task is to establish which one of a body of models will give us the insight we seek.

This section will introduce the models in the context of possibility of insight. However, let us first set some terminology upon which distinctions will be made.

The behavior and performance of any operation, including asset management, is a system of dynamic operational factors. For almost every imaginable factor of operation, there are factors of two or more mutually exclusive conditions.

Tiny or immense, there will be a consequence to the choices made between conditions. The consequence will present as an effect, measured as a mean, to a scoring outcome variable. Our mission is to determine if the effect is significant or merely happenstance.

Analysis of effects is a huge subject. In any serious text on statistics, the subject is typically a third. Because the focus is asset management rather than statistics, this book has reduced the explanation to a workable chapter.

The strategy is to explain the critical mass of what a reader needs to know to competently work with the analytics of difference. In other words, what does the reader need to know rather than everything there is to know.

However, it is always good to have a full background knowledge of the methods. Should a reader be so motivated, Chapters 9 through 14 of the Field text, *Discovering Statistics Using R*, is a recommended read.

13.1.1. Single Factor, Two Conditions

The first distinction for modeling is the number of means being compared by virtue of the number of conditions to a single operational factor. If there are only two conditions, the method to apply is called the two-mean t-test.

As a calculation, it combines the variances to the respective means of the two conditions. Figure 13-2 shows the concept. Notice that we are testing the gap between the means of the conditions against the confidence limits of our conditions, e.g., 95 percent confidence. The standard deviation is calculated as the joint variance to the respective distributions. For an in-depth explanation of the calculation, see Section 9.5.1 of Fields.

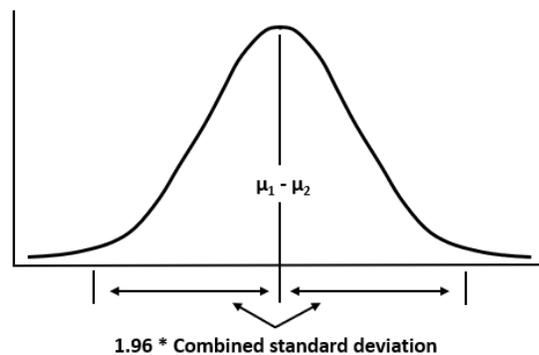


Figure 13-2: T-test for two means.

We are testing whether the gap between the means of the two conditions is larger than the confidence interval? If so, there is a significant difference in means between conditions.

There is another distinction to recognize. It is the difference between an independent and dependent test.

If each factorial condition is applied to a single participant, we have what is called an independent t-test. If each participant is subjected to both conditions, we have a dependent t-test. The dependent case is often called the paired two-mean t-test. Both cases are demonstrated in Section 13.3.1.

13.1.2. Single Factor, Three or More Conditions

Now to step up from comparing only two conditions. We still have a single factorial variable but with three or more conditions.

Our first instinct is to apply the two-mean t-test to each of the possible three pairs. The problem is what is called familywise error. For the two means shown in Figure 13-2,

we are 95 percent confident we would not make a Type I error. A Type I error is akin to a doctor diagnosing a male patient to be pregnant.

If our assumed confidence is 95 percent of not making such an error for each set of means, then the joint confidence is a power of 95 percent. For three conditions, the true joint confidence for each paired t-test is 0.95 to the power of three: 86 percent. This is not an acceptable confidence. Moreover, it declines as the number of conditions are increased.

Figure 13-3 shows the problem. By inspection of confidence limits to each, we may surmise that there are significant differences between A1-A3 and A2-A3. However, the concern for true confidence should cause us to doubt.

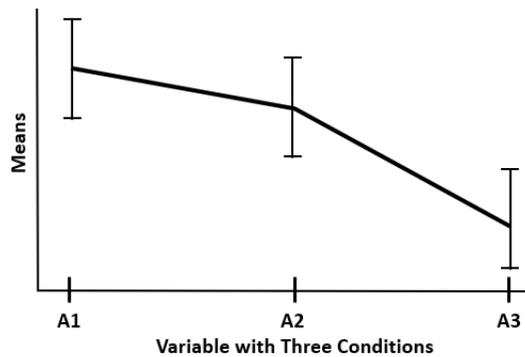


Figure 13-3: Means and confidence intervals to three conditions.

To get past the familywise error, Figure 13-4 shows the concept of ANOVA for three or more conditions. It is called the one-way ANOVA.

In the figure we see the idea of a factorial variable with three conditions. We also see the base model which is defined as a model without a factorial variable as the predictor variable.

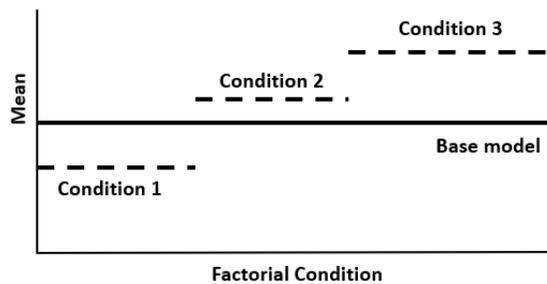


Figure 13-4: The concept of ANOVA as conditions.

In other words, the base model is the mean score to all observations. We can think in terms of it as having a grand variance. Respectively, each of the conditions has a mean, variance and residuals of its own.

The idea of ANOVA is to measure the extent that variance to the three conditions explain the grand variance. The extent is tested as ratio of the explained to the unexplained variances. If the ratio is high, there is one or more significant effects. In other words, one or more of the pairs of conditions in Figure 13-4 are significantly different.

However, ANOVA does not tell us which pairs are significantly different. For that we revert to the linear model. With the model we can use one or both of two methods to find the truly different pairs. They are called post hoc and contrasts. Both will be explained in Section 13.2.

Just as for the two means t-test, there is the matter of conducting the one-way ANOVA for independent and dependent models. In the parlance of statistics, dependent models are called repeated. For them, we will engage a different type of model called the multilevel general linear model.

The independent model will be demonstrated in Section 13.3.2. The repeated model is the subject of Section 13.3.5.

13.1.3. Covariant Variable in ANOVA

So far, we have spoken only of variables with categories or levels as conditions. In statistic-speak, they are experiments. For example, if we measure performance with respect to a process design, the design is the experiment.

A continuous variable, as a covariate, is neither an experiment nor part of the experiment. However, if placed in the ANOVA, it may explain some of the total unexplained variance. In turn, this allows the comparison of means to be made with sharper confidence limits.

However, as shown in Figure 13-5, there is a requirement. The covariate must not share explained variance with the factorial variable. This is also the rule if there are more than one factorial variables.

The left side of the figure shows a model without a covariate. The right side depicts the covariate as two possibilities. The upper right is what we want. The covariate explains part of the unexplained variance. Contrast that with the lower right. The covariate explains a part of the explained variance it shares with the factorial variable.

We need to test that the covariate variable is independent of the factorial variable, the upper right scenario. We test by building a linear regression with the covariate as the

dependent variable and the factorial variable as predictor. If the model reports no relationship, high p-value, we can use the covariate.

The ANCOVA model is demonstrated in Section 13.3.3.

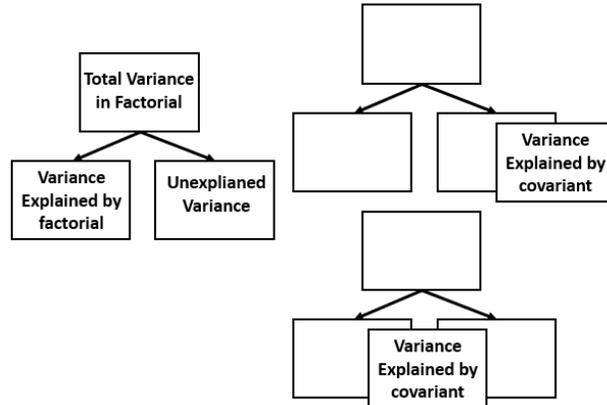


Figure 13-5: The requirement for independence of the covariant variable in the model.

13.1.4. Two or More Factorial Variables

So far, the explanation of ANOVA has been what is called one-way. There is only one factorial variable with three or more conditions. We added one or more covariates to the idea. Now we step up to two or more factorial variables. It is called factorial ANOVA.

The immediate consequence is that we have a longer list of combinations to inspect for significant differences. They are the permutations of the conditions of the factorial variables. Figure 13-6 depicts the situation for two variables.

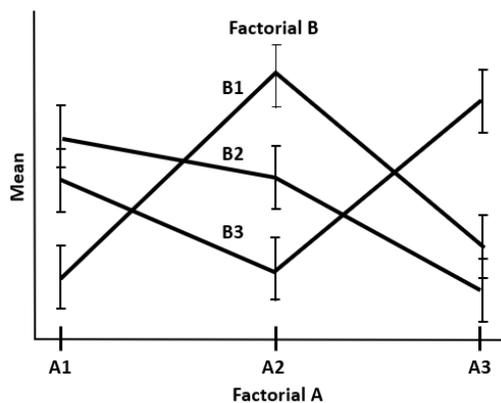


Figure 13-6: Two factorial variables with three conditions to each.

We can see in the figure there are now nine pairs to evaluate. Because of familywise error, judgements made with the two-mean t-test would have a confidence of only 63 percent.

The figure demonstrates a new issue. Is there interaction between the conditions of the factorial variables? There is in the depiction. They are evidenced by non-parallelism between line segments. In some cases, the extreme is such that lines cross.

However, the somewhat non-parallel lines are conceivably not significant interactions. It is possible that the appearance is not actually significant. It is a matter of degree. For this reason, we turn to statistical methods to make the determination.

Next imagine three or more factorial variables, each with two or more conditions. The visualization of the type of Figure 13-6 cannot easily deal with the evaluation. We must visualize two variables, while the others are only in our mind’s eye. In other words, we can only effectively explore two dimensionally.

Factorial ANOVA enables us to explore the factorials as a system. In addition to determining which pairs are different, we can identify which are the true interactions across factorial variables. The factorial ANOVA is demonstrated in Section 13.3.4.

13.1.5. Measures are Repeated

Recall that we earlier distinguished between independent and dependent measures. The latter is also called repeated measures. To this point all has been explained in the context of independent ANOVA. Now to switch to repeated measures.

Recall that, for independent models, each condition of the factorial variables is measured of a single participant. In contrast, all conditions of the factorials are measured of each participant.

Figure 13-7 shows the contrast. Notice the location of the unexplained variance.

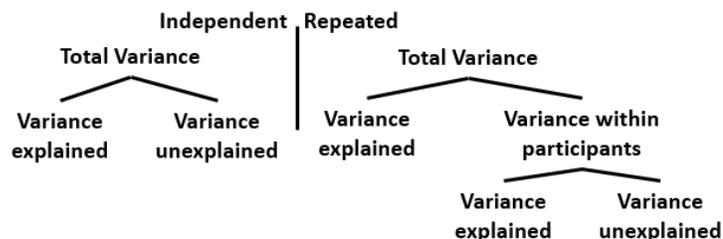


Figure 13-7: Comparison between independent and repeated measures models.

The left side of the figure is the independent model as was seen earlier in Figure 13-5. Unexplained variance was associated with the factorial conditions. Therefore, the variance was “between” participants.

The right side of the figure depicts the repeated measures model. The unexplained variance is associated with each participant. It reflects that each participant will vary in their response to each condition. In statistics speak, the unexplained variance is “within” the participants rather than “between” the participants.

Repeated measures ANOVA requires a different model. We can use variations of ANOVA such as the ezANOVA package. However, we will draw upon what is called a multilevel model.

One reason is that repeated ANOVA models are restrictive and sensitive to certain easily violated arcane assumptions. Furthermore, it is good to introduce the multilevel model because, with it, we can model possibilities beyond ANOVA. In this case, we are using it to work with the repeated measures in the comparison of effects.

Repeated measures for one-way and factorial analyses will be further explained by demonstration in Sections 13.3.5 and 13.3.6. The demonstration will be with the multilevel model. If the readers wish to explore the ANOVA approach as an extension of the previous sections, the method is Section 13.7.4 of Field.

13.1.6. Mixed Models

Now enter mixed models. As the name suggests, a mixed-design model will treat the relationship of some factorials to participants as independent and others as repeated.

Imagine a case where a cost center is being evaluated with respect to all conditions of one or more factorials. At the same time, we wish to evaluate the differences within cost centers reflective of equivalent plants. The differences in effects may be influenced by leadership and other local characteristics to the cost centers.

However, we may expect that there is an independent relationship between craft types or grades. In other words, there are variances “between” participants with respect to craft categories.

The multilevel model allows us to concurrently explore between and within relationships. Building a mixed model is a matter of where factorial variables are placed in the models. Section 13.3.7 will explain the how mixed models are configured.

13.1.7. Robust Models

For all parametric models, there are assumptions. If not met, the accuracy is undermined. When a model fails tests to validate the assumptions, there is a body of robust models to turn to.

The purpose of the chapter is to introduce and explain by demonstration the range of analyses around the nature of effects being evaluated. It will concentrate on the parametric methods. It will not expand the explanation to the robust models to each.

There are robust methods to all introduced analyses except for the repeated measures and mixed model. It is left to the readers to put them in play if an assessment of the parameterized model reveals the necessity. They can be found in the referenced chapters of Field for each parameterized method.

13.2. Test by Post Hoc and Contrasts

As has been mentioned and will be seen in Section 13.3, the ANOVA is an omnibus analytic. What is meant by that is it determines for us if there are one or more significant differences between the effects of the factorial conditions but does not identify which.

Instead, we must call upon additional methods to ferret out the differences. There are two that are engaged in all models except the two-mean t-test. They are called post hoc adjusted confidence limits and contrasts as effects.

Before moving to demonstrating the models, this section will introduce the principles of both. Section 13.3 will explain them further in the context of demonstrating the analytics that were introduced in Section 13.1.

13.2.1. Post Hoc Adjusted Confidence Limits

Recall the principle of familywise error. In Figure 13-8, we may be sorely tempted to accept that there is significant separation of the confidence limits between the conditions of X_1-X_2 and X_1-X_3 . But we also know we can only be 86 percent (0.95^3) confident in our interpretation.

What is depicted in the figure is to apply post hoc adjustments to the nominal confidence limits. When done, it appears that we can be 95 percent confident that, of all the pairs, only X_1-X_3 are significantly different. It is the only one with non-overlapping confidence limits.

There are many offered adjustment methods to choose from. At one time, devising new post hoc adjustments was a growth industry in the world of statistics. This chapter will limit itself to four: Bonferroni, Benjamini-Hochberg (BH), Tukey and Dunnett.

We can see the idea with the Bonferroni as an example. The critical p-value is the desired alpha, 0.05 ($1-0.95$), divided by the number of pairs, k . Accordingly the critical p-value for significance is 0.016. The post hoc limits depicted in the figure represent the confidence intervals associated with the critical p-value.

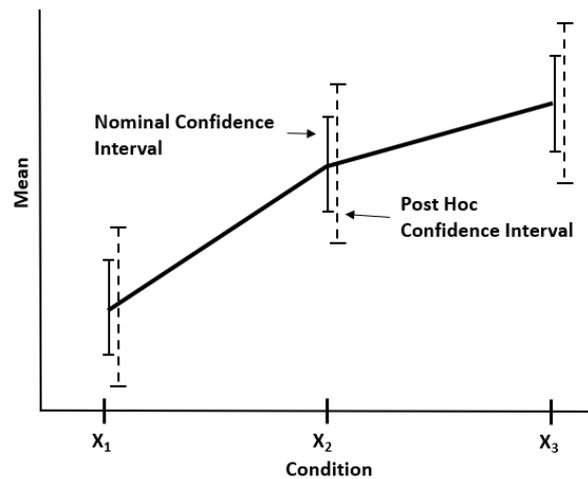


Figure 13-8: Means and confidence intervals of conditions to a factorial variable.

Of the four methods, there are variations to the calculation. A deeper, but not extreme, explanation of the calculations is provided by Section 10.5 of Field.

The big issue underlying any chosen post hoc method is the degree that it protects against Type I error without a commensurate loss of what is called “power.” Power is the ability to conclude there is no significant effect when there is not. With power comes Type II error. As the other shoe to Type I error, it is akin to diagnosing a full-term pregnant woman as not.

The point to accept and live with is that we cannot have our cake and eat it too. With increased protection against Type I error there is a loss of power, and vice versa.

The Bonferroni and B-H adjustments are conservative. They protect heavily against Type I error. The Tukey and Dunnett offer greater power along with reasonable protection against Type I error.

Which of the two sets to apply is guided by the purpose of the research at hand. Which of the two types of error do we most want to avoid? If Type I error is our concern, we would favor Bonferroni and B-H. If Type II error is our greater concern, we would favor Tukey and Dunnett.

13.2.2. Contrasts as Effect

Like the post hoc method, contrasts get us past the problem of familywise errors. The idea is to play on the explained variance that was introduced in Section 13.1.2. The explained variance is successively sliced into pairs of variances. Each pair is called a contrast. This allows us to compare slices as pairs but without familywise error.

We will explain the method with Figures 13-9 and 13-10. Section 13.3 will demonstrate in action what is depicted.

Recall the principle of dummy variables explained in Chapter 8. They were used to convert the levels of categorical variables to numeric variables. In fact, somehow converting all variables to numeric is a prerequisite to modeling. Therefore, we once again create dummies. This time, dummies enable us to model for significant effects in the context of contrasts.

Figure 13-9 shows the process of formulating what is called “weights” as contrast variables. Two alternatives are demonstrated.

The elements C1, C2, C3 and C4 are the conditions to an imaginary factorial variable. Look at the first alternative.

We start with total variance segregated as explained and unexplained variances. Under the branch for explained variance, we are slicing the variance to be pairs. Once a condition is sliced into a group of one branch, it can never again appear in another branch.

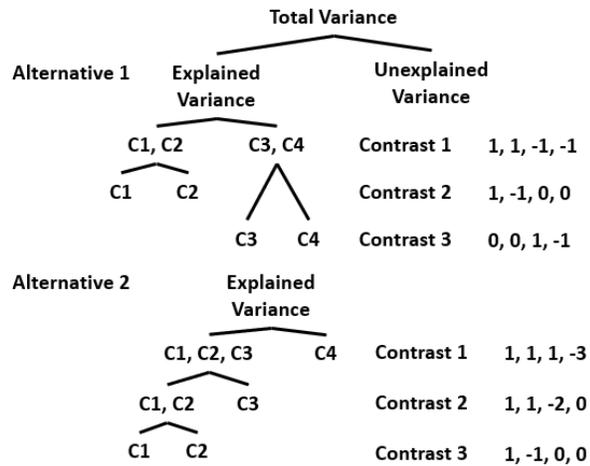


Figure 13-9: Steps to formulate contrasts.

In the hierarchy of branches, each pair constitutes a contrast that will be installed as a dummy variable in the model. There will always be one less contrast than there are conditions to the factorial variable.

Next is to set the “weights” of each condition in a contrast. For the ANOVA analytic, the rule is that the weights must sum across the contrast to equal zero. As contrast 1, we weight both C1 and C2 as 1 and both C3 and C4 as -1. Consequently, the sum of the weights is zero (1 + 1 - 1 - 1).

We have just specified a comparison of the average of the means for C1 and C2 and the average of the means for C3 and C4. Ultimately, the linear model will test for significant difference between the averages.

At the lower branch of the two groups, we build a contrast for each. As before, we place the conditions in paired groups and set the weights to sum to zero.

Contrast 2 will compare the means of C1 and C2. Contrast 3 will compare the means of C3 and C4.

An additional rule for assigning weights emerges in Contrasts 2 and 3. Look at the second contrast. Contrast 2 only incorporates C1 and C2. Therefore, 1 is assigned to C1 and -1 to C2. Because C3 and C4 are not included, weights of 0 are assigned to them. This pattern is mirrored in the third contrast.

The contrasts are our choice to make. Alternative 2 in Figure 13-9 is an example of a different set of contrast for the same factorial variable.

In the first split, the average of the means to C1, C2 and C3 is being contrasted with the mean of C4. Accordingly, C1, C2 and C3 are given weights of 1 and C4 a weight of -3.

The next contrast continues by comparing the average effects of the means for C1 and C2 to the mean of C3. Notice that C4 is weighted as 0. The final contrast compares the effects of C1 and C2.

Figure 13-10 shows how the first alternative is structured for the model. In the two tables, we see the three contrasts as a variable. Under the left column, we see the conditions that reflect in the dummy variables. Section 13.3 will explain by demonstration how the contrast variables are installed and interpreted in a model.

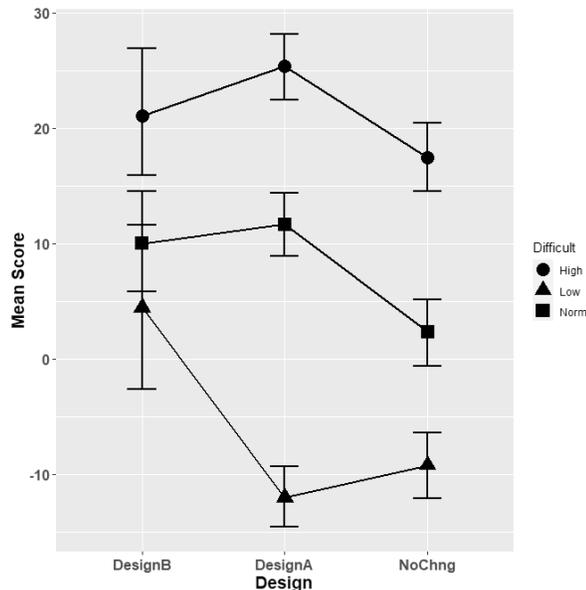
	Variable 1	Variable 2	Variable 3	Product
Group	Contrast 1	Contrast 2	Contrast 3	
Alternative 1				
C1	1	1	0	0
C2	1	-1	0	0
C3	-1	0	1	0
C4	-1	0	-1	0
Total	0	0	0	0
Alternative 2				
C1	1	1	1	1
C2	1	1	-1	-1
C3	1	-2	0	0
C4	-3	0	0	0
Total	0	0	0	0

Figure 13-10: Contrast coded as dummy variable.

We can also see something new to the explanation. The columns are totals and the final column is the products of the weights. Moreover, the products column is totaled.

```
#INSPECT FOR INTERACTION
DesignDiffInt<- ggplot(LongBestFactRepeat, aes(Design, Score, shape=
  Difficult))
DesignDiffInt + stat_summary(fun = mean, geom = "point", size=5) +
  stat_summary(fun = mean, geom = "line", aes(group= Difficult), lwd=1) +
  stat_summary(fun.data = mean_cl_boot, geom = "errorbar", width = 0.2,
  lwd=1) +
  labs(x = "Design", y= "Mean Score", colour= "Difficulty") +
  theme(plot.title=element_text(size=12,face="bold"),
  axis.text=element_text(size=12,face="bold"),
  axis.title=element_text(size=14,face="bold"))
```

In the output, we see the interplay of the two factorial variables. If it were the case that the plots were parallel, we would conclude there are no interactions. However, within the sector DesignB-DesignA, the low condition is an interaction with the average of high and normal difficulty.



Output 13-39: Contrasts for design and difficulty.

With respect to what the main effects tell us, there are two significant differences to be seen in the output. First is the significant difference between design and no change at all levels of difficulty. Second is the significant difference between high versus normal at all designs. To close an earlier point, notice that neither are components of the significant interaction.

Happily, when the dataset was made long, the resulting Groups variable made it possible to conduct post hoc analysis. The variable allows us to convert our two factorial model to a one-way repeated measures model. The following block of code will give us the table of pair-wise comparison of means and confidence intervals using the Tukey method. It is left to the reader to call up and inspect the output.

```
#POST HOC OF THE MODEL
#CREATE MODEL WITH GROUPS AS FACTORIAL VARIABLE
GrpSBestFactRepeatModel<-lme(Score~Groups, random=~1|Craft/Groups,
  data=LongBestFactRepeat, method='ML')
#GENERATE POST HOC ON MODEL
postHocsFactRepeat<- glht(GrpSBestFactRepeatModel,
  linfct=mcp(Groups='Tukey'))
summary(postHocsFactRepeat)
confint(postHocsFactRepeat)
```

The first action of the code is to create a multilevel model with the Groups variable and Craft as the random argument. Next, the model is an argument to the `glht` function with Tukey as the chosen method. The `summary` function returns a table of 36 ($n*(n-1)/2$) unique pair-wise combinations and for each the p-value to indicate significance. Finally, the `confint` function provides the confidence interval to the differences between the averages of the conditions of each pair.

The demonstration has shown how to compare two factorials with a limited number of conditions. We can see from Output 13-39 how to visualize the contrasts as main effects and interactions. If we had a third factorial variable, we would have been forced to do the exercise of visualizing contrasts as three charts of two factorials each. Furthermore, the permutations of pairwise comparison explodes in count. In other words, visual analysis is beyond practical.

That is the beauty of the multilevel repeated factorial model. We can use the returned outputs to arrive at conclusions without being dependent on visuals. As we have seen, the output of the `lme` model identified the interactions of interest we subsequently visualized. If there had not been full or partial interactions, the output would have told us which main effects, not engaged in an interaction, are significantly different.

A note. To this point, we have always identified a robust model to the parameterized model. However, currently there are none for factorial repeated measures. That is a virtue of using the multilevel model over an ANOVA model capable of repeated measures: the `eZANOVA` function. An ANOVA model requires esoteric assumptions to the dataset for validity which are easily violated. It follows that the unavailability of a robust model flows to mixed measures models.

13.3.7. Mixed Models

Until now, we have presented and explained by demonstration either independent or repeated participant analysis. However, it is not an either-or choice.

We can build models when the relationship of factorial conditions to participant is independent with respect to some variables and repeated for others. For example, the participants in the previous demonstration were repeated with respect to design and difficulty. Each participant was subjected to all conditions. We could add craft type as an independent variable for which some of the unexplained variance to the differences in outcome resides in the craft type rather than participant.

We use the `lme` function that was introduced and demonstrated in the previous sections to explain repeated measures models. Its syntax was as follows:

```
Model<- lme(outcome ~ predictor(s), random = random effects, data =
  dataframe, method = "ML")
```

The only difference from the previous sections is that we will add one or more independent predictor variables along with repeated measures to the model argument. The independent predictor is not included in the `random=` argument of the model.

Otherwise, our actions are the same. After preparing our dataset, we build our contrasts for the factorial variables. Thence, as demonstrated in the previous section, we would set the `random=` argument and incrementally build the model one predictor at a time. Finally, we would employ the `anova` function to determine the best fit.

Because the process to a mixed model is the same as for previous sections, this chapter will not demonstrate a case. If the reader wishes to follow a demonstration, one is presented in Chapter 14 of Field.

It should strike us that the multilevel model, as an independent regression extended to include the repeated measures, is very reflective of operational reality. Here we have cast the method as the means to analyze the effects of strategies and situational contexts.

Multilevel models are also powerful in regression to determine the true relationship of predictor variables to outcome. The difference is that multilevel regression tests as dependent variable when all or some of the unexplained variance to relationships resides within variables such as plant, cost center, etc.

The reader is encouraged to read the text, *Multilevel Modeling Using R*, by Finch, Bolin and Kelly. What has been explained in these last three sections will be a leg up to what is explained in the text.

Chapter 14

Recover Lost Classifications

Inherent to the data of any operation are the many classifications captured in its process management systems. They are the categorical variables in our CMMS and other systems. If we comply with our operational procedures, the consequent quality of all classifications is almost unavoidably perfect. In turn, we set off a virtuous cycle of data-driven analysis, making and execution of strategic, tactical and real-time decisions.

The cold reality is that compliance is problematic for classifications that are not automatic within a system. Especially for those that entail some degree of still less than fully informed conclusions. The classifications that are most quick to mind are failure modes and codes. Any degree of sloppy misclassification is doubly problematic because so much of reliability engineering depends upon analytics to which the classifications are the essential feedstock data.

How many times have we gathered in a discussion of methods in asset reliability and maintenance strategy? How many times has it been shoved in our face that we have not accurately captured failure history? How many times do we abandon the idea at hand to improve plant performance because we assume the necessary history has been forever lost to us?

What has been lost can now be found. There are data analytic methods to recover the classifications. This chapter will present the de facto two favorite and most practical methods. They are logistic regression and naïve Bayes.

Of course, there are others. They include K-NN, decision trees, 1R algorithm and K-means. Readers who wish to explore them and consider their possibilities can find them well explained and demonstrated in the book, *Machine Learning with R*, by Brett Lantz.

For the two methods, let's set a comparative point of reference. We all know of linear regression. We determine which variables are predictive of a numeric outcome. We use the learned model to predict outcomes with respect to a continuous variable such as dollars and productivity. In contrast, logistic regression and naïve Bayes predict classifications as outcomes such as failure modes and codes.

Then there is the contrast between logistic regression and naïve Bayes. Logistic regression works with all types of structured variables. Naïve Bayes works with free text variables. Thus, both are classifiers. Between them, they allow us to work with the full range of variables in our super table. We can also use one model to strengthen the other.

The chapter will begin with laying out a generalized procedure to recover lost classifications as explained in Section 14.1. Section 14.2 will explain by demonstration the method of recovering classifications with logistic regression. Section 14.3 do the same with respect to the naïve Bayes.

14.1. Generalized Procedure

Imagine for every work order pertaining to hundreds of pump assemblies in a refinery over the last ten years, we are seeking to understand the occurrence of a particular failure mode. No problem, right? The CMMS contains a classification of failure modes by assembly, including pump assemblies.

We pull up a table with, among all variables, the variables of work order number, assembly per the criterion of pump assemblies and failure mode. But upon inspection we have to ask ourselves what is the chance that 99 percent of all failure modes are “pump?”

First, pump as the component is not a failure mode. Second, a failure mode that is the only classification among multiple and likely possibilities is obviously bogus.

Our feathers drop. But then, an organization seeking to be excellent cannot be stopped. Right? Rather than be defeated, the director of reliability engineering directs an appropriately qualified engineer to inspect each work order to any pump assembly and classify the failure mode to each order.

As was demonstrated in Chapter 3, Super Tables from Operational Data, the engineer would build a super table. In it, are the many variables to the work orders, tasks and resources including craft names. With the super table, the engineer like a detective, classifies the failures.

Readers are thinking this is far too laborious to be practical. Most of us would rather get our tails kicked in the parking lot than be given the task. This is the biggest reason why failure history will remain lost if we cannot come up with a better way.

We cannot completely remove the engineer from the task with machine learning and artificial intelligence. However, with logistic regression and naïve Bayes probabilities, we can reduce the burden to reasonable. We can reduce the task of classification by human intelligence to a relatively few orders and use analytics to classify the many upon the few.

The procedure can be generalized as the following steps:

- Establish classification strategy upon analytic purpose.
- Establish the window of history.
- Build a super table to window.
- Select initial random sample.
- Classify small set of the initial sample by human intelligence.

- Train and evaluate model on the human-predicted classifications.
- Select and subject new random sample set to the model and confirm classifications.
- Append the latest predicted batch to the collective batch of all orders with confirmed classifications.
- Repeat the previous three steps with the expanding table of correctly classified orders.

The next two sections will explain the analytics to the procedure: logistic regression and naïve Bayes. The remainder of this section will explain the steps to the procedure.

Establish classification strategy upon analytic purpose. There is a purpose for the classifications. For example, we may want to do survival and hazard analysis of components to the pump assemblies with respect to an especially significant failure mode. We may be motivated because of the ramifications of the failure to some dimension of plant's performance.

Establish the window of history. Recall that we do not always need the history of an entire population. We may only need a statistical sample. It depends upon the purpose of the study. Therefore, as explained in Section 13.2.1 we establish a window of age and calendar.

Build a super table to match the window. The super table will combine in a single dataset, every variable to each order that will help the engineer to make classifying decisions. New indicative variables will likely be created to upon the system variables. In fact, with the final super table, the expert will likely be able classify many of the orders because criteria to some variables will allow the engineer to isolate or subset the orders on an obvious mode.

Select initial random sample. Depending upon age and calendar, the super table may consist of thousands of orders. This is by virtue of the window. Our next step is to extract a sample of orders randomly from the total set of the window. We want a sample size that the inspecting expert can pass through in less than an hour. The orders must be random so that the subsequent stages will be unbiased history.

Classify a small initial sample upon human intelligence. With the initial randomly selected orders from the full super table, the expert can scan through orders and make classifying decisions. The decisions are joined to be a variable to the sample super table. Now we have a super table that newly contains the orders classified by the mode or modes of interest to us.

Train and evaluate a model on human-predicted classifications. The table of classified cases are now our dataset with which to train the logistic and naïve Bayes models to classify orders by artificial intelligence. Once trained, the model is evaluated for its ability to accurately classify.

Select and subject new random sample to the model and confirm classifications. We return to the window super table and randomly select a new batch of orders that were not included in the previous batch. We submit them to the trained model and attach the surmised classifications to the extracted super table. Thence, the expert will scan for and correct the few orders that may be Type I and II errors. The model has done most of the work, the expert can concentrate on finding those in error.

Append latest batch to all orders with confirmed classifications. We now have the initial and subsequent sample of orders as a block of correctly classified orders. We append each new block to the collective previous blocks such that we have a growing unbiased collection of correctly classified failure modes.

Repeat the previous three steps with the expanding table of correctly classified orders. The idea is to refine the model upon our correctly finalized classification and submit a new sample set to the upgraded classifier. The cycle is repeated until there are an adequate number of observation to conduct the analysis for which the classifications are feedstock.

The next two sections will explore the analytics to the procedure. The demonstration will be as if we are only seeking two classifications, the one of interest and other. If our purpose were to conduct survival hazard analysis, including Weibull, the “exit” event is the classification of interest and the “censored” event is the other.

However, the procedure can be easily expanded to three or more classifications. How will be noted as the models for classification are explained.

14.2. Logistic Regression as Classifier

For all the hype about machine learning and artificial intelligence, it is most frequently a simple two-step dance. Here a model is learning to classify a dataset and then classifying unclassified records as the model has learned to do.

The most utilized classifier is logistic regression. Although powerful, it is hardly a exotic as the media hype want us to imagine. As they say in Hollywood, “Nobody is interested in just an okay story.”

Just as for linear regression, the strength of relationships between predictor variables and an outcome variable are evaluated: machine learning. The learned model is used to predict outcome: artificial intelligence. However, unlike linear regression, we are seeking

strength of relationship to classifications as the outcome rather than continuous numeric outcomes between positive and negative infinity.

This section will describe logistic regression to the depth that the reader needs to know, rather than everything there is to know. We will start by contrasting the linking and systematic components of linear and logistic regression. Then we will get our arms around the solution mathematics and interpretation of logistic regression. Thence, we will expand upon the explanation of logistic regression in the context of a demonstration.

14.2.1. Concept of Logistic Regression

Chapter 7 explained and demonstrated linear regression. The explanation of logistic regression is largely a variation on linear regression.

We can see fundamental differences in Figure 14-1. A linear regression is just that, linear. It predicts a continuous outcome along a straight line with points that can range from positive to negative infinity.

Contrast that to the logistic regression curves and axes. The curve is obviously not linear. The vertical axis ranges from 0 to 1 to reflect the probability of an outcome. The outcomes are categorical variable such as failure modes and codes.

Also notice the comparative observation of the two plots. For the linear regression, the observations are distributed about the fitted line with an even spread of dispersion.

For the logistic regression, the observations appear either at the top or bottom of the frame with horizontal overlap between the extremes. Speaking mathematically, the fitted curve depicts the instantaneous probability of the top observation divided by the sum of the top and bottom observations.

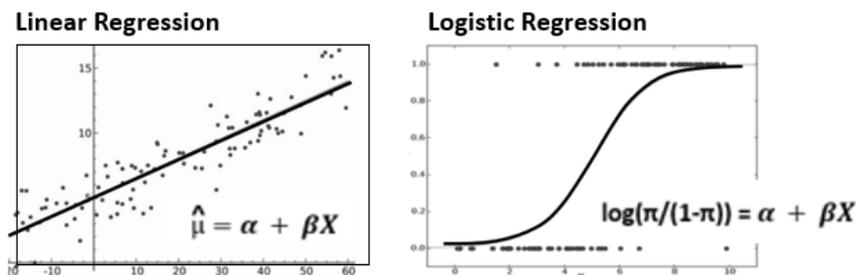


Figure 14-1: Comparative perspectives of linear and logistic regression.

The equations to the respective models are inserted in the charts. The notable difference is what is called the linking functions that are to the left of the equality operators. For linear regression, it is called the identity: $\hat{\mu}$. For logistic regression it is called the log odds: $\log(\pi/(1-\pi))$. The systematic components to the right of the equality operator are identical.

by which to get maximal value from the tremendous labor and expense of having come to own such an expensive dataset.

It has been typical to evaluate the findings without logistic regression. Instead, the findings are evaluated as yes divided by the quantity of yes plus no. We saw earlier that this is the null model, one without predictor variables and, thus, the weakest of all possibilities.

If the dataset is subject to logistic regression, we can inspect probability of “found working” in the context of the relationship to almost limitless operational variables. The variable can be ones captured in the field or extracted from operating systems and joined to the field data.

If the wrench study has not yet been designed, logistic regression gives us options. There is nothing mandating that the study be binary. We could design a multinomial study. The found condition could be more than merely work and not working.

14.3. Naïve Bayes Probability as Classifier

We can tease many classifications from the structured data of our super table. Some can be extracted by merely devising new categorical variables with existing ones. An example was given in Chapter 7 when it was demonstrated how to classify orders by lead craft when the system did not provide the classification.

Other classifications can be made with models. The previous section demonstrated the use of logistic regression as a classifier. It was great but will only work with the structured data in the super table.

The problem is that there are often unstructured free-text variables in the dataset. An example is the work order description variable in a CMMS.

This is noteworthy because words in free-text variables can be indicative of classifications such as failure modes and codes. We can extract the connection with the model type upon what is called naïve Bayes probability.

14.3.1. Concept of Naïve Bayes

The idea of a model built on naïve Bayes is to determine the probability of a classification (e.g., failure mode or code) in union with the probability of a word appearing in the classification out of all classified cases in which the word occurred. Rather than a single word, the probability is generated upon the large body of words found collectively in the free-text variable.

This chapter will not expand upon the explanation of naïve Bayes probability. The reader is referred to Chapter 4 of the book, *Machine Learning With R*, by Lantz for a deeper but practicable grasp of naïve Bayes probability.

The models throughout the book have been to fit a model to a dataset. How well the model would have predicted the data it is trained upon. In contrast, naïve Bayes makes its determination upon probability of occurrence.

Naïve Bayes is the most common of probability-based methods. That is especially so for classification with free text. So much so that it has become the de facto standard for working with free text.

It is called “naïve” because it makes two naïve assumptions. First, it assumes that all words are equally important in the determined probability. Second, it assumes that all words in a string of text are independent. Of course, both are very naïve. However, all is forgiven because the method still performs well.

It is also forgiven because it has other advantages. One is that it requires relatively few cases as an example set and still performs well with large datasets. Another is that it does well with noisy and missing data.

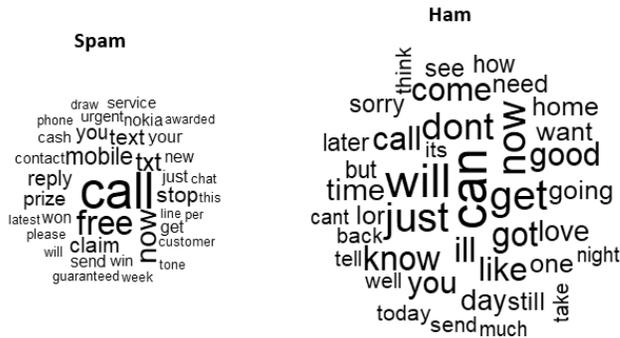
It has other limitations beside naivety. One is that it is not ideal for numeric datasets. However, that is not an issue to us because, rather than numeric variables, we seek a tool to make classifications on free-text data.

Another limitation is that its estimated probabilities are less reliable than the predicted named classes. Its advocates counter with the point that getting good classifications is the whole point. In other words, compared to logistic regression, we receive answers but without any solid measure of conviction.

14.3.2. Case, Dataset and Exploration

The case will classify text messages as ham or spam using a learned model. The idea is to process the messages as a block of text that has been cleansed and standardized with respect to upper and lower case, punctuation, meaningless words, numbers, white space and stem words. A dataset of messages is a good example. This is because text messages are similar to what we see in free text to maintenance operations. In both there are many standard and eccentric abbreviations and insider terms.

The messages in the dataset have been inspected and labeled (classified) as Ham and Spam for a training and testing dataset. This follows the procedure described in Section 14.2. A sample of the data is classified by an expert in preparation for subjecting it to a naïve Bayes algorithm for machine learning.



Output 14-17: Word clouds for ham and spam.

14.3.5. Create the Document Term Matrix

Now it is necessary to split each message into its individual components. Imagine each message as a row and each word in the bag of words as a column. This is depicted in the matrix of Figure 14-5. It is called a document term matrix (DTM).

Message	Word1	Word2	Word3	Word4	Word5
1	0	0	0	0	0
2	0	0	0	2	0
3	1	0	0	0	0
4	0	0	1	0	0
5	0	0	0	0	0
6	0	0	0	0	1

Figure 14-5: Document Term Matrix

As one can imagine, the matrix will be massive and sparse. In the DTM of our dataset, there will be thousands of rows and columns. In the cells containing “0,” a word as term does not appear in the subject message.

Each message must contain at least one word. However, the chance of any one word appearing in a given message is small. It is not surprising that the DTM is called a sparse matrix.

The matrix for our corpus is created with the following code. If curious, the reader can see the statistics of the matrix with the `msg_dtm` command.

```
#CREATE DOCUMENT TERM MATRIX
msg_dtm<- DocumentTermMatrix(msg_corpus_clean)
msg_dtm
```

Output 14-18 is the output of the code. Notice the spread of ham and spam for the respective datasets. They are very close and that is good.

```
> prop.table(table(msg_train_labels))
msg_train_labels
      ham      spam
0.8647158 0.1352842
>
> prop.table(table(msg_test_labels))
msg_test_labels
      ham      spam
0.8683453 0.1316547
```

Output 14-18: Comparison of distribution of Type to the datasets.

14.3.7. Create Indicators of Frequent Words

We need to transform the sparse matrix to a data structure with which to train the naïve Bayes classifier. The DTM has over 6,000 words that appear at least once in a message. We do not need to be an analytical genius to guess that not all are important to accurate classifications.

Instead, we can set a threshold for a number of appearances in the DTM. We use the `findFreqTerms` function in the following code. Notice that the threshold is set to 5. The code retains the remaining 1,183 terms in the object `msg_freq_words`.

The `str` function, if run, would report that there are 1,136 words down from over 6,000 words. If the reader wishes to inspect the list of words, she would call up `msg_freq_words`.

```
#ATTACH FREQ WORDS TO THE DTM
msg_freq_words<- findFreqTerms(msg_dtm_train, 5)
str(msg_freq_words)
```

We next filter the respective DTMs upon our list of frequent words. In the following block of code, notice the list of frequent words within the square brackets as the filtering mechanism.

```
#REDUCE DTM TO ONLY MSG_FREQ_WORDS
msg_dtm_freq_train<- msg_dtm_train[,msg_freq_words]
msg_dtm_freq_test<- msg_dtm_test[,msg_freq_words]
```

The naïve Bayes is typically framed on categorical rather than numeric variables. However, the DTM is a table of numeric word counts to each message. We need to convert

```
#EVALUATING MODEL PERFORMANCE
#NOTE TAKES SOME TIME, BE PATIENT
msg_test_pred<- predict(msg_classifier, msg_test)
```

As noted in the note to the code, the model will require some time, several minutes. Your computer is not broken, it only needs a moment of grace to process over a million cells in a complex calculation of probability.

In the code, notice the `predict` function. Its first argument is the previously trained model. With the second argument we are giving it our withheld test data, absent of preclassifications. The ultimate table of classifications estimated by artificial intelligence, is assigned to the object, `msg_test_pred`.

Next, we want to compare the predicted classifications to the pre-classifications. The code below with the `crossTable` function of the `gmodel` package serves the purpose.

```
#TABLE TO EVALUATE TRAINED MODEL
crossTable(msg_test_pred, msg_test_labels,
  prop.chisq=FALSE, prop.t=FALSE, prop.r=FALSE,
  dnn=c("predicted", "actual"))
```

In the code, that returns the table of Output 14-19, we can see that we are comparing the prediction, `msg_test_pred` and the `msg_test_labels`. The `dnn=` argument attaches axis titles to the table. The legend to the table also comes with what is returned.

```
Cell Contents
|-----|
|                N |
|      N / Col Total |
|-----|

Total Observations in Table: 1390

  predicted | actual
-----|-----|-----|
      ham |      ham |      spam | Row Total |
-----|-----|-----|
      ham |      1203 |          20 |      1223 |
      |      0.997 |      0.109 |      |
-----|-----|-----|
      spam |           4 |         163 |         167 |
      |      0.003 |      0.891 |      |
-----|-----|-----|
Column Total |      1207 |         183 |      1390 |
      |      0.868 |      0.132 |      |
-----|-----|-----|
```

Output 14-19: Analysis table for false positive and negative accuracy.

The arguments, `prop.chisq=FALSE`, `prop.t=FALSE` and `prop.r=FALSE`, would otherwise add elements to the table. The reader can set them to true and see what happens

We can see the improvement. The false positives are reduced by half from 4 to 2. However, note that total misclassifications have increased from 24 to 25. We have gained “sensitivity” against false positive errors, but at the price of loss of “power” to avoid false negatives.

14.3.9. Classify Cases

The previous step demonstrated the training and testing of the naïve Bayesian model. However, the ultimate purpose is to present to the classifier a free-text dataset of cases for which the true classifications have been lost by misstatement, misselection or omission. The data can be from the CMMS and other systems. We may seek any classification including failure modes.

The free-text variable may be the description and notes to work orders. Furthermore, we can expand the variable to be the concatenation of multiple free-text variables from the same or multiple operating systems.

We will now demonstrate the procedure to predict (recover) true classification, pull them from the classifier, and return them to the super table as good data.

For demonstration purposes, we will imagine that the data, which was previously used as the test data set, is unclassified data to be subjected to the classifier. The following code will make the conversion to suit our demonstration.

```
#DEMO DATASET FROM PREPARED DTM
UnClassSet<- msg_test
head(UnClassSet)
```

The reader should not give any meaning to the code line other than to recognize that we are setting up a fictitious DTM to demonstrate the classifier in action. In real life, the DTM upon our unclassified dataset would have passed through the corpus, and standardization and cleansing stages until becoming a DTM.

The next block of code will submit the DTM, UnClassSet, to the classifier. The output is assigned to the data frame, predDf.

```
#MSG_CLASSIFIER2 PREDICTS CLASSIFICATIONS
msg_Unclass_pred<- predict(msg_classifier2, UnClassSet)
predDf<- as.data.frame(msg_Unclass_pred)
head(predDf)
```

Next, we want to prepare the messages from the raw dataset for being placed, side by side, with the predictions returned by the previous block of code. The action to extract them

uses the second argument to the square brackets to call for the text variable. Finally, they are assigned to the created data frame object, message.

```
#PLACE RAW MESSAGES IN DATA FRAME
message<- MsgRaw[4170:5559,2]
messDf<- as.data.frame(message)
head(messDf)
```

It is unimaginable that our cases would not come with a unique identifier variable such as work order number. Naturally, our dataset of messages does not include such a variable. The previous block of code is parallel to what we would do to bring the unique identifier variable to the final match up. The second argument in the squared brackets would identify the column in the raw dataset that is the identifier. Just as for message, the identifier would be assigned a name.

Next, we place the two variables side by side in the data frame object, Classifications. The first 20 cases are inspected with the head function. If an identifier variable had been present, it would have been included in the code to create the data frame.

```
#RETURN ESTIMATED CLASSIFICATION FOR EACH MESSAGE
Classifications<- data.frame(predDf, messDf)
head(Classifications, 20)
```

Assuming CMMS data, we would join the Classifications dataset to the super table using the work order number. We would likely pull the classifications and work numbers into the super table. We have only included the messages variable in the demonstration as proxy for an identifier variable.

Bibliography

- Field, Andy; Miles, Jeremy; Field, Zoe. *Discovering Statistics Using R*. Sage Publications 2012.
- Finch, Holmes; Bolin, Jocelyn; Kelly, Ken. *Multilevel Modeling Using R*. CRC Press 2014.
- Institute for Digital Research and Education. Binary Logistic Regression Example. UCLA. <https://stats.idre.ucla.edu/r/dae/logit-regression/>
- Institute for Digital Research and Education. Multinomial Logistic Regression Example. UCLA. <https://stats.idre.ucla.edu/r/dae/multinomial-logistic-regression/>
- Lantz, Brett, *Machine Learning with R: Expert techniques for predictive modeling*, Packt Publishing, 2015.

Index

- 1R algorithm, 437
- Accessible on demand, 225–30
- Accessible on demand defined, 222
- Achievable availability, 7, 353
- Administration of tables, 80–81
- Aggregation variables, 65–66
 - Expression and Where, 72, 252
 - Summary options, 66
- Akaike information criteria (AIC), 190
- Analytics, interpretation of
 - Akaike information criteria (AIC), 449
 - Akaike information criterion (AIC), 190
 - Centrality and spread, 120
 - Confidence limits, 99
 - Exponentially weighted moving average (EWMA).
 - See Time series, principles of; Exponentially weighted moving average (EWMA)
 - Hypothesis test, 98
 - Kurtosis, 121
 - Mean, 120
 - Mean absolute deviation from median, 121
 - Mean trimmed of extremes, 121
 - Median, 120
 - Min-max, 120
 - Pearson, 97
 - p-value, 99
 - Quartiles, 120
 - Range, 121
 - Skew, 121
 - Spearman, 97
 - Standard deviation, 121
 - Standard error to the mean, 121
- Analytics, types of, 95–100
 - ANCOVA. *See* ANCOVA, principles of; ANCOVA, templates and interpretation; and ANCOVA, R functions
 - ANOVA, factorial. *See* Factorial ANOVA, principles of; Factorial ANOVA, templates and interpretation; and Factorial ANOVA, R functions
 - ANOVA, one-way. *See* One-way ANOVA, principles of; One-way ANOVA, templates and interpretation; and One-way ANOVA, R functions
- ARIMA model. *See* Time series, principles of; ARIMA (Autoregressive integrated moving average)
- Correlation analysis
 - Between categorical and numeric variables, 132
 - Correlation coefficient (R value), 95
 - Correlation defined, 95
 - Regression Coefficient defined, 95
- Correlation compared to regression analysis, 103–8
- Exponentially weighted moving average (EWMA).
 - See Time series, principles of; Exponentially weighted moving average (EWMA)
- Factorial repeated measures. *See* Factorial repeated measures, principles of; Factorial repeated measures, templates and interpretation; and Factorial repeated measures, R functions
- General linear model (GLM). *See* general linear model
- Holt Winters. *See* Time series, principles of; Holt Winters
- Linear regression. *See* linear regression templates and interpretation
- Logistic regression. *See* Logistic regression, principles of; Logistic regression, templates and interpretation; and Logistic regression, R functions
- Missing data, 110–17
 - Steps to, 110
- Mixed model, 384
- Multilevel model, 384
- Naïve Bayes. *See* Naïve Bayes, principles of; Naïve Bayes, templates and interpretation; and Naïve Bayes regression, R functions
- One-way repeated measures. *See* One-way repeated measures, principles of; One-way repeated measures, templates and interpretation; and One-way repeated measures, R functions
- Partial correlation analysis, 100–103
 - Correlation (R), 100
 - Correlation squared (R²), 100
 - Misleading, as, 102

476 | Index

- Robust difference model, 384
- Robust linear regression, 107
- Survival hazard. *See* Duration analytics, principles of, Duration analytics, templates and interpretation, and Duration Analytics, R functions
- Time series, 241. *See* Time series, principles of, Time series, templates and interpretation, and Time series, R functions
- Two-mean t-test, single factor, two conditions. *See* Two-mean t-test, principles of, Two-mean t-test, templates and interpretation, and Two-mean t-test, R functions
- ANCOVA. *See* ANCOVA, principles of; ANCOVA, templates and interpretation; and ANCOVA, R functions
- ANCOVA, principles of, 381–82, 404–13
 - Continuous variable in ANCOVA, 381
 - Covariate, 381
 - Mean, se and confidence limits adjusted for covariate, 408
 - Unexplained variance, 381
- ANCOVA, R functions
 - Anova(), 408, 412
 - aov(), 407, 408, 412
 - effect(), 408
 - glht(), 411
- ANCOVA, templates and interpretation
 - Anova() to test for homogeneity, 412
 - aov() test for significant difference contrasts as main effects, 408
 - aov() to test covariate as independent, 407
 - Build contrast variables and inspect, 407
 - by() for descriptive statistics, 406
 - effect() to adjust mean, se and limits for covariate, 408
 - factor() to set variable base condition, 405
 - ggplot2 to plot points and covariate to score, 410
 - ggplot2 to plot variable condition means and limits, 419
 - ggplot2 to plot variate regression to condition points, 412
 - glht() to adjust limits with Tukey, 411
 - leveneTest() test for constant variance across conditions, 406
 - plot() of variance and normality to residuals, 411
 - summary.lm() test significant difference of means, contrasts as effects, adjusted for covariate, 409
 - summary.lm() to inspect contrasts for difference, 420
- ANOVA. *See* One-way ANOVA, principles of; One-way ANOVA, templates and interpretation; and One-way ANOVA, R functions
- ANOVA, factorial. *See* Factorial ANOVA
- Apparency insight, 40–41
 - Models, 40
- Append query, 50, 56
- ARIMA model. *See* Time series, principles of; ARIMA (Autogressive integrated moving average)
- Artificial intelligence, 438, 441
- Artificial intelligence (AI), 207–17
- Autocorrelation, 37
- Availability performance, 2–4
 - Constituent probability, as, 3
 - Contrast to OEE, 2
 - defined, 2
 - Depicted, 3
 - Equation, as, 3
 - Life curves, 4
 - Operational availability, 9–11
 - Cost effectiveness, 11–13
 - Factors of, 10–11
 - Probability, as, 2
 - Subtypes, as, 6–7
 - Achievable, 7
 - Equations, 7–8
 - Inherent, 6
 - Interrelationships between, 9
 - Operational, 7
- Backlog, workload, 249
- Bad data, imputation
 - Models for
 - Linear regression, 163
 - Logistic regression, 165
 - Multilevel (MLM), 167
 - Naïve Bayes, 165
 - Trees, 166
 - Trees, 163
 - Trees, 168
 - Translation tables, 164
- Bad data, rectification
 - Eliminate, 160
 - Impute estimate (ML and AI), 160
- Bad data, types
 - Inapparent without measurement, 159
 - Misformatted and mis misspelled, 159
 - Missing, 159
 - Outliers, Leveragers and influencers, 161
 - Unexpected, 159
- Base condition, set, 395

- Base model, 380
- Bayes probability, 438
- Benjamini-Hochberg adjustment, 385, 402
- Bibliography, 14, 81, 108, 158, 169, 219, 236, 278, 318, 351, 375, 436, 473
- Binary logistic equation, 442
- Blanchard, Benjamin S, 14
- Bolin, Jocelyn, 443, 474
- Bonferroni adjustment, 385, 402
- Budget and variance, principles and data methodology, 240–42, 243–45
 - Actuals, table of, 253–55
 - Budget, contrast one- and two-dimensional, 242
 - Case as progression, 247–49
 - Flowchart, 248
 - Variance Output reports, 248
 - Workload from history, 250
 - Craft capacity as workforce, 270–78
 - Dual dimensional
 - Activity (AKA, workload), 241
 - Factor of cost, 241
 - Mission, structure and derivation, 238–40
 - Derivation, 239
 - Mission, 238
 - Structure, 238
 - Objectives, three, 237
 - Template tables, queries and pivots, 249–53
 - Variance supertable and pivot, 255–57
 - Variance, contrast one- and two-dimensional, 239
 - Variance, two-dimensional computations, 243, 257
- Budget-based schedule cycle. *See* Schedule cycle, stages to
- Budget-based schedule measures and methods. *See* Schedule cycle, measures of
- Budget-based schedule tables. *See* Schedule cycle, tables to
- Budgeting, types of
 - Predictive-based, 239
 - Zero-based, 239
- Categorical variables, interpret, 188
- Censored event, 358, 360
- Centality and spread, 120
- Classification insight, 39–40
 - Models, 39
- Classification, Procedure to recover, 438–40
 - Artificial intelligence, 438
 - Machine learning, 438
 - X, 438
- Classifiers for categories
 - Aggregation, upon, 72–78
 - Cleanse data, steps to, 207
 - Concatenation operator, 58
 - Confidence interval to regression coefficient, 189
 - Confidence limits, 99
 - Contrasts as effects, 386–89
 - Contrast variables, build, 387
 - Family error, 386
 - Cooks distance, test of cases, 209
 - Corpus, text dataset, 462
 - Correlation (R), 100
 - Correlation analysis
 - Correlation defined, 95
 - Regression Coefficient defined, 95
 - Correlation coefficient (R value), 95
 - Correlation compared to regression analysis, 103–8
 - Correlation defined, 95
 - Correlation squared (R²), 100
 - Criteria row, Access code
 - Conditional expressions, 60
 - Contains, 59
 - Dates, working with, 60
 - Equals, 59
 - Numeric, for, 60
 - OR and AND, 58
 - Text strings, for, 59
 - Critical mass and grass roots, 16–19
 - Characteristics, qualifying, 17
 - Critical mass, 16
 - Grass root, 16
 - Triad of software, 22
 - Cross correlation, 37
 - Crosstab query, 50, 233
 - Cumulative hazard, 358
 - Cumulative hazard function, 361, 364
 - Current to capture, 225–30
 - Current to capture defined, 221
 - Data analytics, principles of
 - Multiple predictor model, 370–73
 - Parametric model
 - Weibull distribution, eras of, 369
 - Data and report as layers, 223
 - First layer, 224
 - Second layer, 224
 - Data and variance, principles and data methodology
 - Reports and interaction analysis, 258–62, 262–70
 - Data driven, 15–16
 - depicted, 16
 - Essential definitions, 19–22
 - Alogrithm, 21

478 | Index

- Artificial intelligence, 21
- Data and big data, 20
- Machine learning, 21
- Insight deliverables. *See* Insight deliverables
- Data frame, 98
- Data table compared to Spreadsheet, 223
- Data, entirety of
 - Accessible on demand, 230–36
 - Relational database, 230
 - Accessible on demand defined, 222
 - Current to capture, 225–30
 - Current to capture defined, 221
 - Data and report as layers, 223
 - First layer, 224
 - Second layer, 224
 - Fused data and report, 223
 - Layered structure, 222–25
 - Layered structure defined, 221
 - Spreadsheet compared to data table, 223
 - Table as format, 22, 222
- Data, reshape, 428
- Decision tree model, 40
- Decision trees, 437
- Delete query, 50
- Design grid rows, Access, 56–61, 56–61, 56
 - Criteria. *See* Criteria row code, Access
 - Field. *See* Field row code, Access
 - Show, 57
 - Sort, 57
 - Table, 57
 - Total, 62, 66
 - Summary options, 62
- Deviance as measure of fit, 448
- DFBeta, 451
- Difference analytics, 383–84, *See also* one-way
 - repeated and factorial repeated measures and respective principles of and templates and interpretation
- Difference insight, 35–36
 - Family error, 36
 - Models, 36
- Document term matrix, 466
- Dual dimensional budget and variance, 240
- Dummy variables, 182–85
- Dunnett adjustment, 385, 403
- Duration analytics, principles of
 - Baseline model, defined, 366
 - Empirical construct
 - Censored event, 360
 - Cumulative hazard function, 361
 - Hazard function, 361
 - Kaplan-Meier estimator, 360
 - Nelson-Aalen estimator, 360
 - Survival function, 362
 - Empirical construct of, 360–62
 - Empirical model, 366–68
 - Hazard defined, 357, 359
 - Left truncated, 359
 - Life data, event and window, 358–60
 - Censored event, 358
 - Cumulative hazard, 358
 - Event, defined, 358
 - Life as hazard, 358
 - Mathematical construct, 362–64
 - Beta parameter, slope, 362
 - Cumulative hazard function (H(t)), 364
 - Eta parameter, scale, 362
 - Failure function (F(t)), 363
 - Hazard function (h(t)), 364
 - Mean time to specified event (MTTF), 364
 - Survival function (R(t)), 363
 - Median rank regression method (MRR), 374
 - Most likely estimate method (MLE), 374
 - Parametric model, 368–70
 - Survival defined, 355
 - Weibull plot, 373–75
- Duration analytics, R functions
 - coxph(), 366
 - MLEW2p(), 374
 - MRRW2p(), 374
 - phreg(), 368, 370, 372
 - strata(), 370
 - Surv(), 366, 368, 370, 372
 - survfit(), 366
- Duration analytics, templates and interpretation
 - coxph() for empirical model of data, 366
 - phreg() baseline parametric model, 368
 - phreg() parametric model with one or more variables, 372
 - phreg() parametric model with strata variable, 370
 - Weibull model with Median rank regression method (MRR), 374
 - Weibull Model with most likely estimate method (MLE), 374
 - Weibull prepare data for model, 373
- Duration insight, 38–39
 - Models, 39
 - Plots to insight
 - Cumulative hazard, 38
 - Hazard, 38

- survival, 38
- Durbin Watson test, 200
- Entirety of data. *See* Data, entirety of
- Event history analysis, 38
- Exponentially weighted moving average (EWMA). *See* Time series, principles of; Exponentially weighted moving average (EWMA)
- Factorial ANOVA, principles of, 382–83, 413–21
 - Post hoc, method for, 421
 - Visualize variance and normality to convert to one-way ANOVA, 421
- Factorial ANOVA, R functions
 - Anova
 - (), 418
 - factor(), 414
 - summary.lm(), 420
- Factorial ANOVA, templates and interpretation
 - aov() fit and test for presence of differences, 418
 - by() descriptive statistics, individual and interactive, 416
 - Contrast() to build contrast variables and inspect, 418
 - factor() t set base conditions to variables, 414
 - ggplot2 to visualize relationship of factorial variables, 415
 - leveneTest() for constant variance across conditions, 417
 - plot() to visualize variance and normalcy to factorial converted to one-way ANOVA, 421
- Factorial repeated measures, principles of, 427–34
 - Reshape data, 428
 - Robust model, 434
- Factorial repeated measures, R functions
 - anova(), 431
 - lme(), 431
- Factorial repeated measures, templates and interpretation
 - Build contrast variables and inspect, 430
 - by() for descriptive statistics, 430
 - ggplot2 to compare conditions and interaction, 433
 - glht() post hoc of multilevel model with Tukey, 434
 - lme() one variable subjected to multiple measures and progressive predictor variables, 431
 - melt() make wide data long, 429
- Failure function, 363
- Family error, 36, 379, 385, 386
- Field row, Access code
 - Aliases, 69, 78
 - Calculated, 59
 - IIF, 60, 71
 - Switch, 60
 - Combined with concatenation, 58
 - Concatenation operator, 58
 - Names common to multiple subtables, 58
- Field, Andy, 473
- Finch, Holmes, 443, 473, 474
- F-statistic, 187
- Fused data and report, 223
- General linear model (GLM), 172–74
 - Defined, 172
 - Dummy variables, 182–85
 - Interaction variable, 173
 - Main effects, 173
 - Null model, 172
 - Select variables, 185–93
 - Three components
 - Linking function and solution, 174–77
 - Three regressions, 171
 - Transformation of variables, 173, 193–94
- ggplot2 geoms
 - geom_bar(), 149, 150
 - geom_boxplot(), 141, 143, 157
 - geom_contour(), 138
 - geom_count(), 138, 139
 - geom_density(), 147, 205, 206
 - geom_freqpoly(), 144, 146
 - geom_hex(), 138, 139
 - geom_histogram(), 144, 145, 147, 204
 - geom_jitter(), 138, 141, 142
 - geom_line(), 152, 156, 157
 - geom_path(), 154
 - geom_point(), 135, 137, 139, 141, 142, 143, 154, 157, 194, 195, 198, 199, 200, 202, 209, 212
 - geom_raster(), 138
 - geom_smooth(), 135, 137, 156, 194, 195, 198, 199, 200, 202, 209, 212
 - geom_violin(), 142
- ggplot2 templates
 - Barchart as mean, 150
 - Barchart with dodge, 149
 - Boxplot with bins for continuous variable, 143
 - Boxplot with geom_jitter(), 141
 - Comparison of confidence limits of mean effects, 396
 - Density plot with geom_density, 205, 206
 - Density with geom_histogram, 204
 - ggplot2 with geom_boxplot, geom_point for mean and points, and geom_jitter, 391

480 | Index

- Histogram and polygon overlay with
 - `geom_histogram` and `geom_freqpoly`, 144
- Histogram as density, 147
- Histogram subsetted categories with
 - `geom_histogram`, 145
- Line chart for groups with line fit to overall, 156
- Line chart with boxplot and mean, 157
- Line chart with `geom_line`, 152
- Path chart over `geom_point` with `geom_path`, 154
- Polygon as density, 147
- Polygon with categories overlaid by
 - `geom_freqpoly`, 146
- Q-Q plot, 122, 127, 204
- Q-Q plot with `facet_grid()`, 129
- Q-Q plot with `facet_wrap`, 125, 128
- Scatter chart with `geom_point`, 154
- Scatter plot of multiple datasets with `geom_point` and `geom_smooth`, 209
- Scatter plot with `geom_count()` for overplotting, 139
- Scatter plot with `geom_hex()` for overplotting, 139
- Scatter plot with `geom_point`, `geom_smooth` and `facet_grid`, 194, 195, 198, 199, 200, 202
- Scatter plot with `geom_point`, `geom_smooth` and `facet-wrap`, 135
- Scatter plot with `geom_smooth()`, 134
- Scatter plot with multiple datasets, `geom_point` and `geom_smooth`, 137, 212
- Violin chart with `geom_jitter` for overlapping, 142
- Hat test of cases, 208
- Hazard function, 361, 364
- Heap, Howard F, 14
- Holt Winters. *See* Time series, principles of; Holt Winters
- Independence of errors, 200
- Industrial internet of things (IIoT), 18
 - Feasibility of, 19
- Inherent availability, 6, 353
- Inner join, 53
- Insight deliverables, 31–41
 - Know-thy-data, 32–33
 - Modeled, 33–41
 - Apparency. *See* Apparency insight
 - Classification. *See* Classification insight
 - Difference. *See* Difference insight
 - Duration. *See* Duration insight
 - Relationship. *See* Relationship insight
 - Time series. *See* Time series insight
- Recountive, 32
- System reports, 31
- Institute for Digital Research and education, 474
- Interaction of variables, 197
- JoinProperties window, 54
- Joins complex
 - Concatenated join, 69, 78, 251
 - Multiple variables join, 69, 78
- Joins in query
 - Join properties window, 54
 - Type
 - Inner, 53
 - Left, 53
 - Outer, 54
 - Right, 53
- Jones, James V, 14
- Kaplan-Mier estimator, 360
- Kelly, Ken, 443, 474
- K-means, 437
- K-Means model, 40
- K-NN, 437
- Know-thy-data, 90–95
 - Layered charting, 94
 - Normal distribution, 92
 - Q-Q graphic, 93
 - Normal distribution, tests of
 - Shapiro-Wilk test, 93
 - Pairs panel, 92
- Kurtosis, 121
- Lamb, Richard G, 14
- Lantz, Brett, 437, 474
- Layered charting, 29–31, 94
 - Defined, 29
 - Traditional, compared to, 29, 32
- Layered structure, 222–25
- Layered structure defined, 221
- Left join, 53
- Left truncated, 359
- Leverage, 451
- Life curves, 4
- Linear regression, 34, 163
- Linear regression templates and interpretation
 - `lm()` with main effect and interaction variables, 212
 - `lm()` with main effect and transformed variables, 195
 - `lm()` with main effect variables, 104, 189, 191
 - `lm()` with main effect, interaction and transformed variables, 215, 217
 - p-value, 187
 - Residual standard error, 190, 192, 197
 - Test for generalization

- Variance inflation factor (VIF), 201
- Linear regression, principles of
 - Akaike information criterion (AIC), 190
 - Artificial intelligence (AI), 207–17
 - Categorical variables, interpret, 188
 - Cleanse data, steps to, 207
 - Confidence interval to regression coefficient, 189
 - Cooks distance, test of cases, 208, 209
 - Dummy variables, 182–85
 - F-statistic, 187
 - Hat test of cases, 208
 - Imputation by estimate, 213
 - Interaction of variables, 197
 - Machine learning, 177–78
 - Predict individual cases, 217–18
 - Residual standard error, 187
 - R-squared, 187, 192
 - Select variables, 185–93
 - Standardized effect, 185
 - Standardized residuals, 194
 - Test for generalization, 200–207
 - Durbin Watson test, 200
 - Independence of errors, 200
 - Normal distribution of residuals, 203
 - Transformation of variables, 193–94
- Linear regression, templates and interpretation
 - cooks.distance() to each case in data set, 209
 - lm() with main effect variables, 185, 192
 - lm() with main effect, interaction and transformed variables, 196
 - lm()with main effect variables, 187
- Linking function, 441, 442
- Logistic regression, 34, 165, 438
- Logistic regression, principles of, 457–59
 - Artificial intelligence, 441
 - Binary logistic equation, 442
 - Build and interpret, 445–49
 - Classification as probability, 450–51
 - Classify cases, 452–54
 - Confidence limits to probability, computation, 453
 - Deviance as measure of fit, 448
 - DFBeta, 451
 - Leverage, 451
 - Linearity, method of, 456
 - Linearity, test for, 454–57
 - Linking function, 441
 - Logistic regression as classifier, 440–59
 - Machine learning, 441
 - Mulinomial logistic equation, 442
 - Multicollinearity, test for, 454–57
 - Multilevel model, 443
 - Multinomial logistic regression, 445
 - Null model, 459
 - Odd ratio, 446
 - Odds ratio, upper and lower limits, 447
 - Ordinal logistic regression, 443
 - Regression coefficients, effects of, 443, 446
 - Residuals, evaluate, 451
 - Solution equation, 442
 - Standardized residuals, 451
 - Studentized residuals, 451
 - Systematic component, 443
- Logistic regression, R functions
 - (), 451
 - dfbeta(), 451
 - dffits(), 451
 - glm(), 445, 455, 456, 457
 - rstandard(), 451
 - rstudent(), 451
- Logistic regression, templates and interpretation
 - Cases to be classified, 452
 - Compare deviance to null model, 448, 449
 - Evaluate residuals to logistic model, 451
 - Expand cases table to include probabilities and confidence limits, 453
 - glm() test for linearity, 456
 - glm() to build logistic model, 445, 455
 - Logistic regression to classify sample data, 457
 - Predict() of probability of the base classification, 452
 - relevel() set baseline category to model, 444
 - Table of classification as probabilities, 450
 - Upper, lower limits to odds ratio, 447
 - vif() to test for multicollinearity, 455
- Machine learning, 177–78, 438, 441
- Main effects, 173
- Maintenance tasks from FMECA, 5–6
- Make table query, 74
- Make Table query, 50
- Matrix, 98
- Mean, 120, 263
- Mean absolute deviation from median, 121
- Mean time to specified event, 364
- Mean trimmed of extremes, 121
- Median, 120
- Min-Max, 120
- Missing data, 110–17
- Mixed model, 384, 435
- Moubray, John, 14

482 | Index

- Multilevel model, 381, 384, 389, 422, 425, 426, 427, 431, 434, 435, 443
- Multinomial logistic equation, 442
- Multinomial logistic regression, 445
- Naïve Bayes, principles of, 459–73
 - Classify cases, 472–73
 - Cleanse and standardize text data, 462–64
 - Cleanse and standardize text, steps, 463
 - Concept of Naïve Bayes, 460
 - Corpus, text dataset, 462
 - Document term matrix, 466
 - Frequent words, indicators of, 468–69
 - Rectify misclassifications, 165
 - Train and evaluate model, 469–72
 - Train and test datasets, create, 467–68
 - Word clouds, visualize text data, 464–66
- Naïve Bayes, R functions
 - DocumentTermMatrix(), 466
 - findFreqTerms (), 468
 - inspect(), 462
 - naiveBayes(), 469, 471
 - tm_map(), 463
 - VCorpus(), 462
 - wordcloud(), 464, 465
- Naïve Bayes, templates and interpretation
 - Convert document term matrix to categorical data, 469
 - CrossTable() to evaluate model trained with Laplace, 471
 - CrossTable() to evaluate model error, 470
 - DocumentTermMatrix() to generate matrix of terms and freq, 466
 - Download text data, 461
 - findFreqTerms() to reduce document term matrix to frequency threshold, 468
 - gsub() to remove one or more sequential items in text, 462
 - Improve Bayes model with Laplace, 471
 - Inspect text in text corpus, 462
 - naiveBayes() to create trained model, 469
 - predict() to estimate unclassified cases with model, 473
 - predict() to test model with text dataset, 470
 - Subset document term matrix, 467
 - tm_map() in steps to cleanse and standardize text dataset, 463
 - VCorpus() for corpus of text dataset, 462
 - wordcloud() of cleansed and standardized dataset, 464
 - wordcloud() subset on classified text dataset, 465
- Nelson-Aalen estimator, 360
- Normal distribution, test of
 - Q-Q plots, 204
- Normal distribution, tests of
 - Histograms, 64
 - Q-Q plots, 93
 - Q-Q plots, 64
 - Q-Q plots, 122
 - Q-Q plots, 123
 - Q-Q plots, 127
 - Shapiro-Wilk, 64, 93, 124, 126
- Nowlan, F. Stanley, 14
- Null model, 172, 448, 459
- Objects
 - Prefixes to object, 51
- odds ratio, 446
- Odds ratio, upper and lower limit, 447
- One-way ANOVA, principles of, 379–81, 394–404
 - Base condition, set, 395
 - Base model, 380
 - Benjamini-Hochberg adjustment, 402
 - Bonferroni adjustment, 402
 - Concept of ANOVA, 380
 - Dependent one-way ANOVA, 381
 - Dunnett, 403
 - Family error, 379
 - Independent one-way ANOVA, 381
 - Post hoc adjustment, 381
 - Robust model for one-way ANOVA, 404
 - Tukey, 403
 - Welch's F ANOVA, 399
 - XContrasts as effects, 381
- One-way ANOVA, R functions
 - aov(), 398, 401
 - glht(), 403
 - leveneTest(), 397
 - oneway.test(), 399
- One-way ANOVA, templates and interpretation
 - aov() for significant difference to contrasts as main effects, 398, 401
 - Build contrast variables and inspect, 400
 - by() for descriptive statistics, 397
 - factor() to set variable base condition, 395
 - ggplot2() to compare confidence limits to effects, 396
 - glht() to adjust confidence intervals with Tukey and Dunnett, 403
 - leveneTest() test for constant variance across conditions, 397

- oneway.test() for difference adjusted for unequal variance, 399
- pairwise.t.test() to adjust confidence intervals with Benjamini-Hochberg, 403
- pairwise.t.test() to adjust confidence intervals with Bonferroni, 402
- plot() of variance and normality to residuals, 399
- One-way repeated measures, principles of, 422–27
 - Multilevel model, 422
 - Multilevel model, lme(), 425
- One-way repeated measures, R functions
 - glht(), 427
 - lme(), 426
- One-way repeated measures, templates and interpretation
 - anova() to test multilevel model with predictor against baseline model, 426
 - Build contrast variables and inspect, 425
 - by() for descriptive statistics, 425
 - ggplot2 to compare conditions, 424
 - glht() post hoc of multilevel with Tukey, 427
 - lme() baseline multilevel model, single repeated measure variable, 426
 - lme() model with predictor variable and single variable repeated measure, 426
 - melt() make wide data long, 423
- Operational availability, 7, 353
- Or row, Access code, 58
- Ordinal logistic regression, 443
- Outer join, 54
- Outliers, seek
 - Confidence limits
 - Contrast Z- to T-Score, 70
 - Normal distribution, assumptions of, 63
 - Student's T-Score
 - Excel functions for T_Score, 71
 - Translation table to measure for outliers, 71
 - Student's T-Score, 70–71
 - Z-Score, 63–65, 67–70
- Overall equipment effectiveness OEE, 2
- Pairs panel, 92, 131
- Partial correlation analysis, 100–103
 - Correlation (R), 100
 - Correlation squared (R²), 100
 - Misleading, as, 102
- Pearson, 97
- Poisson regression, 35
- Post hoc adjusted confidence limits, 385–86
 - Benjamini-Hochberg, 385
 - Bonferroni adjustment, 385
 - Dunnett, 385
 - Family error, 385
 - Tukey, 385
- Power BI, 26
- Predictive-based budgeting, 239
- p-value, 99, 187
- Q-Q graphic, 93
- Quartiles, 120
- Query process, overview, 50–53
- Query types
 - Append, 50, 340
 - Crosstab, 50, 233
 - Delete, 50
 - Make Table, 50
 - Make-table, 340
 - Select, 50
 - Update, 50
- Query, table and pivot templates
 - Budget and variance
 - pvtActualVsBudget, 258
 - qryActualGrp2020, 254
 - qryActualVsBudget, 256
 - qryBudget2020, 251
 - tblActual2020, 254
 - tblHistory2019, 251
 - Craft capacity
 - pvtCraftCapacity, 272
 - qryAggCraftBaseline, 274
 - qryAggCraftSample, 274
 - qrySampleVsBaseline, 276
 - tblCraftBaseline, 274
 - tblCraftSample, 274
 - Schedule week and day
 - ctqStatusCount, 345
 - qryScheduleVsActual020320, 339
 - tblCraftHrs, 334
 - tblSchedVsActual020520, 336
 - Status History
 - ctqStatusHistoryWide, 233
 - qryStageElapsed, 235
- Super table, classifier and outlier
 - qryCase1_Foundation, 56
 - qryCraftGroupHours, 72
 - qryCraftGrpStats, 76
 - qryHoursOrder, 65
 - qryHrsLeadCraft, 75
 - qryLeadCraftClassifier, 74
 - qryOrderGrpStats, 67
 - qryZScoreOrder, 69
 - qryZScoreOrderCraft, 77

484 | Index

- Variance as outlier or systemic
 - pvtOutlierOrders, 268
 - qryOutlierAggregate, 263
 - qryOutlierAggWithTest, 266
 - qryOutlierOrders, 266
- R coding, 447, *See also* R functions
 - [] as filter, 96, 113, 127, 132, 180, 208, 209, 211, 212, 214, 215, 216, 287, 295, 368, 373, 429, 455
 - '~' as '=', 104
 - '|' as OR, 132
 - <- to assign, 89
 - Case sensitivity, 86
 - confint() for confidence interval, 104
 - Data frame, 98, 202
 - Hash mark (#), 85
 - List, 202
 - Matrix, 98, 202
 - na.rm=TRUE to ignore missing data, 117
 - table\$variable syntax, 95, 298, 308, 314, 368, 393, 429
 - Vector, 98
- R functions
 - (contrasts()), 425
 - abline(), 300
 - acf(), 295, 298, 314
 - aggregate(), 284, 302
 - AIC(), 190, 191, 192, 196, 199
 - anova(), 196, 199, 431
 - Anova(), 408, 412, 418
 - aov(), 398, 401, 407, 408, 412
 - as.character(), 120, 462
 - as.data.frame(), 473
 - as.factor, 452
 - as.factor(), 120, 181
 - as.matrix() convert object to matrix, 98
 - as.numeric(), 120, 304
 - as.vector(), 373
 - attributes(), 182
 - auto.arima(), 298, 311
 - autoplot(), 311
 - boxplot(), 282, 302, 406
 - by(), 397, 406, 416, 425, 430
 - c(), 218, 312
 - c() combine function, 94, 96
 - cbind(), 183, 288, 294, 310, 312, 400, 407, 418, 430, 452
 - ccf(), 301
 - class(), 282
 - confint(), 187, 189, 403, 411, 427, 447
 - contrasts(), 182, 183, 418, 430
 - cor(), 455
 - cor() for correlation analysis, 97
 - cor.test() for correlation analysis, 99, 133
 - coxph(), 366
 - CrossTable(), 470, 471
 - data(), 289
 - Data(), 295
 - data.frame, 452
 - data.frame(), 218, 304, 310, 312
 - date_decimal(), 312
 - decompose(), 292, 295, 301, 304
 - describe(), 91, 121
 - dev.off() to close output device, 114, 158
 - dfbeta(), 451
 - dffits(), 451
 - diff(), 314, 318
 - distinct(), 214
 - DocumentTermMatrix(), 466
 - durbinwatsonstest(), 200
 - effect(), 408
 - end(), 282, 288, 289
 - exp(), 368, 447
 - factor(), 395, 405, 414, 461
 - findFreqTerms(), 468
 - fitted(), 450
 - forecast(), 311
 - format(), 310, 312
 - frequency(), 282
 - ggarrange(), 136, 206
 - gl(), 429
 - glht(), 403, 411, 427, 434
 - glm(), 445, 455, 456, 457
 - gsub(), 462
 - Guidance, 96
 - hatvalues(), 208, 451
 - head(), 90, 119
 - help(), 96
 - Holt Winters(), 308
 - ifelse(), 133, 181
 - inspect(), 462
 - install.packages(), 86, 179, 281
 - lapply(), 462, 463
 - layout(), 282, 284, 286, 289, 295, 300, 301, 302, 314
 - length(), 208, 211, 212, 295
 - leveneTest(), 397, 406, 417
 - library(), 87, 179
 - lines(), 313
 - lm() for linear regression, 104

- lm.beta, 186
- lme(), 426, 431
- log(), 456
- md.pattern(), 113, 121, 180
- mean(), 318
- melt(), 423, 429
- MLEW2p(), 374
- MRRW2p(), 374
- naiveBayes(), 469, 471
- names(), 429
- oneway.test(), 399
- pairs.panel(), 92, 131, 180
- par(), 399, 406, 411, 465
- par() for parameters, 94
- pchisq(), 448, 449
- pcor() for partial correlation, 102
- pcor.test() for partial correlation, 103
- pdf() output device to pdf document, 158
- phreg(), 368, 370, 372
- plot(), 282, 284, 288, 289, 295, 366, 399
- png() to output .png file, 114
- predict(), 215, 218, 309, 452, 470, 471, 473
- prop.table(), 467
- qqline(), 94
- qqnorm(), 94
- rbind(), 216
- rcorr() for correlation analysis, 98
- read.csv(), 89, 112
- read.delim() to read txt. and dat. file, 89
- read.xlsx(), 89, 113, 119, 286, 299
- relevel(), 184
- rlm() for robust linear regression, 107
- rstandard(), 194, 195, 198, 212, 451
- rstudent(), 451
- seq(), 135
- shapiro.test(), 93, 124
- sqldf(), also see SQL in background, 90
- start(), 282, 288
- str(), 90
- Strata(), 370
- subset(), 465
- summary(), 91, 120, 366, 398, 401, 427
- summary.lm(), 420
- Surv(), 366, 368, 370, 372
- survfit(), 366
- t.test(), 392, 393
- table(), 467
- time(), 312
- tm_map(), 463
- ts(), 284, 286, 287, 300, 302, 314
- ts.intersect(), 287
- ts.plot(), 294, 300, 309
- update(), 431
- VCorpus(), 462
- vif(), 201, 455
- window(), 285, 301, 304
- within(), 452
- wordcloud(), 464, 465
- write() export data from R, 90
- write.xlsx(), 284, 287
- xlsx(), 179
- R packages
 - car, 405, 444
 - car(), 394
 - e1071, 461
 - effects, 405
 - forecast, 281
 - ggfortify, 281
 - ggm, 86, 110
 - ggplot2, 86, 110, 179, 389, 405, 422, 428
 - ggplot2(), 394
 - ggplotr, 179
 - ggpubr, 110, 179
 - gmodels, 461
 - Hmisc, 86, 110
 - lubricate, 281
 - MASS, 86, 110, 179
 - mice, 110, 179
 - mlogit, 444
 - multicomp, 405, 422, 428
 - multicomp(), 394
 - nlme, 110, 179, 422, 428
 - pastec, 389, 428
 - pastecs, 86, 405, 422
 - pastecs(), 394
 - polycor, 86, 110
 - psych, 86, 110, 179
 - qqplotr, 110
 - QuantPsyc, 179
 - RColorBrewer, 461
 - reshape, 422, 428
 - rlm, 86, 110
 - Snowballc, 461
 - wordcloud, 461
 - X, 461
 - xlsx, 110, 112, 179, 281, 405, 422, 428
- R session, 84–87
 - Console, 84
 - Download and install, 84
 - Export data, 90

486 | Index

- Import data, 89–90
- Install packages, 85–87
- Know-thy-data, 90–95
 - Layered charting, 94
 - Normal distribution, 92
 - Pairs panel, 92
- Script window, 84–85
- Working directory, 89
- R software, 26–29
 - Download and install, 26
 - Layered charting. *See* Layered Charting
 - Power BI, 26
 - Script (AKA, code), 28
 - Tableau, 26
 - Windows of, 28
- Range, 121
- Regression Coefficient defined, 95
- Regression coefficients, effects of, 443, 446
- Relational database, 230
- Relationship insight, 34–35
 - Linear regression, 34
 - Logistic regression, 34
 - Poisson regression, 35
- Reshape data, 428
- Residual standard error, 187, 190, 192, 197
- Right join, 53
- Robust difference model, 384
- Robust linear regression, 107
- R-squared, 187, 192
- Schedule cycle, measurement of, 341–51
 - Accuracy and variance, job plans, 346–49
 - Flowchart, queries, 347
 - qryPlanToActualVar, 349
 - Execution, planned and scheduled, 343–46
 - ctqStatusCount, 345
 - Flowchart, queries, 345
 - Sustainment, confirm, 341–43
 - Flowchart, queries, 341
 - Time series, measured by, 349–51
 - ARIMA (Autoregressive integrated moving average), 350
 - Holt Winters, 350
 - Lead-lag relationship, 350
 - Seven-day cycle, 350
- Schedule cycle, stages to, 321–29
 - Datasets, measures and reports, 327–29
 - Flowchart, stage, 327
 - Goals, two, 321
 - Schedule and craft capacity, baseline, 321–24
 - Flowchart, stage, 321
 - Stages, three, 321
 - Weekly schedule and execution, conduct, 324–26
 - Flowchart, stage, 325
- Schedule cycle, tables to, 329–40, 333–34
 - Craft classification table, 334–36
 - Craft hours
 - tblCraftHrs, 334
 - Day schedule, 331–33
 - tblDaySchedule, 332
 - Super table to measurement, 336–40
 - Flowchart, queries, 337
 - qryScheduleVsActual020320, 339
 - tblSchedVsActual020520, 336
 - System of tables, 329
 - Weekly schedule, 330
- Schedule measures and methods. *See* Schedule cycle, measures of
- Select query, 48, 50
- Shapiro-Wilk test, 64, 93, 203
- Skew, 121
- Solution equation, 442
- Spreadsheet compared to data table, 223
- SQL in background, 25–26, 79
- Standard deviation, 121, 263
- Standard error to the mean, 121
- Standardized, 451
- Standardized effect, 185
- Standardized effects of variables, 186
- Standardized residuals, 194
- Studentized residuals, 451
- Super table with Access, 22–26
 - Generate, 61–62
 - Pictorial view of, 22
 - Process to build, 23
 - SQL in background, 25–26
 - Steps to build, 46
 - Table as format, 22, 222
- Survival function, 362, 363
- Survival hazard analysis, 38
- Systematic component, 443
- Table as format, 22, 222
- Tableau, 26
- Three models, 163
- Time series analytics, 241, *See* Time series, principles of, Time series, templates and interpretation, and Time series, R functions
- Time series insight, 36–38, *See also* header, Time series, principles of
 - Autocorrelation, 37
 - Components, 36

- Cycles, 36
 - Trend, type of, 37
- Cross-correlation, 37
- Lead-lag relationship, 37
- Models, 38
- Prediction compared to forecast, 37
- Time series, principles of
 - Aggregate months to year, 284
 - ARIMA (Autoregressive integrated moving average), 350
 - ARIMA model (Autoregressive integrated moving average), 311–13
 - ARIMA(Autoregressive integrated moving average), 298
 - Autocorrelation, 295–99
 - Correlogram, 295
 - Cross correlation, 301
 - Cycle, 291
 - Decompose and Holt Winters, compared, 308
 - Decomposition, 291
 - Drift in random walk, test for, 317
 - Exponentially weighted moving average (EWMA), 305
 - Holt Winters, 305–11, 350
 - Four-year forecast, 309
 - Smoothing equation and prediction, 306
 - Lead, lag relationship, 299–302
 - Lead-lag relationship, 350
 - Moving 12-month average, 290
 - Notation, of time series, 290
 - Questions of a series, 280
 - Seven-day cycle, 302–5, 350
 - Test as deterministic or stochastic, 314–17
 - Time series object, 282
 - Trend, 291
- Time series, R functions
 - acf(), 295, 298, 314
 - aggregate(), 284, 302
 - auto.arima(), 311
 - autoplot (), 311
 - ccf(), 301
 - class(), 282
 - data(), 289
 - Data(), 295
 - decompose(), 292, 301, 304
 - diff(), 314, 318
 - end(), 282, 288, 289
 - forecast(), 311
 - frequency(), 282
 - Holt Winters(), 308
 - layout(), 314
 - start(), 282, 288
 - ts(), 284, 286, 287, 300, 302
 - ts.intersect(), 287
 - ts.plot(), 294, 300, 309
 - window(), 285, 301, 304
- Time series, templates and interpretation
 - ARIMA forecasted 4-years out, 311
 - ARIMA plot of forecasted mean and limits, 313
 - ARIMA, table of forecast mean, and upper and lower limits, 312
 - Comparative correlograms to observed and random series, 301
 - Correlogram of random series, 314
 - Correlogram of random series and difference series, 314
 - Correlogram of random series in decomposition, 295
 - Correlogram of residuals to an ARIMA model, 298
 - Correlogram of time series, 295
 - Extract factors of daily cycle, 304
 - Extract random series from time series, 301
 - Extract windows from time series object, 285
 - Format to time series object, 284
 - Holt Winters forecast, plot and data table, 310
 - Holt Winters, 4-year forecast, 309
 - Holt Winters, smoothing parameters and coefficient, multiplicative, 308
 - Lead-lag correlogram, multiple datasets, 300
 - Multiple variables synchronized by date, 287
 - Plot multiple time series dataset, different axis, 286
 - Series cycle plotted over trend, 294
 - Test random walk for drift, 318
 - Test series as random walk, 316
 - Time series decomposed, additive and multiplicative, 292
 - TS object inspect, 282
 - Weekly pattern, observations, aggregate and day range, 302
- Totals row, Access code
 - Summary Option, 62
- Translation tables, 49–50, 164, 264
- Trees, 166, 168
- Triad of software, 17
- T-Score, 70–71, 262
- T-Score, table of, 265
- Tukey adjustment, 385, 403
- Two-mean t-test, principles of, 379, 389–94
 - Data, long and wide, 390

488 | Index

- Data. Descriptive statistics of, 392
- Dependent t-test (AKA, paired), 379
- Independent t-test, 379
- Paired two-mean t-test, 393
- Two-mean t-test, R functions
 - t.test(), 392, 393
- Two-mean t-test, templates and interpretation
 - by() for descriptive statistics, 392
 - t.test() for significance difference in means, 392
 - t.test() to test for significant of paired observations, 393
- Update query, 50
- Variance inflation factor (VIF), 201
- Variance reports, 248
- Vector, 98
- Weibull plot, 373
- Work schedule, stages to. *See* Schedule cycle, stages to
- Work schedule, tables to. *See* Schedule cycle, tables to
- Workload from history, 250
- Workload-based budget and variance. *See* Budget and variance, principles and data methodology
- Zero-based budgeting, 239
- Z-Score, 63–65, 262