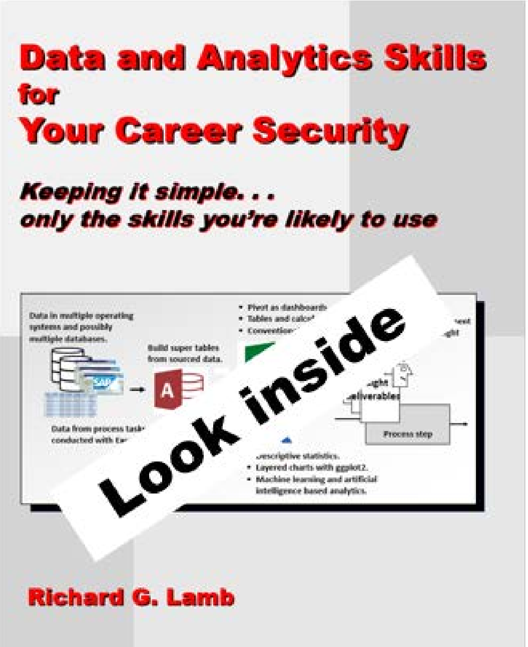


Data and Analytics Skills for Your Career Security

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

Richard G. Lamb

Look inside
Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
SAP
A
Data from process tasks conducted with Ex
Pivot as dashboards
Tables and calc
Convention
deliverables
Process step
descriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

- Table of Contents
- Preface
- All Chapters
- Index
- Back cover

Note:
Chapter 2, as overview of data-driven operations, is presented in its entirety.

Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds

Contents

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 mandatary 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, 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.

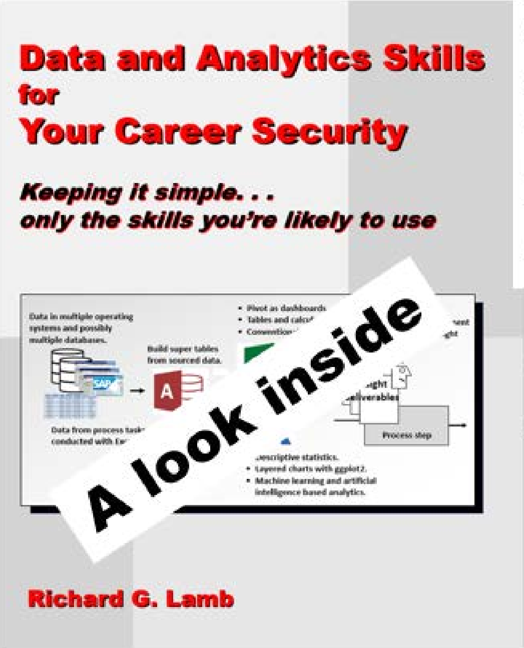
Unfortunately, the cost to produce the book in color is prohibitive to pricing. To keep the price reasonable, the book is produced in grayscale. The readers, of course, can view each graphic in full color by running the provided R code. Alternatively, the reader can view any visualization in full color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

Is it 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.



Data and Analytics Skills for Your Career Security

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

Richard G. Lamb

A look inside
Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Pivot as dashboards
Process step
Descriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

All operations entail setting direction, aggregate planning, budget and variance control, planning, scheduling and executing the core task, staffing and optimizing the retention time of the core task in the stages to the operation. It follows that the some data and analytics skills are common to all. But to present the skills, we need a reference operation. The book has chosen availability management as the reference.

Chapter 1: Operational Availability is the Purpose

Excerpts:

1.3. Top Factors of Operational Availability

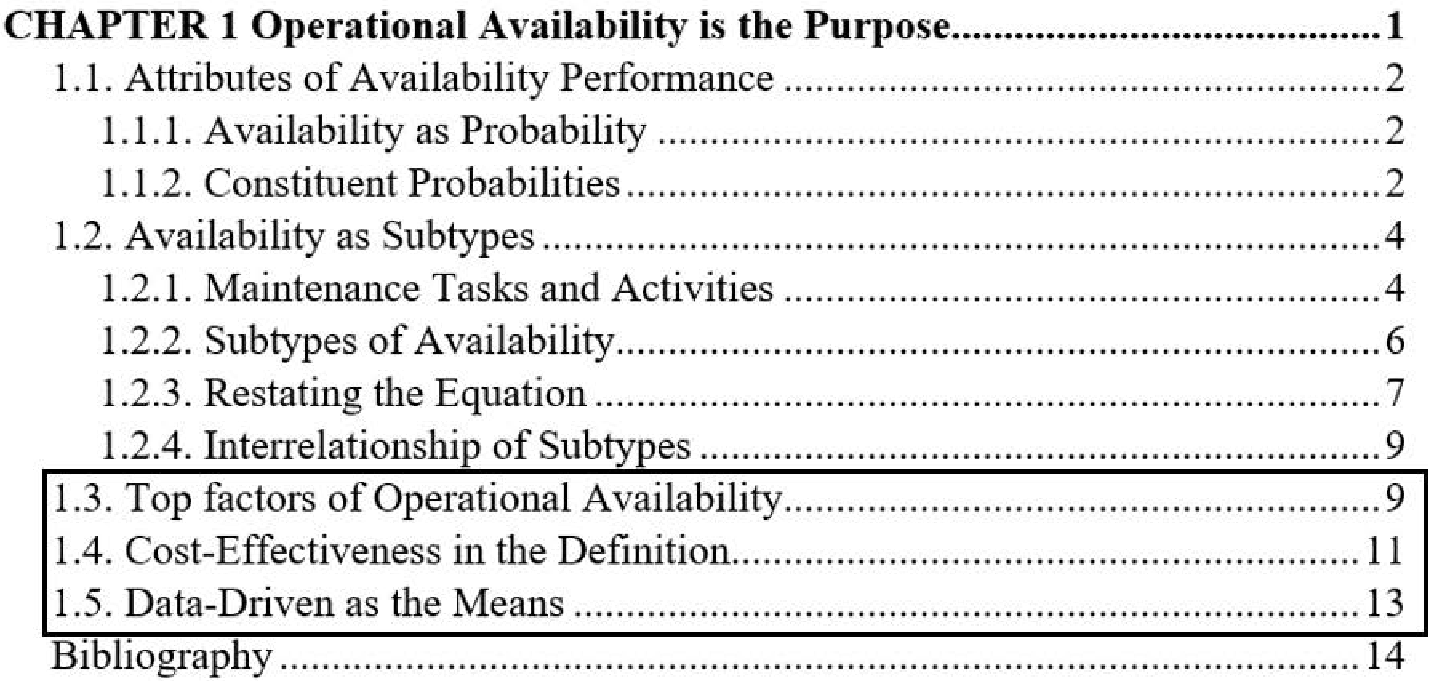
1.4. Cost-Effectiveness in the Definition

1.5. Data-Driven as the Means

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside

Book is available from Amazon

Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds



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.

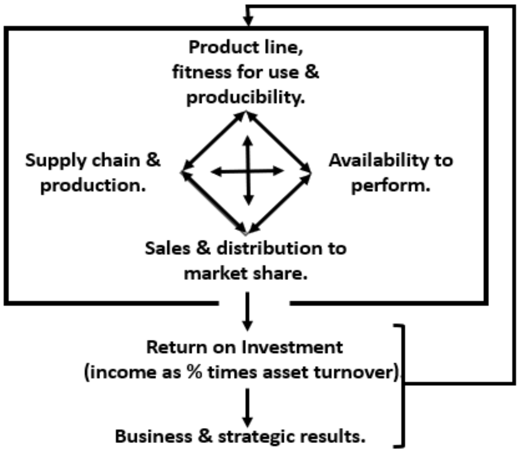

Product line, fitness for use & producibility.
Supply chain & production.
Availability to perform.
Sales & distribution to market share.
Return on Investment (income as % times asset turnover)
Business & strategic results.

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 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.

1.3. Top Factors of Operational Availability

The chapter brings to the surface the overarching purpose of asset management. Aligned with the enterprise's business strategy, its purpose is to plan, organize, conduct and control availability performance through the mean time between maintenance (MTBM) and mean time to maintain (MTTM). As a collaboration of reliability, maintenance and production operations, the firm's asset management organization is responsible and accountable for the cross-enterprise top-level factors of operational availability.

Operational availability is the synthesis and optimization of the factors. This section will group the factors at the intersections of the mean time and life cycle dimensions of operational availability. We can imagine that for many factors, data and analytics allow us to maximize the enterprise's operational outcome by optimizing the parts. .

Reliability	**Maintainability**
Goal: Increase mean time between maintenance (MTBM).	**Goal:** Reduce mean time to maintain (MTTM).
Factors Driven by Design Decisions	
• Selection of equipment. • Operating environment and context. • Equipment rated capacity. • Maintenance while function continues. • Installed spare components within an equipment item. • Redundant equipment and subsystems. • Simplicity of design and elimination of weak points.	• Plant ingress and egress. • Accessibility to work points. • Features that make for ease of maintenance. • Work environment.
Factors Driven by Maintenance Decisions	
• Scheduled maintenance tasks decided upon on survival-hazard analytics or rational equivalent upon experience and judgement. • Skill-levels engaged in maintenance tasks. • Quality of conducted maintenance tasks.	• How maintenance tasks are detailed, developed and presented to the maintenance technicians. • Probability of human, parts and materials, and facility resources being available to maintenance tasks. • Training program. • Operational effectiveness of the maintenance process and its leadership. • Durability of handling, support and test equipment.
Factors Driven by Production Decisions	
• Use of equipment relative to its rated capacity.	• Collaboration in the troubleshooting process.

• How spares are cycled in normal process operation. • Shutdown and startup procedures. • Choices in production parts and raw materials.	• Procedures to make equipment ready for maintenance and return to capable.

Ideally, development of the top-level factors begins with incorporating availability engineering in the traditional design, build and startup stages of the capital project. Furthermore, its incorporation must be on par with the stature given the traditional disciplines to capital projects.

Once the window of opportunity has closed, the top-level factors for maintenance and production are still pliable. Relative to the production factors, the maintenance factors offer the greatest potential of the two. Unfortunately, some business ramifications of asset management are permanently lost.

1.4. Cost-Effectiveness in the Definition

Let's revisit the definition of availability performance. 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. .

Notice the expression "cost-effective support" in the definition. Recall that cost effectiveness in the definition of availability rolls up through the definition of maintainability as the probability of retention and the probability of the availability and cost of resources.

Cost-effective support is not automatic to reaching and holding the enterprise's quested operational availability. Without cost-effective support in the definition of maintainability, the default is to incur maintenance cost in excess of necessary. Of course, planned support can also fall short of necessary but tends to be self-limiting or self-correcting because the consequences inevitably become unavoidably apparent.

Cost-effectiveness is measured up through the high-level factors of operational availability. There are two categories of cost—operating expense and expensed capital spending. Operating expense is a cost that flows directly from incurrence to the income statement. Expensed capital spending is a cost that is allocated to income statements as an expense over the accounting life of capital assets.

Now to define cost-effective with respect to operational availability. Cost-effectiveness relates measures of plant, system or asset performance or merit to the total life cycle cost.

The book is of special interest for industries in which the capitalized value of production assets looms large in the balance sheet and the expenses of asset availability loom large for the income statement. For these industries, the profit margins may be so tight that cost-effective support can be the difference between strong or dismal financials and return on investment.

With the system design as fixed by the decisions at the design stage, the decisions for the top-level factors of maintenance most decide cost effectiveness. There are two areas for evaluating the cost effectiveness of the decisions for the top-level factors of maintenance operations. They are proficiency in the conduct of the system's designed workload and the magnitude of support resources engaged or consumed to conduct the workload.

Proficiency of the maintenance operation is the outcome of engaged skill levels, the quality of maintenance, how maintenance work is detailed and presented, and the effectiveness and efficiency of the maintenance work processes. For these, cost effectiveness is quantitatively and qualitatively evaluated in ways that relate the price of enhanced proficiency to a unit change in the probability of operational availability.

The second area of cost effectiveness are the decisions that result in the probability of human, parts and materials, and facility and equipment resources being available to the timely and efficient conduct of the maintenance workload.

The competitive ramifications of probability of available craft resources loom largest among all resources. The first determinate of cost effectiveness is the probability of being able to execute a representative day's workload. The second determinate is the match of actual and planned crafts count and hours to a statistical sample of work orders. In other words, an optimally sized craft body with respect to persistently, rather than sporadically, delivering the daily workload is a primary measure of cost effectiveness.

Cost effectiveness for parts and materials is also an issue for support resources, but differently so. The issue is not so much the total value and per-item cost of the maintenance inventory as it is for production and finished goods inventory. This is because maintenance inventory is often a small percent of total assets. Significance to assets in the balance sheet of reducing the inventory is measured by the before and after calculations of revenues divided by total assets with respect to a change in maintenance inventory. Otherwise, the expense of maintenance parts and materials is largely fixed by plant design and assigned maintenance tasks.

Instead of dollar value, inventory is best evaluated with respect to the statistical play in the probability of operational availability. Once set on probability, the magnitude of the administrative, logistic and holding expenses of maintenance inventory are evaluated for their ramifications to profit margin.

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.

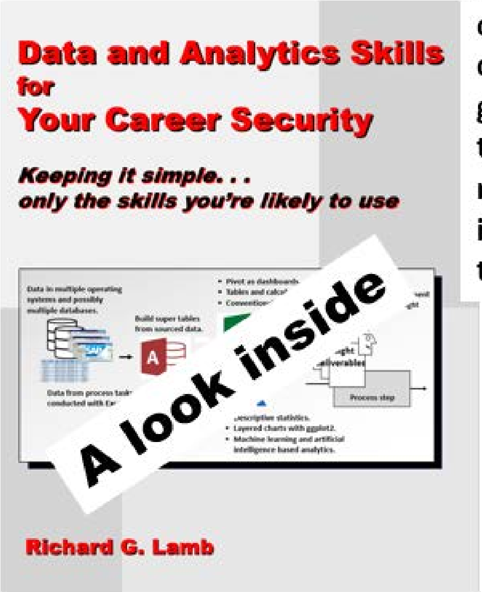
1.5. Data-Driven as the Means

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.



Data and Analytics Skills for Your Career Security

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

A look inside

Richard G. Lamb

Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Pivot as dashboards.
Tables and calc
Data from process task conducted with Ex
Process step
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

Say "data-driven" to management and they cringe. They envision a complex, high-tech, deep-capital initiative and every horror that goes with it. This is a gross misperception that has been propagated by purveyors and media as they speak of grand and glorious initiatives, e.g., IoT. The reality is opposite to the perception.

Chapter 2: Data, Analytics and Software to be Data Driven

Look Inside the book at https://analytics4strategy.com/book-look-inside
Book is available from Amazon.com
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds

Note:

The advance graphics created with the ggplot2 package of the R software depend heavily on color to communicate the insight they are coded to give us. Some such graphics appear in the chapter.

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.

For convenience, the herein excerpts will present the color version.

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.

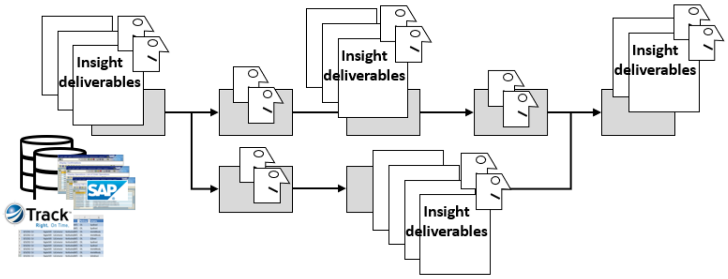


Figure 2-1: What a data-driven operation or process looks like.

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.

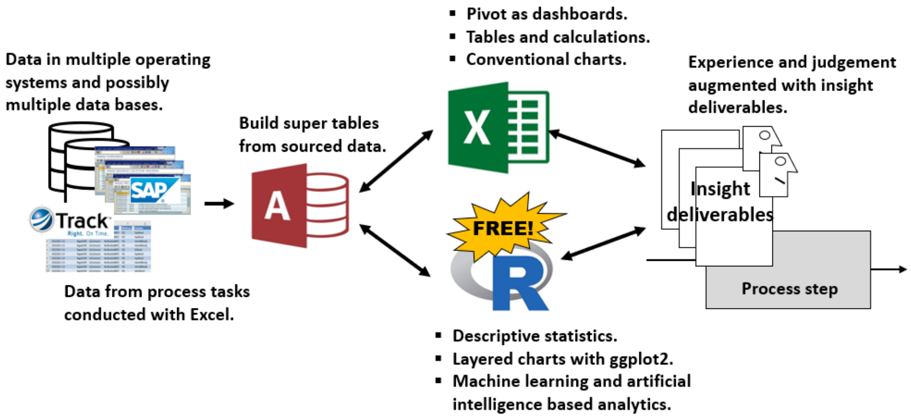

▪ Pivot as dashboards.
▪ Tables and calculations.
▪ Conventional charts.
Data in multiple operating systems and possibly multiple data bases.
Experience and judgement augmented with insight deliverables.
Build super tables from sourced data.
SAP
Track
Insight deliverables
FREE!
Process step
Data from process tasks conducted with Excel.
▪ Descriptive statistics.
▪ Layered charts with ggplot2.
▪ Machine learning and artificial intelligence based analytics.

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 grassroot 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.

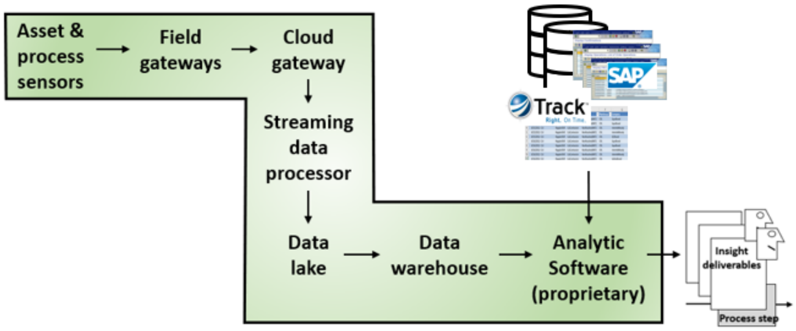


Figure 2-3: Insight deliverables built upon IIoT infrastructure.

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.

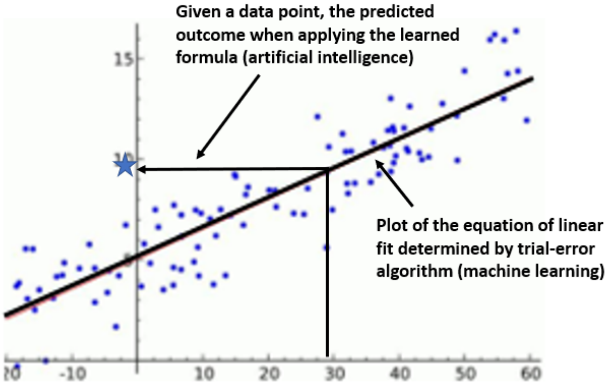


Figure 2-4: A two-variable regression demonstrates machine learning, artificial intelligence and algorithms.

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.

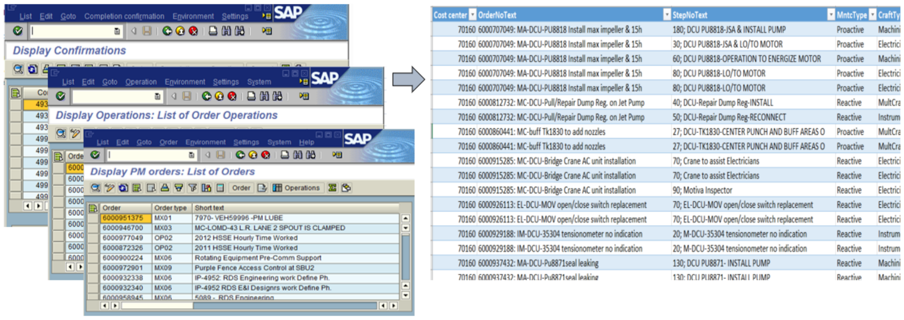
Super tables can be built in "R" with the structured query language (SQL) capability and with additional coding in the rare occasions when it is necessary to go beyond the ability of SQL. The variables from their sources are imported into Excel Pivot and "R."

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.


List Edit Goto Completion confirmation Environment Settings
SAP
List Edit Goto Operation Environment Settings System
SAP
List Edit Goto Order Environment Settings System Help
SAP
Order
Operations

Display Confirmations

Display Operations: List of Order Operations

Display PM orders: List of Orders

Order	Order type	Short text
6000951375	MX01	7970- VEH59996 -PM LUBE
6000946700	MX03	MC-LOMD-43 L.R. LANE 2 SPOUT IS CLAMPED
6000977049	OP02	2012 HSSE Hourly Time Worked
6000872326	OP02	2011 HSSE Hourly Time Worked
6000900224	MX06	Rotating Equipment Pre-Comm Support
6000972901	MX09	Purple Fence Access Control at SBU2
6000932338	MX06	IP-4952: RDS Engineering work Define Ph.
6000932340	MX06	IP-4952 RDS E&I Designrs work Define Ph.
6000958945	MX06	5089 - RDS Engineering

Cost center	OrderNoText	StepNoText	MntcType	CraftTy
70160	6000707049: MA-DCU-PU8818 Install max impeller & 15h	180; DCU PU8818-JSA & INSTALL PUMP	Proactive	Machini
70160	6000707049: MA-DCU-PU8818 Install max impeller & 15h	30; DCU PU8818-JSA & LO/TO MOTOR	Proactive	Electrici
70160	6000707049: MA-DCU-PU8818 Install max impeller & 15h	60; DCU PU8818-OPERATION TO ENERGIZE MOTOR	Proactive	Machini
70160	6000707049: MA-DCU-PU8818 Install max impeller & 15h	80; DCU PU8818-LO/TO MOTOR	Proactive	Electrici
70160	6000707049: MA-DCU-PU8818 Install max impeller & 15h	80; DCU PU8818-LO/TO MOTOR	Proactive	Electrici
70160	6000812732: MC-DCU-Pull/Repair Dump Reg. on Jet Pump	40; DCU-Repair Dump Reg-INSTALL	Reactive	MultCra
70160	6000812732: MC-DCU-Pull/Repair Dump Reg. on Jet Pump	50; DCU-Repair Dump Reg-RECONNECT	Reactive	Instrum
70160	6000860441: MC-buff Tk1830 to add nozzles	27; DCU-TK1830-CENTER PUNCH AND BUFF AREAS O	Proactive	MultCra
70160	6000860441: MC-buff Tk1830 to add nozzles	27; DCU-TK1830-CENTER PUNCH AND BUFF AREAS O	Proactive	MultCra
70160	6000915285: MC-DCU-Bridge Crane AC unit installation	70; Crane to assist Electricians	Reactive	Electrici
70160	6000915285: MC-DCU-Bridge Crane AC unit installation	70; Crane to assist Electricians	Reactive	Electrici
70160	6000915285: MC-DCU-Bridge Crane AC unit installation	90; Motiva Inspector	Reactive	Electrici
70160	6000926113: EL-DCU-MOV open/close switch replacement	70; EL-DCU-MOV open/close switch replacement	Reactive	Electrici
70160	6000926113: EL-DCU-MOV open/close switch replacement	70; EL-DCU-MOV open/close switch replacement	Reactive	Electrici
70160	6000929188: IM-DCU-35304 tensionometer no indication	20; M-DCU-35304 tensionometer no indication	Reactive	Instrum
70160	6000929188: IM-DCU-35304 tensionometer no indication	20; M-DCU-35304 tensionometer no indication	Reactive	Instrum
70160	6000937432: MA-DCU-Pu8871seal leaking	130; DCU PU8871- INSTALL PUMP	Reactive	Machini
70160	6000937432: MA-DCU-Pu8871seal leaking	130; DCU PU8871- INSTALL PUMP	Reactive	Machini

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 a MS Access.

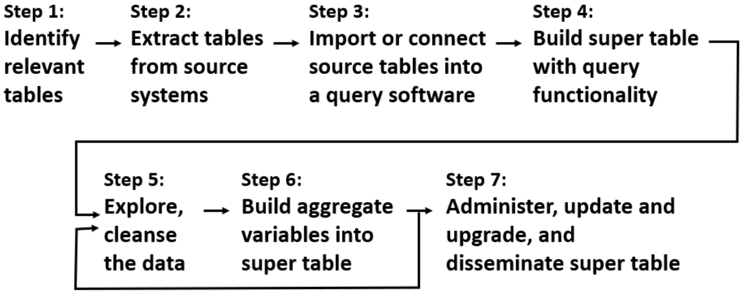

Step 1:
Identify relevant tables
Step 2:
Extract tables from source systems
Step 3:
Import or connect source tables into a query software
Step 4:
Build super table with query functionality
Step 5:
Explore, cleanse the data
Step 6:
Build aggregate variables into super table
Step 7:
Administer, update and upgrade, and disseminate super table

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

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.

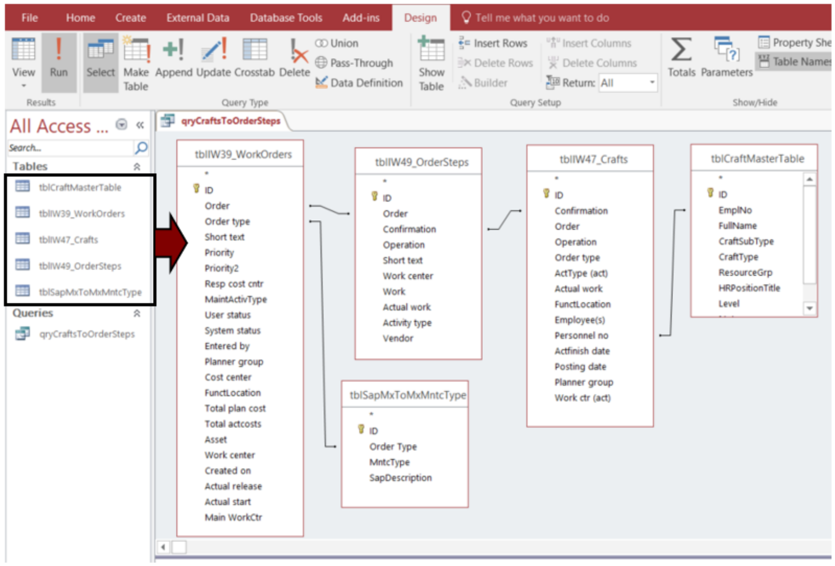


Figure 2-7: Tables imported to the query software and joined as a single table.

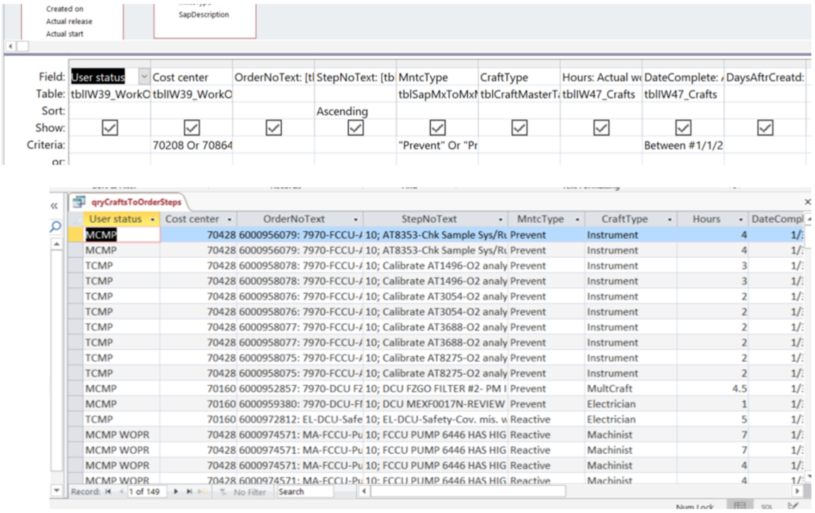
Notice the lines between the tables that were pulled into the work area. The line symbolizes that the tables are joined in a massive raw table. Each pair of tables is joined by a unique identifying variable they have in common.

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.



Field:	User status	Cost center	OrderNoText: [tl	StepNoText: [tb	MntcType	CraftType	Hours: Actual w	DateComplete: /	DaysAftrCreatd:
Table:	tblIW39_WorkO	tblIW39_WorkO			tblSapMxToMxI	tblCraftMasterT	tblIW47_Crafts	tblIW47_Crafts	
Sort:				Ascending					
Show:	☑	☑	☑	☑	☑	☑	☑	☑	☑
Criteria:		70208 Or 70864			"Prevent" Or "Pr			Between #1/1/2	
or:									

User status	Cost center	OrderNoText	StepNoText	MntcType	CraftType	Hours	DateCompl
MCMP	70428	6000956079: 7970-FCCU-/	10; AT8353-Chk Sample Sys/Ru	Prevent	Instrument	4	1/
MCMP	70428	6000956079: 7970-FCCU-/	10; AT8353-Chk Sample Sys/Ru	Prevent	Instrument	4	1/
TCMP	70428	6000958078: 7970-FCCU-/	10; Calibrate AT1496-O2 analy	Prevent	Instrument	3	1/
TCMP	70428	6000958078: 7970-FCCU-/	10; Calibrate AT1496-O2 analy	Prevent	Instrument	3	1/
TCMP	70428	6000958076: 7970-FCCU-/	10; Calibrate AT3054-O2 analy	Prevent	Instrument	2	1/
TCMP	70428	6000958076: 7970-FCCU-/	10; Calibrate AT3054-O2 analy	Prevent	Instrument	2	1/
TCMP	70428	6000958077: 7970-FCCU-/	10; Calibrate AT3688-O2 analy	Prevent	Instrument	2	1/
TCMP	70428	6000958077: 7970-FCCU-/	10; Calibrate AT3688-O2 analy	Prevent	Instrument	2	1/
TCMP	70428	6000958075: 7970-FCCU-/	10; Calibrate AT8275-O2 analy	Prevent	Instrument	2	1/
TCMP	70428	6000958075: 7970-FCCU-/	10; Calibrate AT8275-O2 analy	Prevent	Instrument	2	1/
MCMP	70160	6000952857: 7970-DCU FZ	10; DCU FZGO FILTER #2- PM I	Prevent	MultCraft	4.5	1/
MCMP	70160	6000959380: 7970-DCU-FI	10; DCU MEXF0017N-REVIEW	Prevent	Electrician	1	1/
TCMP	70160	6000972812: EL-DCU-Safe	10; EL-DCU-Safety-Cov. mis. w	Reactive	Electrician	5	1/
MCMP WOPR	70428	6000974571: MA-FCCU-Pu	10; FCCU PUMP 6446 HAS HIG	Reactive	Machinist	7	1/
MCMP WOPR	70428	6000974571: MA-FCCU-Pu	10; FCCU PUMP 6446 HAS HIG	Reactive	Machinist	7	1/
MCMP WOPR	70428	6000974571: MA-FCCU-Pu	10; FCCU PUMP 6446 HAS HIG	Reactive	Machinist	4	1/
MCMP WOPR	70428	6000974571: MA-FCCU-Pu	10; FCCU PUMP 6446 HAS HIG	Reactive	Machinist	4	1/

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 repression 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.

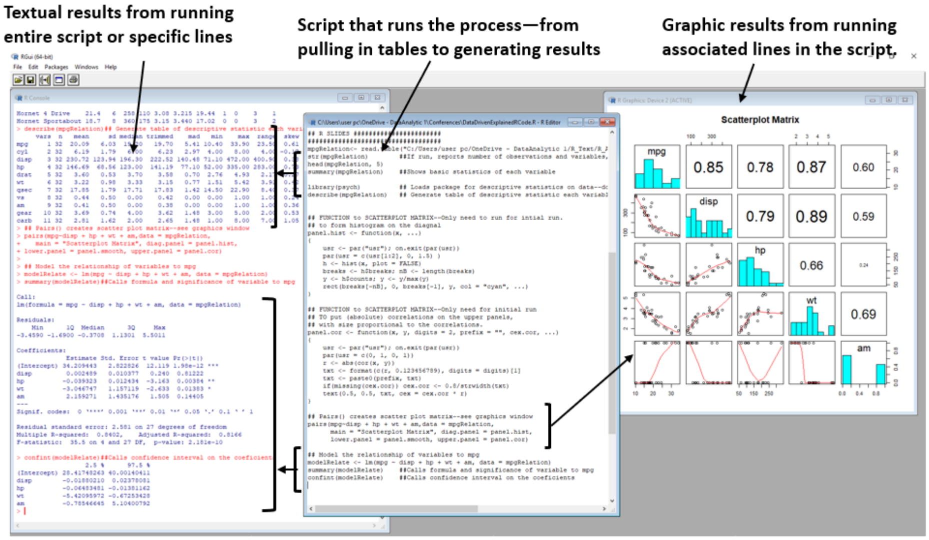
Figure 2-9 show the primary three windows of "R." Like all software they can be moved and sized.

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.



Textual results from running entire script or specific lines

Script that runs the process—from pulling in tables to generating results

Graphic results from running associated lines in the script.

RGui (64-bit)
File Edit Packages Windows Help
R Console
C:\Users\user pc\OneDrive - DataAnalytic 1\Conferences\DataDrivenExplainedRCode.R - R Editor
R Graphics: Device 2 (ACTIVE)
Scatterplot Matrix
mpg
disp
hp
wt
am
0.85
0.78
0.87
0.60
0.79
0.89
0.59
0.66
0.24
0.69
100
300
2
3
4
5
10
20
30
50
150
250
0.0
0.4
0.8

```
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
> describe(mpgRelation)## Generate table of descriptive statistic each vari
     vars  n   mean     sd median trimmed    mad   min    max  range  skew
mpg     1 32  20.09   6.03  19.20   19.70   5.41 10.40  33.90  23.50  0.61
cyl     2 32   6.19   1.79   6.00    6.23   2.97  4.00   8.00   4.00 -0.17
disp    3 32 230.72 123.94 196.30  222.52 140.48 71.10 472.00 400.90  0.38
hp      4 32 146.69  68.56 123.00  141.19  77.10 52.00 335.00 283.00  0.73
drat    5 32   3.60   0.53   3.70    3.58   0.70  2.76   4.93   2.17  0.27
wt      6 32   3.22   0.98   3.33    3.15   0.77  1.51   5.42   3.91  0.42
qsec    7 32  17.85   1.79  17.71   17.83   1.42 14.50  22.90   8.40  0.37
vs      8 32   0.44   0.50   0.00    0.42   0.00  0.00   1.00   1.00  0.24
am      9 32   0.41   0.50   0.00    0.38   0.00  0.00   1.00   1.00  0.36
gear   10 32   3.69   0.74   4.00    3.62   1.48  3.00   5.00   2.00  0.53
carb   11 32   2.81   1.62   2.00    2.65   1.48  1.00   8.00   7.00  1.05
> ## Pairs() creates scatter plot matrix--see graphics window
> pairs(mpg~disp + hp + wt + am,data = mpgRelation,
+     main = "Scatterplot Matrix", diag.panel = panel.hist,
+ lower.panel = panel.smooth, upper.panel = panel.cor)
>
> ## Model the relationship of variables to mpg
> modelRelate <- lm(mpg ~ disp + hp + wt + am, data = mpgRelation)
> summary(modelRelate)##Calls formula and significance of variable to mpg

Call:
lm(formula = mpg ~ disp + hp + wt + am, data = mpgRelation)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4590 -1.6900 -0.3708  1.1301  5.5011

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 34.209443   2.822826  12.119 1.98e-12 ***
disp         0.002489   0.010377   0.240  0.81222
hp          -0.039323   0.012434  -3.163  0.00384 **
wt          -3.046747   1.157119  -2.633  0.01383 *
am           2.159271   1.435176   1.505  0.14405
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.581 on 27 degrees of freedom
Multiple R-squared:  0.8402,    Adjusted R-squared:  0.8166
F-statistic:  35.5 on 4 and 27 DF,  p-value: 2.181e-10

> confint(modelRelate)##Calls confidence interval on the coeficient
                  2.5 %      97.5 %
(Intercept) 28.41748263 40.00140411
disp        -0.01880210  0.02378081
hp          -0.06483481 -0.01381162
wt          -5.42095972 -0.67253428
am          -0.78546645  5.10400792
>
```

```
## R SLIDES ########################
####################################
mpgRelation<- read.table("C:/Users/user pc/OneDrive - DataAnalytic 1/R_Text/R_
str(mpgRelation)          ##If run, reports number of observations and variables,
head(mpgRelation, 5)
summary(mpgRelation)      ##Shows basic statistics of each variable

library(psych)            ## Loads package for descriptive statistics on data--dc
describe(mpgRelation)     ## Generate table of descriptive statistic each variabl


## FUNCTION to SCATTERPLOT MATRIX--Only need to run for intial run.
## to form histogram on the diagnal
panel.hist <- function(x, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
}

## FUNCTION to SCATTERPLOT MATRIX--Only need for initial run
## TO put (absolute) correlations on the upper panels,
## with size proportional to the correlations.
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}

## Pairs() creates scatter plot matrix--see graphics window
pairs(mpg~disp + hp + wt + am,data = mpgRelation,
      main = "Scatterplot Matrix", diag.panel = panel.hist,
      lower.panel = panel.smooth, upper.panel = panel.cor)

## Model the relationship of variables to mpg
modelRelate <- lm(mpg ~ disp + hp + wt + am, data = mpgRelation)
summary(modelRelate)    ##Calls formula and significance of variable to mpg
confint(modelRelate)    ##Calls confidence interval on the coeficients
```

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.

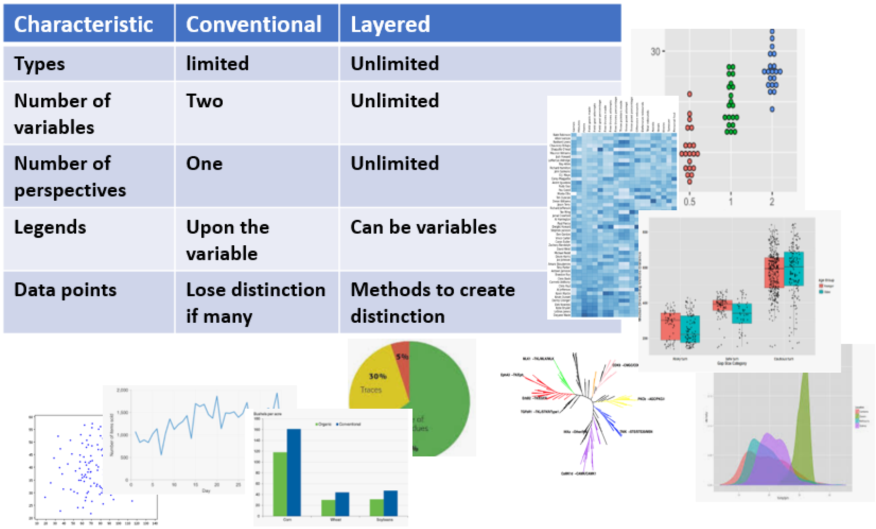
The book defines layered charting as presenting information in layers. The difference can be seen in Figure 2-10. Traditional and layered charting are shown side-by-side.

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.



Characteristic	Conventional	Layered
Types	limited	Unlimited
Number of variables	Two	Unlimited
Number of perspectives	One	Unlimited
Legends	Upon the variable	Can be variables
Data points	Lose distinction if many	Methods to create distinction

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.

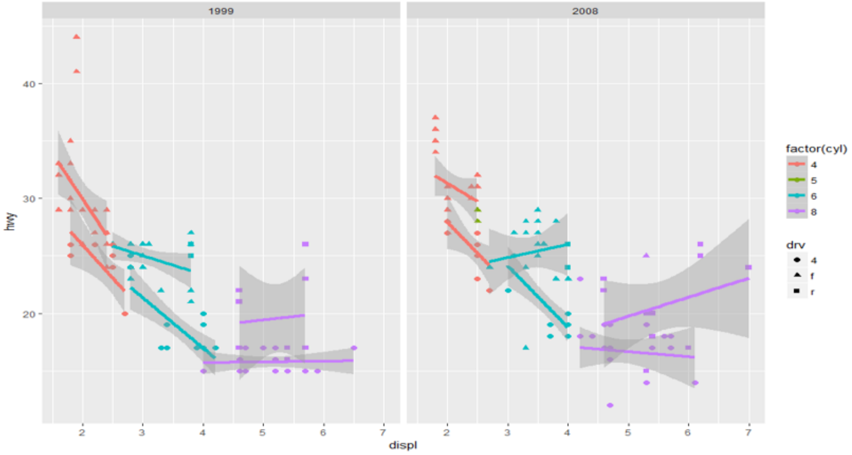


Figure 2-11: Layered charting to present measures of performance.

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

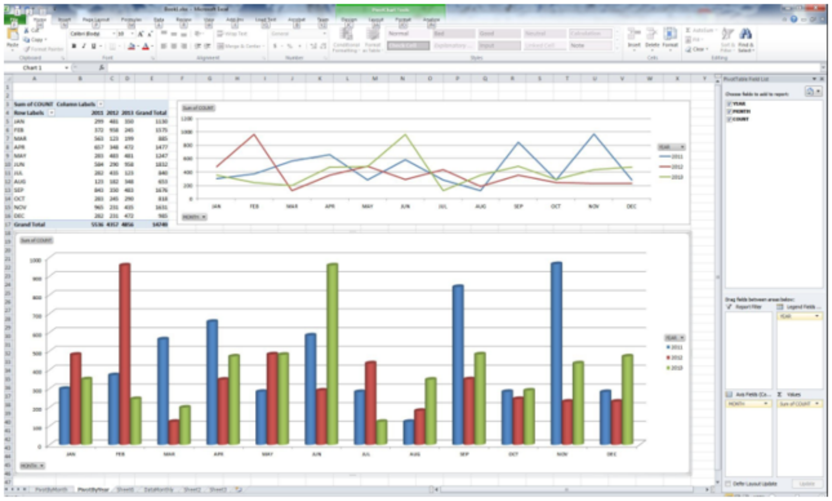
System reports are the insights that our systems have been configured to give us. The number of asset management organizations for which system reports are the extent of insight is shrinking.

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.



Sum of COUNT	Column Labels			
Row Labels	2011	2012	2013	Grand Total
JAN	299	481	350	1130
FEB	372	958	245	1575
MAR	563	123	199	885
APR	657	348	472	1477
MAY	283	483	481	1247
JUN	584	290	958	1832
JUL	282	435	123	840
AUG	123	182	348	653
SEP	843	350	483	1676
OCT	283	245	290	818
NOV	965	231	435	1631
DEC	282	231	472	985
Grand Total	5536	4357	4856	14749

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

2.3.2. Know-Thy-Data Insight

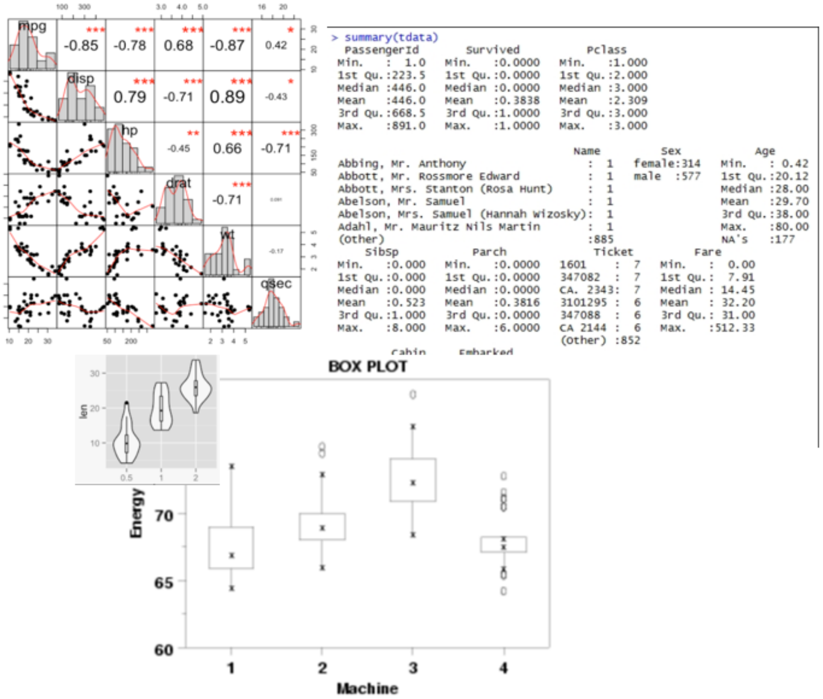
One-time and periodic inspections of process data often beget deep insight. One reason is that the persistent compliance to operational processes is readily confirmed through its data. Another reason is that an initial and unfolding inspection of data often reveals hidden realities, findings contrary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



```
> summary(tdata)
  PassengerId       Survived          Pclass
 Min.   :  1.0   Min.   :0.0000   Min.   :1.000
 1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000
 Median :446.0   Median :0.0000   Median :3.000
 Mean   :446.0   Mean   :0.3838   Mean   :2.309
 3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000
 Max.   :891.0   Max.   :1.0000   Max.   :3.000

                                    Name         Sex           Age
 Abbing, Mr. Anthony                  :  1   female:314   Min.   : 0.42
 Abbott, Mr. Rossmore Edward          :  1   male  :577   1st Qu.:20.12
 Abbott, Mrs. Stanton (Rosa Hunt)     :  1                Median :28.00
 Abelson, Mr. Samuel                  :  1                Mean   :29.70
 Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00
 Adahl, Mr. Mauritz Nils Martin       :  1                Max.   :80.00
 (Other)                              :885                NA's   :177
     SibSp           Parch             Ticket          Fare
 Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00
 1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91
 Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45
 Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20
 3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00
 Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33
                                  (Other) :852
        Cabin        Embarked
```

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

2.3.3. Modeled Insight

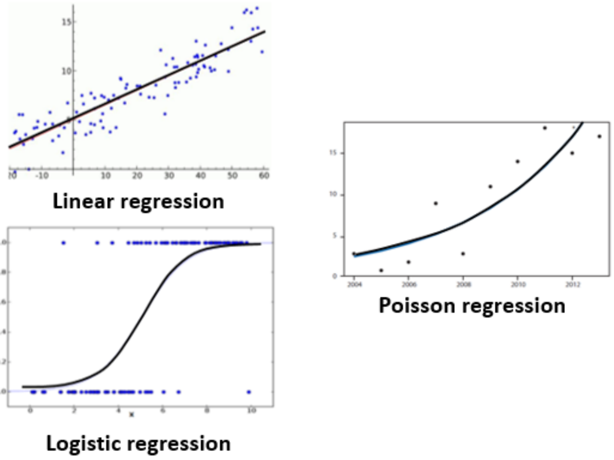
The final category of insight deliverables, as named by this book, is modeled insight. Insight is gained as the data flows through analytic models for machine learning and AI.

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

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.



Linear regression

Poisson regression

Logistic regression

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 regressiondeals 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.

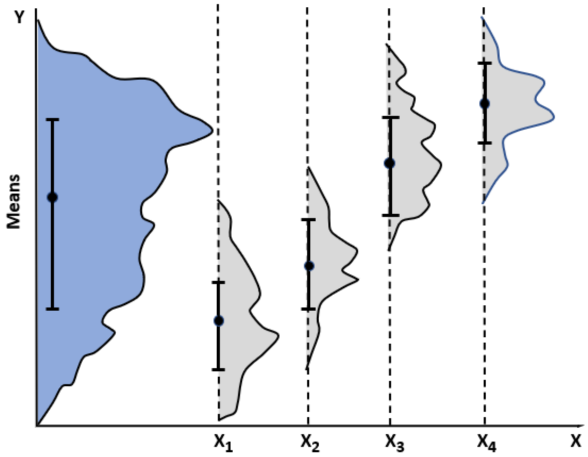


Figure 2-15: The concept of analysis of variance **(ANOVA)**

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.

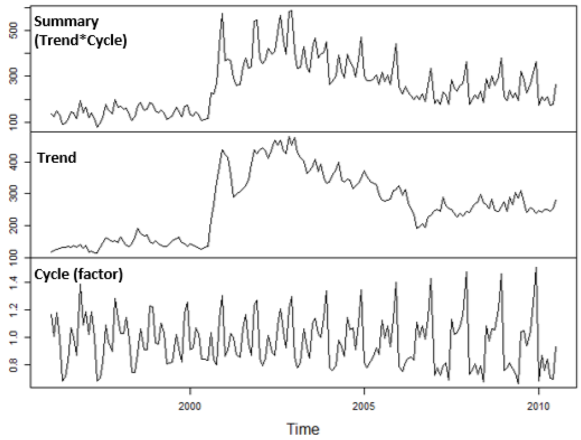


Figure 2-16: Cycle removed from the summary level to reveal the trend of the series.

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.

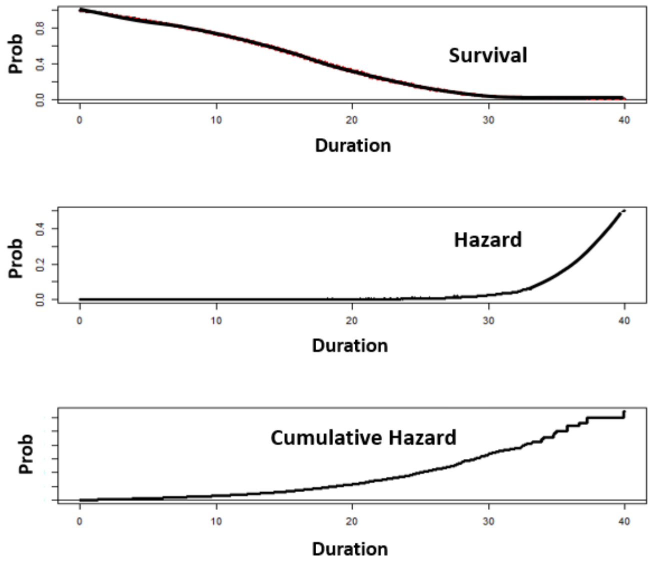


Figure 2-17: The three perspectives of duration analysis.

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.

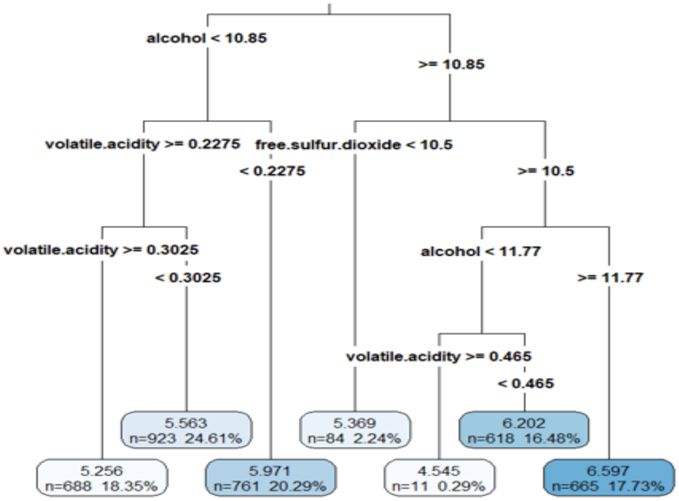


Figure 2-18: Decision tree with numeric outcomes.

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.

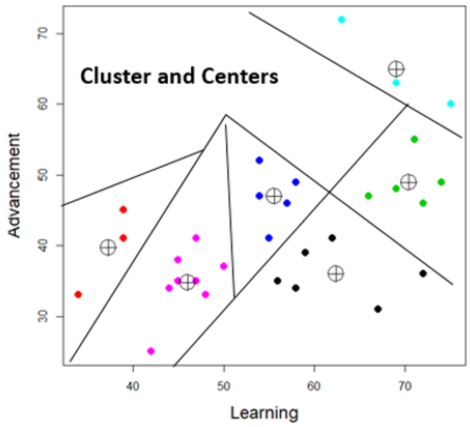


Figure 2-19: Six hidden variables teased out two input variables.

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.

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

Data and Analytics Skills for Your Career Security

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

A look inside

Richard G. Lamb

Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Data from process tasks conducted with Ex
Pivot as dashboards
Tables and calc
Convention
ught deliverables
Process step
descriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

Your career security will leap forward when you learn to work with MS Access to build what the book calls super tables. Access makes building super tables easy with the intuitive tabular interface to SQL called Query by Example. The skills learned transfer to any software, e.g., Power BI. And all software have the functionality to import your super tables directly from Access.

Chapter 3: Super Tables from Operational Data

Excerpts:

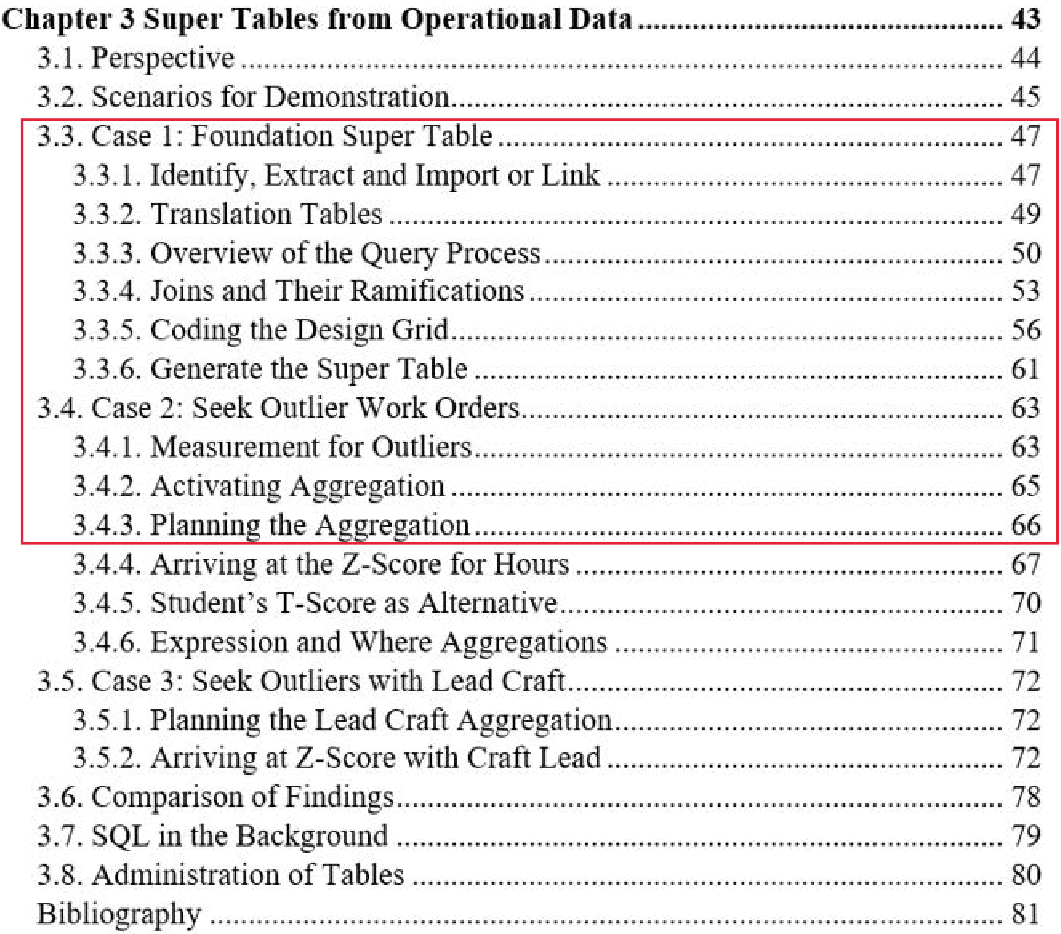
3.3. Case 1: Foundation Super Table

3.4. Case 2: Seek Outlier Work Orders (partial)

Additional “Look Inside” at https://analytics4strategy.com/book-look-inside

Book is available from Amazon

Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds



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 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.

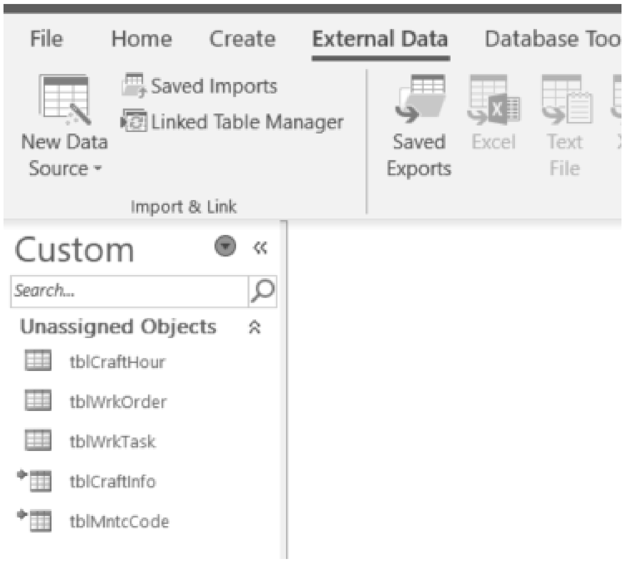


Figure 3-1: Three downloaded tables and two clarifying tables.

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.

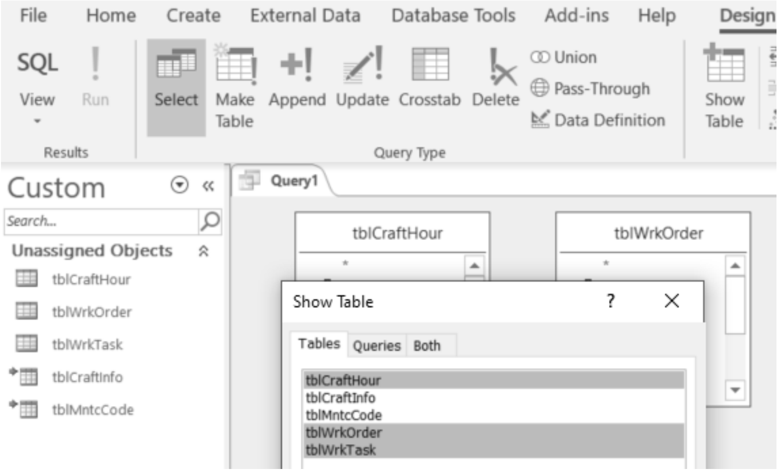


Figure 3-2: Tables imported and linked to Access for building super tables.

Notice that there are six types of queries. Most work is with select queries. The others have purposes supplemental to the select query. The query types will be explained as the chapter unfolds.

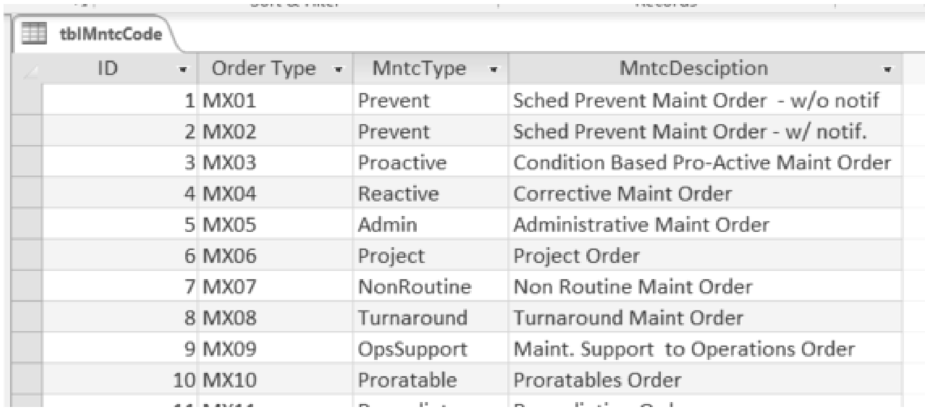
The pop-up shows the five tables. We can highlight the tables we want in the super table and click Add (not shown in the figure). Alternately, we can drag the tables listed in the left-most view of available objects into the work area.

Another point, we are not restricted to table objects. We can also pull queries into a query.

3.3.2. Translation Tables

Two of the five tables in the object view are what the book calls translation tables. They can be used to clarify variables in the final super table and, thus, insights delivered from it. They can be used to cleanse data. They can be used to create uniformity between organizational entities. I have used them to join the data of two CMMS generations, when the supplanting system would have otherwise left the organization without contiguous historical data.

It is an easy, down-dirty method. To explain the concept, Figure 3-3 shows the table, tblMntcCode, to translate arcane order type code to plain English.



tblMntcCode

ID	Order Type	MntcType	MntcDesciption
1	MX01	Prevent	Sched Prevent Maint Order - w/o notif
2	MX02	Prevent	Sched Prevent Maint Order - w/ notif.
3	MX03	Proactive	Condition Based Pro-Active Maint Order
4	MX04	Reactive	Corrective Maint Order
5	MX05	Admin	Administrative Maint Order
6	MX06	Project	Project Order
7	MX07	NonRoutine	Non Routine Maint Order
8	MX08	Turnaround	Turnaround Maint Order
9	MX09	OpsSupport	Maint. Support to Operations Order
10	MX10	Proratable	Proratables Order

Figure 3-3: Translation table for order type.

Translation tables can be built and updated inside Access or elsewhere such as Excel. The explanation will assume the Excel strategy. To build, update and upgrade them in Access, the readers are referred to Chapter 3 of Alexander and Kusleika. Once built, they can be maintained and made available to any super table.

The Excel process begins by building a list of all existing categories in Excel. The list is usually taken from the operating system or systems.

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:

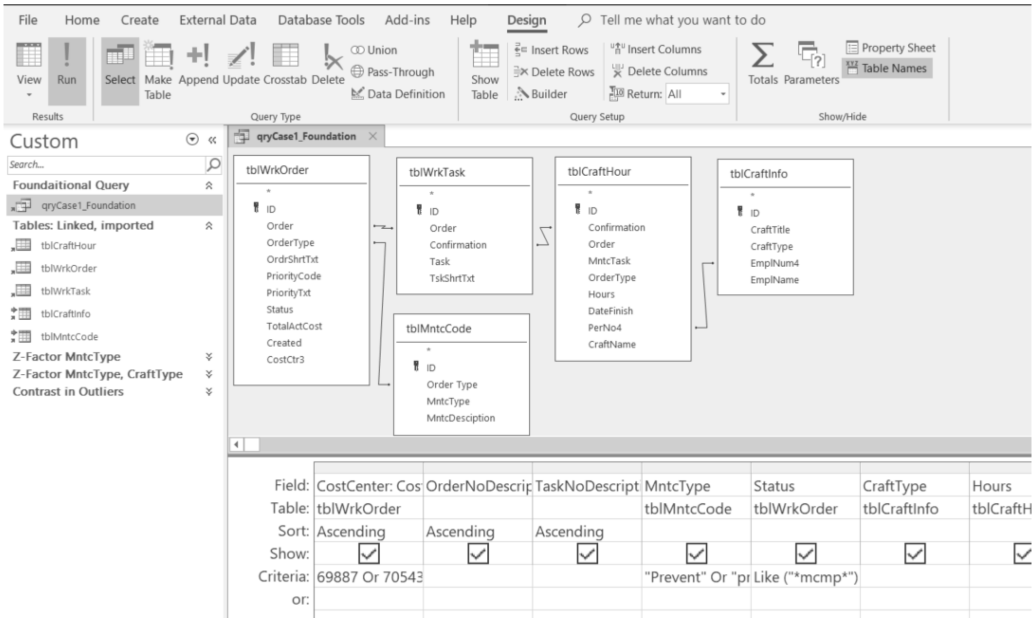
- **Select:** Builds the super table from one or more subtables. Aggregation is constructed in the select query.
- **Make Table:** Converts a select query to a “hard” table.
- **Append:** Adds rows of data to an existing table.
- **Update:** Changes rows to variables in a table or query.
- **Delete:** 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.



Field:	CostCenter: Cos	OrderNoDescrip	TaskNoDescript	MntcType	Status	CraftType	Hours
Table:	tblWrkOrder			tblMntcCode	tblWrkOrder	tblCraftInfo	tblCraftH
Sort:	Ascending	Ascending	Ascending				
Show:	☑	☑	☑	☑	☑	☑	☑
Criteria:	69887 Or 70543			"Prevent" Or "pr	Like ("*mcmp*")		
or:							

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.

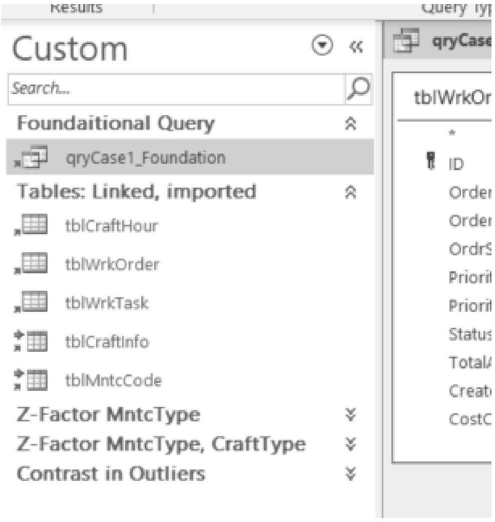


Figure 3-5: List of objects associated with the super table.

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.

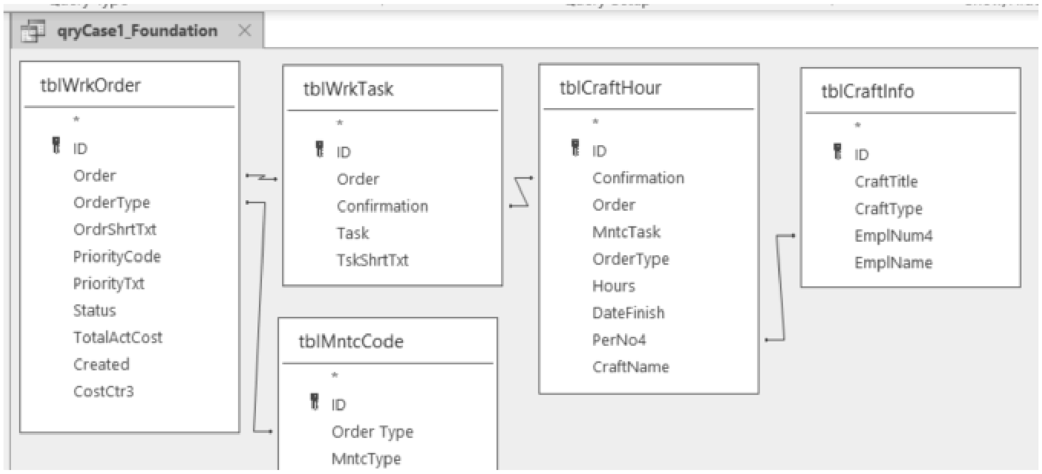


Figure 3-6: All five tables to the super table and their joins.

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.

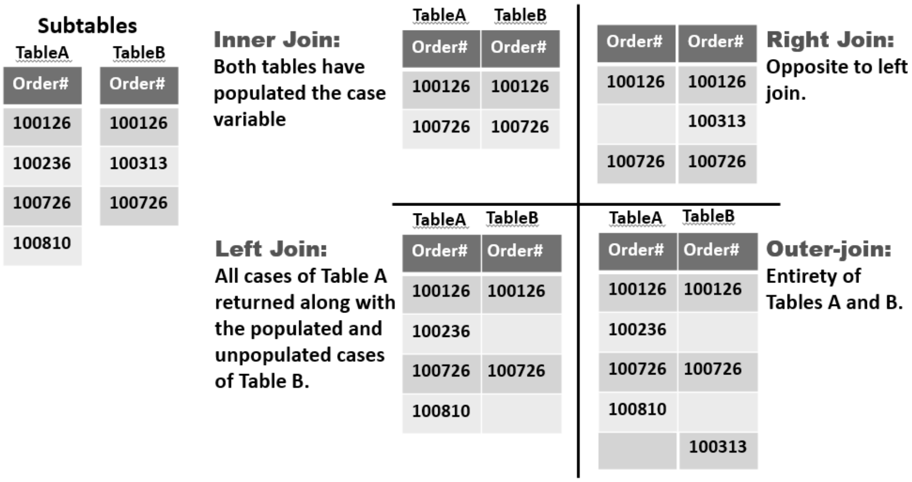
3.3.4. Joins and Their Ramifications

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.



Subtables

TableA	TableB
Order#	Order#
100126	100126
100236	100313
100726	100726
100810	

Inner Join: Both tables have populated the case variable

TableA	TableB
Order#	Order#
100126	100126
100726	100726

Right Join: Opposite to left join.

Order#	Order#
100126	100126
	100313
100726	100726

Left Join: All cases of Table A returned along with the populated and unpopulated cases of Table B.

TableA	TableB
Order#	Order#
100126	100126
100236	
100726	100726
100810	

Outer-join: Entirety of Tables A and B.

TableA	TableB
Order#	Order#
100126	100126
100236	
100726	100726
100810	
	100313

Figure 3-7: Tables can be joined to return four different outcomes.

The point of the left and right join is that one table is the basis of the other. When working with them, it is good practice to confirm we are getting what we want rather than the opposite. We make the confirmation by running and inspecting the returned super table.

The nature of the outer join is apparent in the figure. Notice that all cases are returned. Non-matched cases to both tables are shown as empty (null) cells.

Joins have ramifications for know-thy-data, audit and control. This is because our attention is drawn to a simple question. Why are there empty cells? When building and molding a super table it is good practice to vary the joins and look for nulls in the returned table. We often discover something.

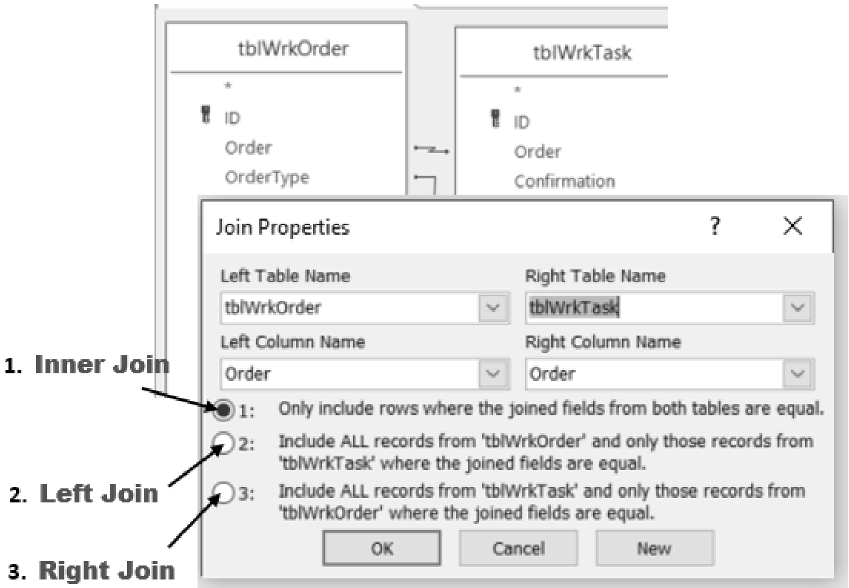
Figure 3-8 shows how we set the joins between tables with Access. The process begins with a right-click on a join line in Figure 3-6 and selecting the Join Properties option. The Join Properties window shown in Figure 3-8 will pop up for specifying the join we want.

In the figure, notice that the first option is an inner join. It is also the default. It is not necessary to work in the window if the intended join is an inner join.

In the pop-up, notice that Access regards tblWrkOrder as the left object to the join and tblWrkTask as the right object. It shows that the joining variables are the order number variable in each.

If we want a join other than the inner join, we select one of the two remaining alternatives—left and right. However, notice there is no option for an outer join. The absence is one of the few disappointments of Access.

However, there is an easy work around for creating an outer join. Do a left or right join. Thence, append it to the empty (null) variable rows to the opposite join. We would 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.



Left Table Name: tblWrkOrder
Right Table Name: tblWrkTask
Left Column Name: Order
Right Column Name: Order

● 1: Only include rows where the joined fields from both tables are equal.
○ 2: Include ALL records from 'tblWrkOrder' and only those records from 'tblWrkTask' where the joined fields are equal.
○ 3: Include ALL records from 'tblWrkTask' and only those records from 'tblWrkOrder' where the joined fields are equal.

1. Inner Join
2. Left Join
3. Right Join

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	OrderNoDescript ion: [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			

Field	MntcType	Status	CraftType
Table	tblMntcCode	tblWrkOrder	ttblCraftInfo
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 (column) makes no sense. It would mean we want records that are a combination of the four cost centers. We would get a table with no records.

Figure 3-9 provides a list of criteria expressions for all data types. Connecting back to the Or operator, if we want to apply more than one criteria expression to a variable, we would insert the Or operator between each expression.

Criteria name	Code	Effect
Equals	“x” for text; x for numeric.	Searches for values equal to x
Does Not Equal	Not in (“x”) for text. Not in (x) for numeric.	Searches for all values except those equal to x
Null	Is Null	Searches for empty fields
Not Null	Is Not Null	Searches for non-empty fields.

Figure 3-9: Criteria expressions for all data types.

The And condition exists between the columns of the design grid. Collectively, the criteria across the grid define the records to remain in the super table. Figure 3-4 and Code 3-2 showed the criteria that have been coded for 4 of the 11 variables. In the super table, they collectively set filters for cost center, maintenance type, status and dates created.

Although not necessary for Case 1,what if we want an Or case between variables in the grid? The answer is the Or row of the grid. We can create groups with different And criteria between the variables. To do it, each group would be coded in a separate Or row.

The next two variables in Code 3-2—OrderNoDescription and TaskNoDescription—return a combination of variables as a single variable. Additionally, the code demonstrates some other important basics to building and molding super tables.

Let’s look at the first of the two variables. The purpose of the code is to combine the order number and text description variables in a single variable. An example is the order and description variables coded to return a cell such as 6000937432: Pump1871 seal leaking.

Notice the piece of code to the variable—[tblWrkOrder]**.**[Order]. The code is necessary because a variable named Order occurs in more than one sub table. When dragged into the grid, Access will tell us that it is not able to know which one we want until we provide additional information. The additional information is the source table as identified within the first pair of square brackets. The variable is identified within the second pair. Finally, to make it work, notice that ‘.’ (dot) is placed between the table and field brackets.

Next notice the & operator. The geek word for it is the concatenation operator. By placing the operator between two variables, they are returned as a single (concatenated) variable.

However, in this case we want some punctuation and spacing to make the variable more readable. This is done with the code—“: “ (colon space)—between the concatenation operators. The parentheses are required because the colon and space are text rather than numeric. Note that it is typical to all codes that text be bracketed with parentheses and the numbers be absent of parentheses.

Now let’s look at the next three variables in the grid of Figure 3-4 and Code 3-2—MntcType, Status and CraftType. All three are simple drags. MntcType includes a criteria using the Or operator. The code causes the table to return 3 types of maintenance from the 17 categories that are inherent to the data extracted from the CMMS.

Notice the parenthesis bookends to the desired types using the Or operator. Except the parentheses, this is the same pattern as seen previously for the CostCenter criteria. This is because the maintenance type variable is text rather than numeric. In contrast, cost center is a variable of numeric values.

Next notice the code—Like(“*MCMP*”)—as the criteria to the Status variable. It is called the Contains criteria. We could have coded the Equals criteria—“MCMP”—that was included in Code 3-2.

However, upon inspecting the data, some of the status cells to the variable were found to be populated with multiple statuses. The Equals criteria would not include them in a table, but the Contains criteria would.

Among the raw data, the pattern was observed as occasionally including MCMP. Although a check of the forming super tables did not find the pattern, we want a criteria to recognize the pattern if it should arise in future monthly updates.

Figure 3-10 provides the list of available text criteria. We all know of many or all of them from our experience with Excel and other software.

Criteria name	Code	Effect
Contains	Like (“*x*”)	Searches for all values that contain x
Does Not Contain	Not like (“*x*”)	Searches for all values that do not contain x
Begins With	Like (“x*”)	Searches for all values beginning with x
Ends With	Like (“*x”)	Searches for all values ending with x
Comes After	>= “x”	Searches for all values that come before x in alphabetical order
Comes Before	<= “x”	Searches for all values that come after x in alphabetical order

Figure 3-10: Criteria for working with text strings.

The next three variables in the design grid are Hours, DaysPastCreated and Created. Two additional and commonly occurring codes can be spotted in the design grid. One demonstrates the creation of calculated variables. The other is to specify date ranges.

The code in the Field row—DaysPastCreated: [DateFinish]-[Created]—is a calculation we have given the name DaysPastCreated. The big point is that any calculation is possible—simple or complex. There is no great magic. The expressions of our Excel experience and experience with other software will often transfer. If what we are attempting upon our experience does not work, we can query the internet for a solution—query for “access expressions.”

Even though they have not been called for the super table, two expressions will be introduced here because we find ourselves frequently using the first of them. They are the conditional expressions—IIF() and Switch(). They are coded as shown in Figure 3-11.

Function name	Code	Effect
If then	IIF(logical test, value if true, value if false)	Evaluates a specific condition and specify results whether the condition meets True or False values
Switch	Switch(logical test1, value1, logical test2, value2, ... logical test_n, value_n)	Evaluates a list of paired expressions and returns a value or an expression associated with the first expression in the list that is True

Figure 3-11: The conditional expressions IIF() and Switch().

It is apparent from the table that the first IIF(), is familiar to most of us. Alternately, Switch() allows us to avoid IIF() expressions that get messy when we want to nest an IIF() within an IIF() and so on.

For the variable, Created, we are working with dates. Notice that # (pound) bookends the dates. This differs from the bookends for text and numeric variables. The expression calls for all records within and including the dates.

This is a good place to introduce the body of criteria for bounding date and numeric variables. Figure 3-12: shows them for date variables and Figure 3-13 for numeric variables.

Criteria name	Code	Effect
Between	Between “#mm/dd/yyyy#” and “#mm/dd/yyyy#”	Searches for dates that fall between the specified dates
Before	< “#mm/dd/yyyy#”	Searches for dates before a certain date
After	> “#mm/dd/yyyy#”	Searches for dates after a certain date
Today	=Date()	Searches for all records containing today’s date
x Days Before Today	<=Date()-x	Searches for all records containing dates x or more days in the past

Figure 3-12: Criteria for working with date variables.

Criteria name	Code	Effect
Between	Between x and y	Searches for all values in the range between x and y
Less Than	< x	Searches for all values smaller than x
Less Than or Equal To	<=x	Searches for all values smaller than or equal to x
Greater Than	> x	Searches for all values larger than x
Greater Than or Equal To	>=x	Searches for all values larger than or equal to x

Figure 3-13: Criteria for bounding numeric variables.

Now for the final two variables to the super table. They are Order and Confirmation.

Including the two variables in the super table is a proactive practice rather than necessary for the foundation super table. As we work along to build the super table, we may find that we need them.

Another reason to include them is that we may find ourselves building other super tables with the foundation super table. It is very likely that the need for the inherent identifiers will arise.

With qryCase1_Foundation, we could join on the concatenated variable. Or we could tease the order number out of the concatenated variable with string functions. However, it is good practice to bring along the untouched inherent identifiers.

In the practice of building super tables with Access, we will find that the demonstrated and additional referenced code in this section covers most of everything we would ever call for. We will also recognize that most of what has been introduced matches or is very similar to what we know from working with Excel spreadsheets and other software.

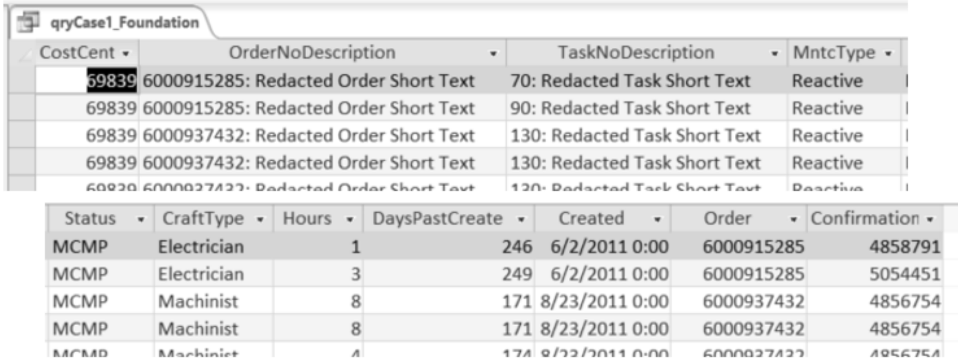
This brings us back to a point made earlier. Building super tables is more a matter of creativity with an insight deliverable in mind than it is to be a data geek. In fact, the skills of data science have little meaning to us normal people working with our data to reach an insight. We can function quite nicely and still be largely ignorant of data science beyond what we already know as modern role holders.

3.3.6. Generate the Super Table

Now to generate the foundation super table. However, rather than a final event, we should flip between the design and table views as we develop the table. One reason is to confirm, at each step, that we are getting what our code was intended to give us. Another reason is that we will often find ourselves following discoveries that take us to new insights and revelations.

Notice the table icon at the far left of the ribbon shown in Figure 3-4. It returns the super table as shown in Output 3-1. The figure cuts the table in two pieces and shows only the first 4 of its 687 rows. Notice the variables to the table are as we named them.

This is a good place to introduce a functionality with which to confirm and explore the table. Notice the Totals option in the Records section of the ribbon (not shown in the figure). When selected, a Totals row will appear at the bottom of the table view. The row cell to each variable offers a pull-down menu of options for summarizing the variable. For numeric variables the options are sum, average, count, maximum, minimum, standard deviation and variance. For text variables it is count.



qryCase1_Foundation

CostCent	OrderNoDescription	TaskNoDescription	MntcType
69839	6000915285: Redacted Order Short Text	70: Redacted Task Short Text	Reactive
69839	6000915285: Redacted Order Short Text	90: Redacted Task Short Text	Reactive
69839	6000937432: Redacted Order Short Text	130: Redacted Task Short Text	Reactive
69839	6000937432: Redacted Order Short Text	130: Redacted Task Short Text	Reactive
69839	6000937432: Redacted Order Short Text	130: Redacted Task Short Text	Reactive

Status	CraftType	Hours	DaysPastCreate	Created	Order	Confirmation
MCMP	Electrician	1	246	6/2/2011 0:00	6000915285	4858791
MCMP	Electrician	3	249	6/2/2011 0:00	6000915285	5054451
MCMP	Machinist	8	171	8/23/2011 0:00	6000937432	4856754
MCMP	Machinist	8	171	8/23/2011 0:00	6000937432	4856754
MCMP	Machinist	4	174	8/23/2011 0:00	6000937432	4856754

Output 3-1: The table returned from the code.

The super table can be imported into Excel or another Access file. In fact, all modern software has standard functionality to import an Access super table as their source data. For Excel and Access, go to the Get Data icon and follow the guided trail. Of course, the super table can be developed further in all final destinations.

All software, including "R," provide the option to link to the super table in Access. Linking is often preferred over importing.

Imagine working with the table in Excel Pivot. We often discover that we want to modify the super table for a host of insightful reasons. We can pop the Access hood on the super table query and make the upgrades.

In turn, the background data to our Pivot will be refreshed upon command or the next time the Pivot file is opened. If we are doing a monthly update, the link will update the Pivot whenever it is opened for a session or refreshed during a session. Consequently, the analyst does not need to distribute the Pivot report upon every update.

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 the 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:

Z-Score Standardized = (R - Avg)/Std Dev (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.

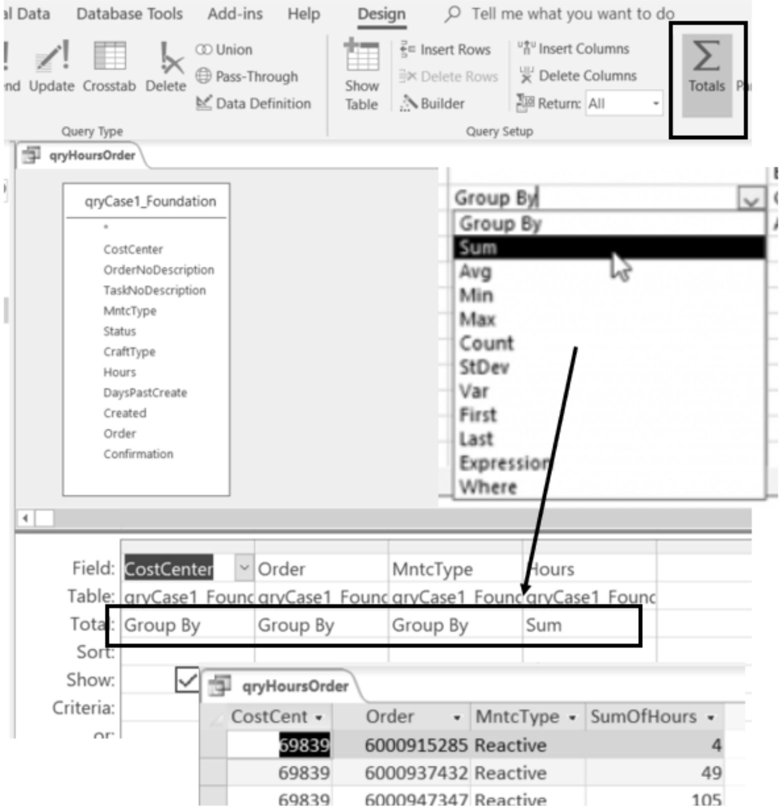
If we were interested in orders beyond the two tails of the distribution, we would look for orders with a Z-score equal to or greater than 1.96 absolute. It is called the two-sided test.

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.



Field:	CostCenter	Order	MntcType	Hours
Table:	qryCase1_Founc	qryCase1_Founc	qryCase1_Founc	qryCase1_Founc
Total:	Group By	Group By	Group By	Sum
Sort:				
Show:	☑			
Criteria:				
or:				

qryHoursOrder

CostCent	Order	MntcType	SumOfHours
69839	6000915285	Reactive	4
69839	6000937432	Reactive	49
69839	6000947347	Reactive	105

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_Foudation	qryCase1_Foudation	qryCase1_Foudation	qryCase1_Foudation
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.

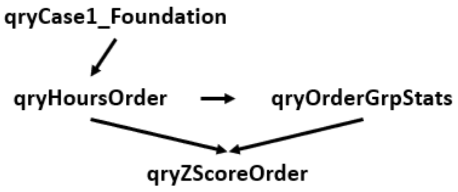
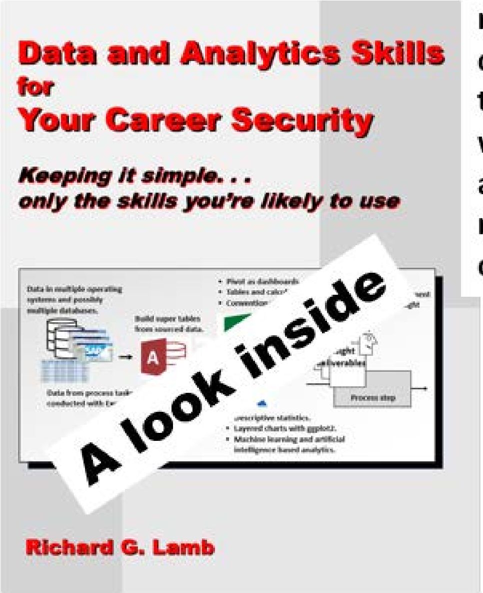


Figure 3-16: Case 2 flowcharted to the final insight deliverable.

From the super table, qryCase1_Foundation, as its input, we will develop the qryHoursOrder object. It was previously shown in Figure 3-16 and followed with an explanation of its code.

Thence, we will develop a table of statistical elements to the Z-score computed with Equation 3-1—qryOrderGrpStats. The query will return the group averages and standard deviations.

Finally, qryHoursOrder and qryOrderGrpStats will be joined to generate qryZScoreOrder. The object will compute the Z-score for each order and return only those with a score equal or greater than 1.65.



For us regular stiffs, it is easiest to learn the R software in the context of conducting analytics rather than generally. The book explains and demonstrates mainstream analytics and, in turn, explains the code to do them. While working through the book, the reader will amass the competency in the R code they need rather than dive into a bottomless pit that coding can be.

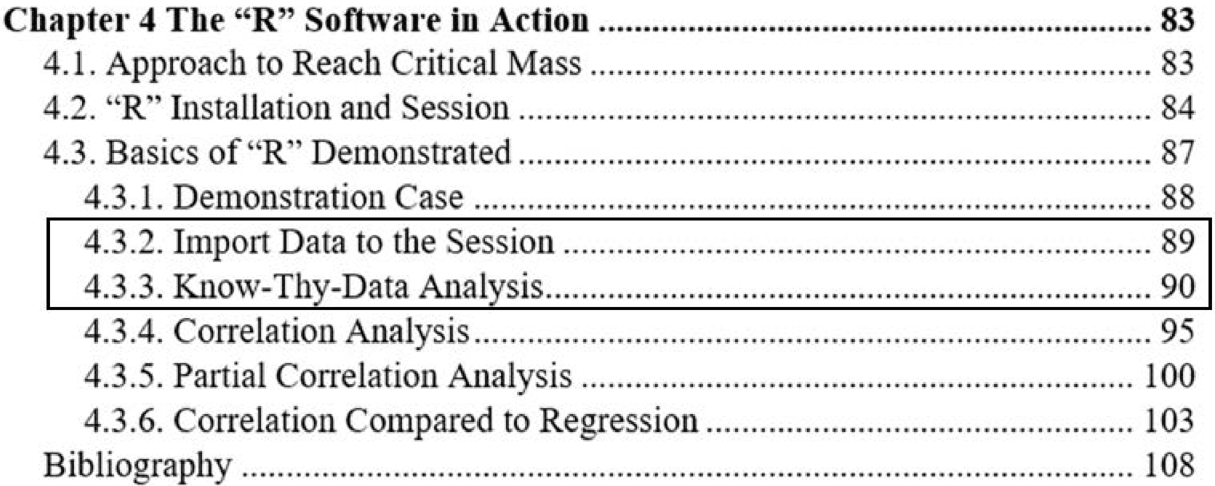
Chapter 4: The R Software in Action

Excerpts:

4.3.2. Import Data to the Session

4.3.3. Know-Thy-Data Analysis

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside
Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds



4.3.2. Import Data to the Session

The first action is to import our data into the “R” session. The code to do so is:

```
TaskData <- read.csv("C:\\<PATH>\\ScoreComplexPrep_Dataset.csv",
   header=TRUE, sep=",")
```

We are showing the case in which our data, a super table, is imported into “R” with the `read.csv` function. The function is used because the super table is a file of comma-separated variables from Excel.

However, there are functions for other types of files. They are `read.xlsx` for .xlsx files and `read.delim` for .txt and .dat files. For Access files we would install and open the RODBC package and set up the location and arguments for importing a query or table from Access.

There are three arguments in the `read.csv` function just as there are for the other read-type functions. The first is the file and its location. The second tells the function if the super table has headers. The third indicates commas or something other as the separator.

The second and third arguments are shown for completeness. However, they are the default arguments. The arguments need not be coded unless our data are exceptions to them.

Let’s talk about the file and location argument. There are options in “R” to identify the files with respect to a “working directory” rather than code out the entire location. However, the full address is shown in the code.

This is preferable because it places a paper trail in the code. This makes it simple to locate the file after we have long forgotten its name and location or to make it possible for recipients of the analytic to find the data without knowing of some working directory and file. In the code, `<PATH>`, represents the complete address for the reader to fill in.

Also note the syntax of the file argument. The location string for Microsoft typically codes with backward slashes. In “R” they must be changed to either a double backward slash as shown in the code or a forward slash.

Another fundamental. Notice the code string “`<-`“ in the code line. It assigns what is created by the code to its right to be the “object” to its left. By virtue of the element in the code, the imported table, ScoreComplexPrep_Dataset.csv, is now an object called TaskData. Notice that throughout the code we refer to the object rather than the csv file.

A comment. Tables can be built from scratch in “R” with standard query language (SQL) by using the sqldf function. Beyond what is possible with SQL, we can do much more with “R” code.

As mentioned in Chapter 2, building super tables with SQL and additionally with R coding requires substantial skills. As a measure, the skills for each require six weeks of structured formal training. Therefore, because Access moves SQL coding to the background, we are better served by building super tables in Access and subsequently importing them to R. When additional treatment is called for, the go-by that is being followed for the subject analysis will provide the necessary code.

The data of one or more imported super tables may be manipulated in the conduct of an analytic. However, the code will typically be such that the imported data will remain unchanged—as TaskData in this case.

Although not demonstrated, the tables built with an analytic can be exported with the write function. The reader can easily query the internet for the code which is like the functions to read data into the session. We may do so to make our super table more super. For example, we would likely want to join the findings of a classification analytic to the super table.

4.3.3. Know-Thy-Data Analysis

This section will demonstrate a range of descriptive, statistical and graphic insights—know-thy-data. What is demonstrated should be basic to all data to be engaged in insight. It will be apparent that doing so is an easy task of importing the data and subjecting it to the functions of this section as a go-by or tool.

The section will look at data along three dimensions. First is to view the content and type of the variables. Second is to view the data per statistical measures. And third is to view the data graphically.

As always, there are functions to the three dimensions. For the first, we want to be sure we know what is in the table. Three functions immediately come to mind. One, the head function, will return the first six rows (records). The other shoe to the function is the tail function to return the last six rows. We are not limited to six rows. The argument n= allows us to specify a different number of rows.

The str function, shows the number and nature of the variables and number of records. Its information complements the insight provided by the head function.

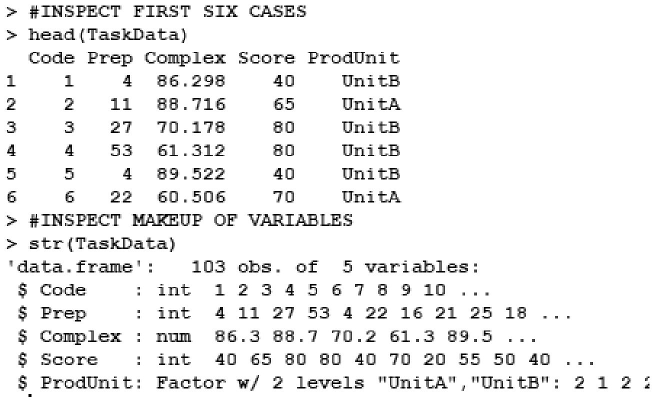
Both are returned by the following lines of code and the outputs are shown in Output 4-1:

```
#INSPECT FIRST SIX CASES
```

```
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
```

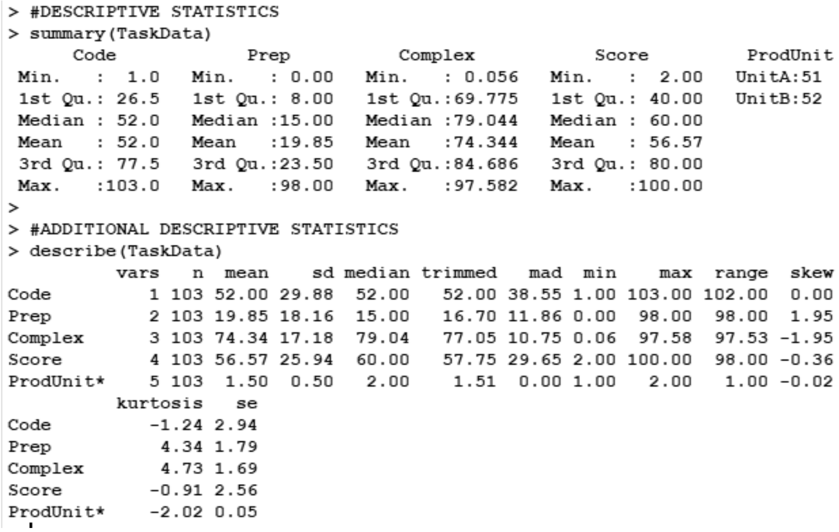
Output 4-1: Head and summary views the data.

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.

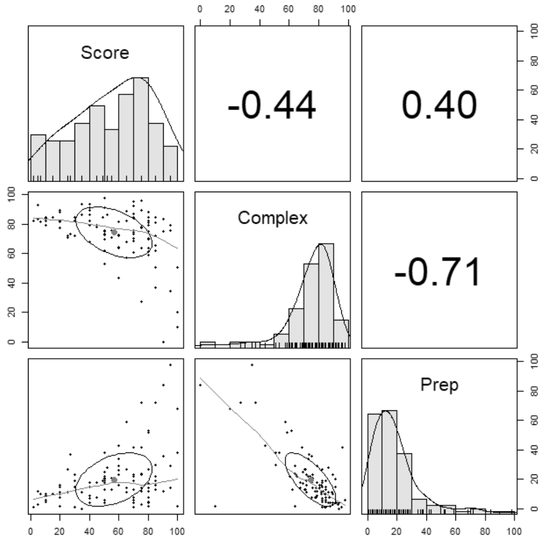
First let's look at the graphic from the `pairs.panel` function of the psych package for a tremendous amount of insight. In it, we can inspect the shape or distribution of each variable in the dataset. We can inspect the cross-plot relationship of each variable with all other variables. The fitted smooth 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.

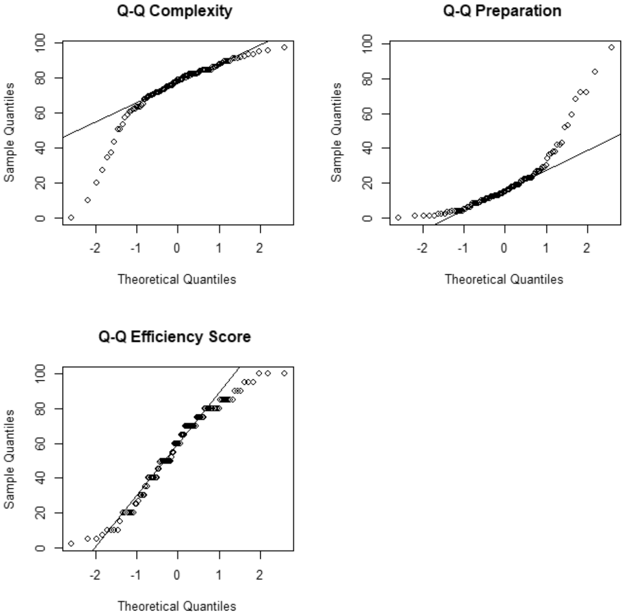
We can test normalcy with graphic insight. However, we should note that the `shapiro.test` function can be used to test if the data is significantly different from the normal distribution.

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 the 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.

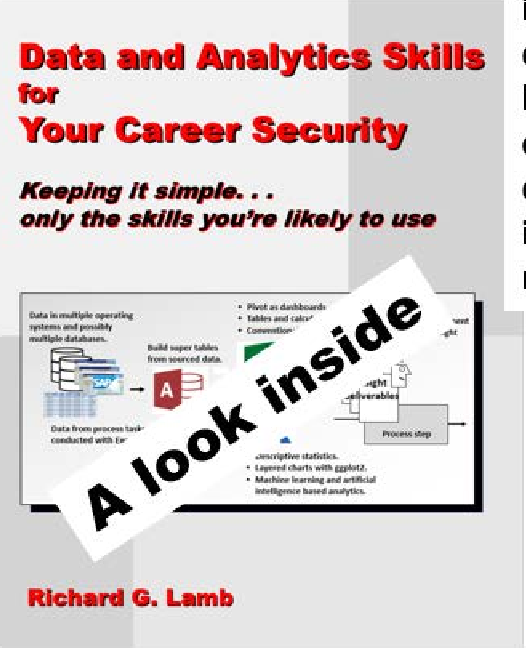
If we highlighted and ran the `qqnorm` code, we would get the plotted of points. If we did the same with `qqline` code, we would get a straight line. It represents how a plot of data with the mean and variance of the data should form as quantiles if it were a normal distribution.

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.

There is another element to recognize in the code. It is one of the ways we can specify variables to functions.

Notice the syntax `TaskData$Complex` in the code to the first graphic. Also notice the same pattern in the code in which the other two variables are engaged for evaluation. The code line tells the function that we are working with a variable (after the $) from a table (before the $).

There are other ways to identify variables. Some will arise as the chapter progresses. However, the advantage of the `table$variable` syntax is that we leave a paper trail in the code.



Layered charting integrates perspectives as layers in a single visualization without any one issue being lost in muddle. The layers can be different types of charts and legends, and can be built on different datasets. On or more layers can be the output of an analytic such that the datasets are of the analytic rather than an imported dataset. The only rule is that all must relate the same base axes.

Chapter 5: Layered Charting to Know Thy Data

Excerpts:

5.2.2. Test for Normal Distribution

5.2.3. Inspect Correlation Between Variables

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside
Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds

Note:

The advance graphics created with the ggplot2 package of the R software depend heavily on color to communicate the insight they are coded to give us. Some such graphics appear in the two excerpts.

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.

For convenience, the herein two excerpts will present the color version.

5.2.2. Test for Normal Distribution

We should test the numeric variables in our dataset for normal distribution. This is especially so if the variables are to be used in measures and models. Furthermore, a tested variable can lead us to recognize the levels to our categorical variables with respect to the numeric variable that are individually homogenous but heterogenous across the distribution. As said in measurement and analytics, “subset, subset and subset.”

An example to a maintenance and reliability operation is a non-normal distribution of a numeric variable such as hours per order. Our inspection may find that we should search out the levels to our categorical variables that have disparate relationships to hours. Insights may be hidden in the variables such as maintenance type, priority and craft type.

This section will demonstrate how to test a numeric variable for normal distribution. It will then demonstrate methods for inspecting the categorical levels as subsets to the numeric variables for normality.

The R script to the section includes the visual and statistical test of normal distribution for the hwy, cty, displ and cyl variables. However, we will only follow the case for the hwy variable because the process is the same for all variables.

The code below returns Output 5-9. The figure compares the variable, hwy, against a template of normal distribution.

```
#Highway compared to normal distribution
#Hwy variable
qqhwy<- ggplot(data = mpgKtd,
               mapping = aes(sample = hwy)) +
        stat_qq_band() +
        stat_qq_line() +
        stat_qq_point(aes(color=class)) +
        labs(x = "Theoretical Quantiles",
               y = "Sample Quantiles") +
        ggtitle("Q-Q Test of the Hiqhway Variable")
qqhwy
```

Let's explore the code for the basics of ggplot2. The graphic is assigned to the `ggplot2` object, `qqhwy`. We call the figure up with the final line to the code.

As was introduced in Section 2.2.3, the code creates a layered chart. Notice the "+" code inserted between the base plot, `qqplot`, `stat_qq_band` and so on. At each occurrence a layer is stacked on the base plot or overrides its defaults.

The first element, `ggplot`, creates the base plot. It established the data and axes of the graph. Within it are what are called the aesthetics. In this case a single variable, hwy. However, we will step past an esoteric explanation of "aesthetic."

The remaining elements place layers over the base plot. The first three stack individual graphs as layers upon the base plot. The fourth, `labs`, overrides the default axes titles—x and y. The fifth, `qqtitle`, stacks a plot title on the base plot.

Let's dig deeper into the ggplot element. We assign the data with the argument `data =`. We set the variables and aesthetics of the graph with the argument `mapping =`.

However, the point to note is that we can code the data and aesthetic without explicitly identifying the arguments. Instead, our code could be `ggplot(mpgKtd, aes(sample = hwy))`. That is the style we will take from here on.

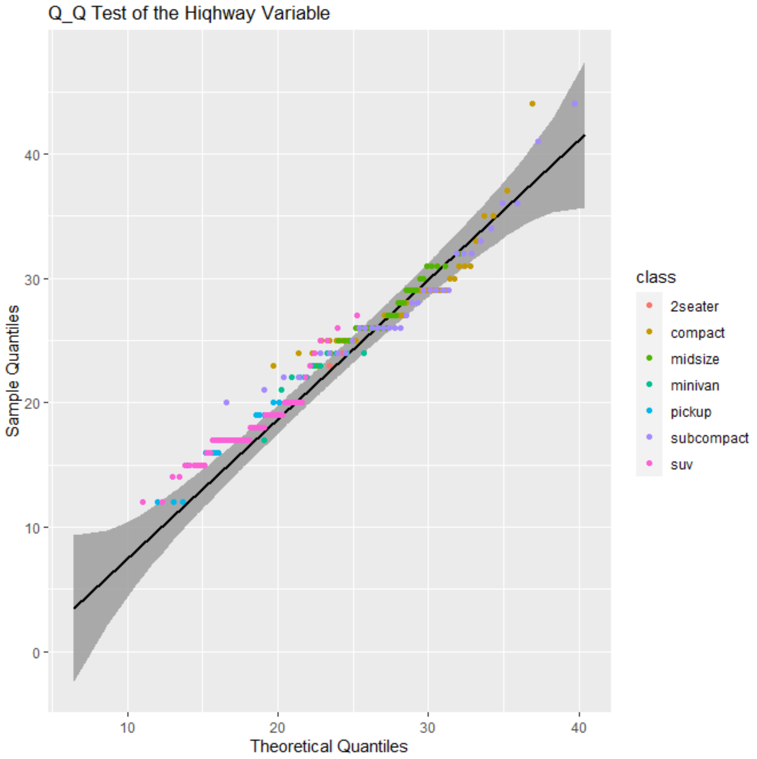
At this point the reader is advised to the read Chapter 2 of the Wickham text. What is explained and demonstrated in the sections to come do not require reading the Wickham chapter but will greatly enhance and enrich one's appreciation of what is being explained in this chapter.

Here is a tip for understanding the code in the pages to follow. Highlight and run each element of code from the left side of the `+` that follows it. Since ggplot2 is layered, you will see instantly what each element causes in the graph. If an argument is of interest, remove and repeat the layered run.

If we did that with the code of Output 5-9, the first output would be a blank plot of the graph with its axes labels x and y. Adding the second would generate the gray standard error zone. The third would add the straight line. The fourth would add the plotted points of the variable and gives them color according to the class variable. The legend is also returned. The remaining two elements would have dealt with the axis and chart titles.

The figure is a visual test if the hwy variable has a normal distribution. If it did, all but a few of the points would reside within the zone of standard error. If we seek a 95 percent confidence, only approximately 5 percent of the 234 points would be outside the zone. This is obviously not the reality.

The principle of the test is that the line and error zone are representative of a variable with the mean and standard deviation of the hwy variable. If normal, at each point along the straight line, a percentile of the data points would have previously occurred. If perfectly normal, the points of the subject variable would fall exactly on the theoretical line.

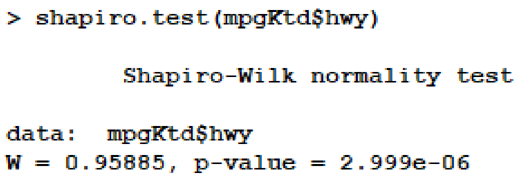


Output 5-9: Q-Q plot of the highway mileage variable.

However, we should confirm the visual test with a statistical test. We can do that with the function `shapiro.test`. Coded as follows, its output is shown in Output 5-10:

```
#Test hwy variable for normal distribution
##Small p-value indicates is not normal.
shapiro.test(mpgKtd$hwy)
```

The outcome of the test verifies the findings of the visualization. The test is based on a null hypotheses. It is that the tested data is not significantly different than the theoretical normal distribution of data with the same mean and standard deviation. The small p-value tells us that the data is significantly different.



```
> shapiro.test(mpgKtd$hwy)

        Shapiro-Wilk normality test

data:  mpgKtd$hwy
W = 0.95885, p-value = 2.999e-06
```

Output 5-10: The findings of the Shapiro analytic disproves a normal distribution.

Back to the code to the graph. Note the code expression `stat_qq_point(aes(color = class))`. More specifically, note the `aes` argument `color = class`. It causes the chart points to be colored according to the class of vehicle and generates the associate legend.

The coloring of points reveals that there are subsets clustered on class. The clusters seem to have their own neighborhoods along the plot. Consequently, testing them in aggregate may be misleading. We should inspect them as subsets.

In Output 5-9, we have subset based on class. We could have otherwise done so for cylinders, drive, transmission and fuel.

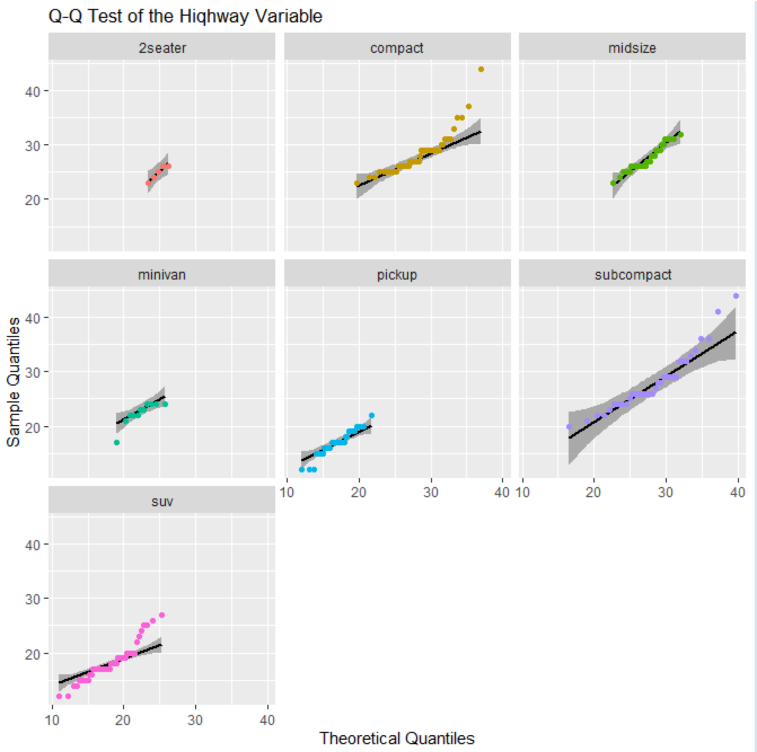
We have other options for bringing out subsets. They are size and group. Furthermore, we can present more than one subset, in a single graph. For example, we could have assigned shape to a second variable with `shape =` and size to another yet with `size =`.

However, it quickly becomes difficult to visually get our arms around the many possible subsets that are permutations of the dataset's categorized variables. Meanwhile, some will be overlapped by others. At some point the chart becomes a dog's breakfast.

()To get over the obstacle, let's look at the `facet_wrap` function as a means to subset the variable and visually test their individual distributions. The code to return Output 5-11 is as follows:

```
#Subset the test by facet
qqhwyFac<- qqhwy +
   facet_wrap(~class) +
   theme(legend.position = "none")
qqhwyFac
```

Rather than build the entire chart from scratch, two additional functions are added to the earlier base graph; `qqhwy`. Notice the geom `facet_wrap`. In it, note the code `~class`. The code specifies class as the variable to be faceted upon. The argument in the theme function removes the legend.



Output 5-11: Distribution of the hwy variable subset upon class.

Upon inspection of Output 5-11, we can wonder if all but compact and suv vehicles would test as normal with the Shapiro test. We could extract each from the mpgKtd dataset and test them for normal distribution. The codes to test each variable for normal distribution are as follows:

```
#Test subsets to hwy for normality on class
shapiro.test(mpgKtd$hwy[mpgKtd$class=="2seater"])
shapiro.test(mpgKtd$hwy[mpgKtd$class=="compact"])
shapiro.test(mpgKtd$hwy[mpgKtd$class=="midsize"])
shapiro.test(mpgKtd$hwy[mpgKtd$class=="minivan"])
shapiro.test(mpgKtd$hwy[mpgKtd$class=="pickup"])
shapiro.test(mpgKtd$hwy[mpgKtd$class=="subcompact"])
shapiro.test(mpgKtd$hwy[mpgKtd$class=="suv"])
```

The code would return the same statistical analyis as shown in Output 5-10 for the hwy variable. The outputs will not be shown here and are left to the reader to execute. However, only the 2seater class shows a normal distribution (p-value = 0.42) and pickup trucks are on the fence (p-value = 0.049). Typically, analysts require that a tested variable must return a p-value of 5 percent or greater to be accepted as normal.

Let’s take the opportunitiy to review subsetting with R. We will subset on a single class. The first step is to subset the table on our chosen classs—midsize—with the following code:

```
#Create a table of midsize records
mpgKtdMid<- mpgKtd[mpgKtd$class=="midsize",]
```

To review, the code to understand are the square-brackets. Within the brackets we are subsetting the mpgKtd table. The expression `mpgKtd$class` identifies the source table and variable. From what is within the square brackets and to the left of the comma, a TRUE/FALSE vector occurs behind the curtains. The TRUEs in the vector will cause only the records in the `mpgKtd` table parallel to the TRUE cases to be returned to the `mpgKtdMid` object.

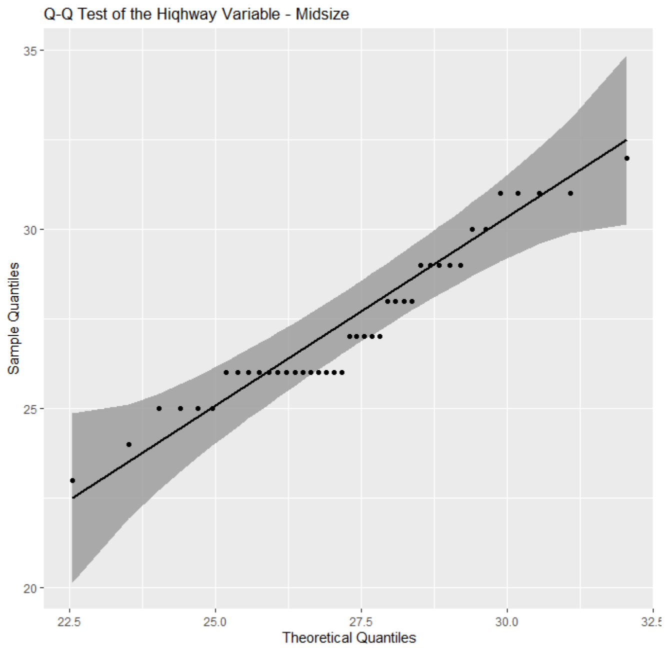
The empty space within the brackets to the right of the comma causes all variables to the `mpgKtd` table to be included in the returned object. Thus, we have a table of only records to midsize cars.

This is a simple subset. We can code any filter in the spaces to either side of the comma within the brackets

Next, let’s look at the Q-Q chart for hwy with respect to the midsize car class. As we would expect, the code is as follows and returns the graph of Output 5-12.

```
#Q-Q Test of midsize class
qqhwyMid<- ggplot(data = mpgKtdMid,
               aes(sample = hwy)) +
        stat_qq_band() +
        stat_qq_line() +
        stat_qq_point() +
        labs(x = "Theoretical Quantiles",
               y = "Sample Quantiles") +
        ggtitle("Q-Q Test of the Hiqhway
               Variable - Midsize") +
        theme(legend.position = "none")
qqhwyMid
```

Once again, the visualization infers a non-normal distribution. Although not shown, we should confirm that with the Shapiro test. However, we obviously need to more fully introduce ourselves to the nuances of our data.



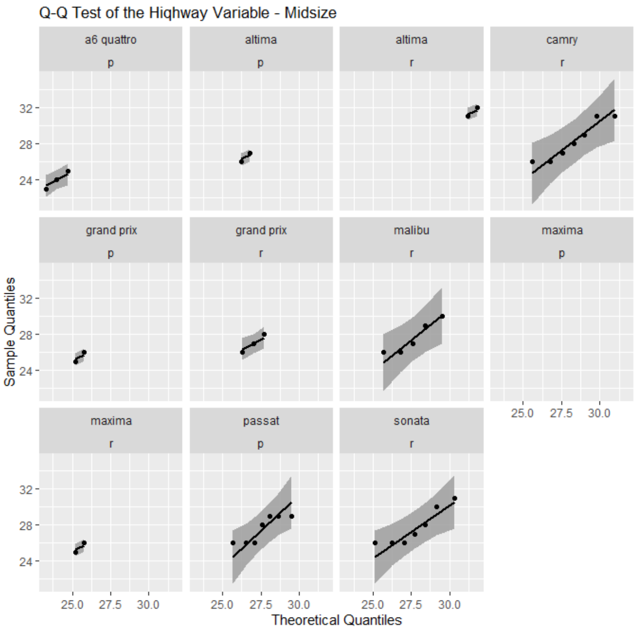
Output 5-12: The midsize class does not show normal distribution.

We earlier saw, in Output 5-11, faceting in practice. The code below to facet on models introduces an additional twist. It is apparent in Output 5-13.

```
#Wrap subset midsize on model and fl
qqhwyMidFac<- qqhwyMid +
   facet_wrap(~model + fl) +
   theme(legend.position = "none")
qqhwyMidFac
```

Notice the geom, `facet_wrap(~model + fl)`. It will create new subsets as can be seen in Output 5-13. Rather than only subset on model, the faceted subsets are permutations of model and fuel type.

As they exclaim in late night low-budget commercials for gadgets, “But wait, there is more!” We are not limited to single a dimension. We can view the Q-Q plots at the intersection of categorical variables.



Output 5-13: Subsetted facets on model and fl to the Q-Q test of the highway variable.

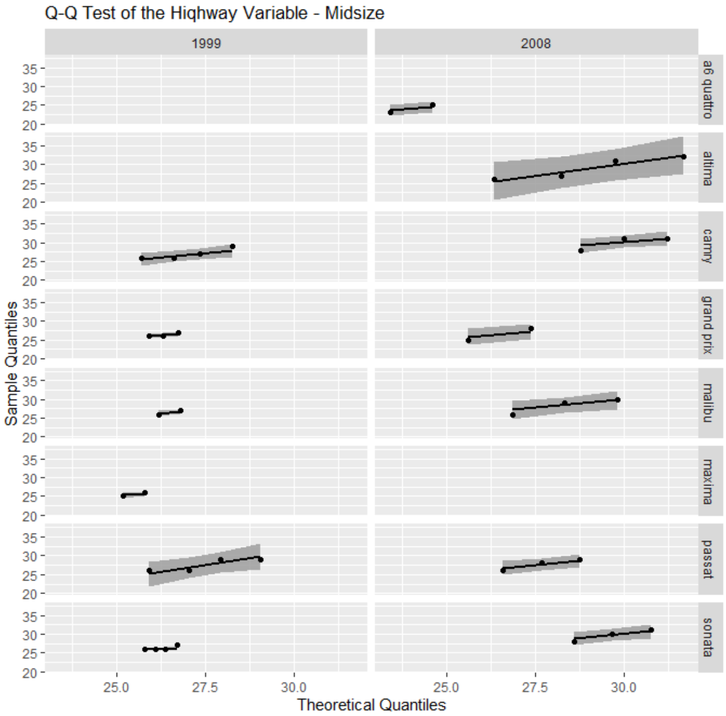
To demonstrate, let's contrast model and year. The following code will return Output 5-14.

```
#Grid subset midsize on model and year
qqhwyMidGrd<- qqhwyMid +
   facet_grid(model~year) +
   stat_qq_point() +
   theme(legend.position = "none")
qqhwyMidGrd
```

Notice that `facet_wrap` is replaced with `facet_grid`. Within `facet_grid`, the code `model~year` returns a grid with year as columns and model as rows with the expression.

There are many rich possibilities for wrap and grid facets. Too many to attempt to explore here. Section 7.2 of Wickham introduces and demonstrates the many variations.

Returning to Output 5-14, let's make an important observation. Notice that the x and y scales are the same for all facets. This allows us to more easily inspect for differences in location and spread among subsets. Of course, ggplot2 offers options to allow one or both scales to float freely.



Output 5-14: Q-Q test by model and year.

In the returned graph we are inspecting the Q-Q plot with respect to model and year. It is notable that there seems to be fit to the normal distribution lines when we introduce year as a facet. As always, we should test what we see with the Shapiro test.

However, as we get to know our data what we have discovered in the facets may cause us to loop back to explore the hwy variable for normal distribution while distinguishing between year. The lack of normal distribution without the distinction may indicate eras of performance characteristics.

The R script to the chapter contains the code for the Q-Q analysis of the city mileage, displacement and cylinder variables. The reader may want to explore the variables after subsetting on year.

Let’s imagine what has been demonstrated with respect to the data of our CMMS and other source systems. Are our costs per order, hours per order, crafts hours per order, count of orders by lead craft normally distributed? What do the distributions look like if we subset them by craft, maintenance types, priority and cost center? How should we subset to get a true sense of what is hidden in our data? Are we embedding misinformation in our standard reports by not subsetting?

5.2.3. Inspect Correlation Between Variables

We should also inspect the correlations between the variables in our datasets. We can do that between numeric variables and between categorical and numeric variables. The explanation of the code and interpretation of correlation was the topic of Section 4.3.4. Partial correlation was the topic of Section 4.3.5.

Rather than recook the beans, the reader is referred to the sections for review. This section will largely demonstrate methods to visually inspect the correlation between the variables of a dataset. In addition to Section 4.3.4, this section will introduce the technique to measure correlation between categorical and numeric variables.

The data of the chapter can be substituted into the code that was explained in Sections 4.3.4 and 4.3.5 The R script to this chapter utilizes the same code but with the substitution of the variables to the mpgKtd dataset.

We can get a visual summary of correlation with the function `pairs.panel` of the psych package. The code below demonstrates the function with 7 of the 11 variables to the mpgKtd dataset. The output is shown in Output 5-15:

```
#Visualize correlation
pairs.panels(mpgKtd[c("hwy", "cty", "cyl", "displ",
   "drv", "model", "trans")])
```

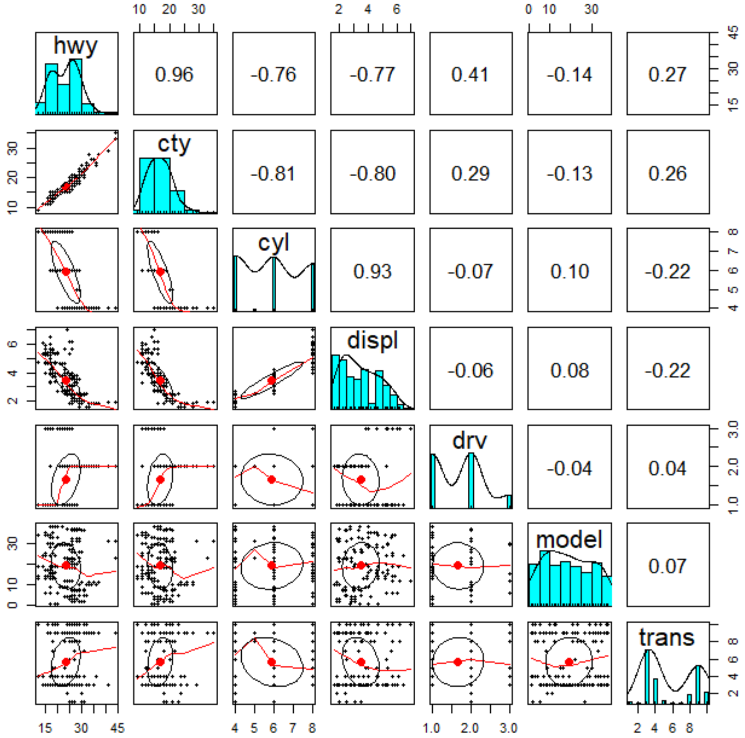
We see in the code that we are subsetting the dataset upon the seven variables it identifies. The `c` function within the square brackets caused their return. If we had written the code as `pairs.panels(mpgKtd)`, we would have returned an inclusive table.

The figure provides a great deal of visual information. However, there is one caution. It is that only the correlation values between numeric variables—hwy, cty, displ and cyl—have credence. Three categorical variables have been included—drv, model and trans—but their correlation coefficients should be ignored.

Other than to demonstrate how to be selective with respect to datasets of many variables, there is no technical reason for showing only seven variables. The reason here is that showing the full set would create a figure of proportions that are impractical to the pages of a book.

Although the correlations for categorical data are not useful, it is still useful to include the variables in the pairs graphic. This is because so much information in the pairs panel is relevant to any type of variable.

We can see 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 oval shape is called the correlation ellipse, the more elongated the greater correlation. The fitted smooth line gives a sense of pattern to the cross plots. The large dot indicates the mean value of plot points.



Output 5-15: Grid of correlations and more to mpg variables.

Although not legitimately presented by the pairs grid, we can calculate a correlation between categorical and numeric variables. However, it is done with respect to levels to the categorical variable.

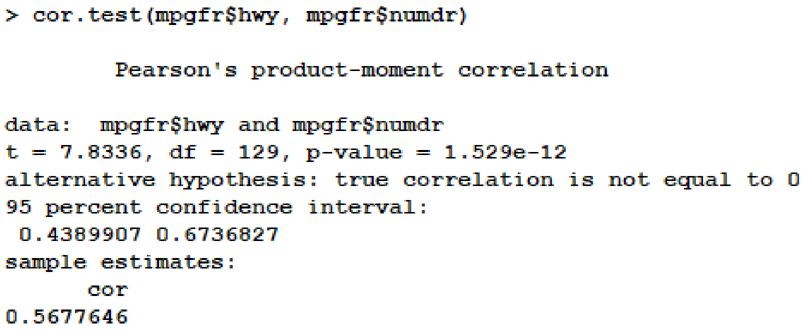
The code below shows how to obtain the correlation of front-wheel drive to highway mileage. The correlation sets rear drive as the base level—zero so to speak—and front-end and highway as the measured correlation. The returned output is presented in Output 5-16.

```
#Correlation of front drv and hwy with rear as base
mpgfr<- mpgKtd[(mpgKtd$drv=="f" | mpgKtd$drv=="r"),]
mpgfr$numdr<- ifelse(mpgfr$drv=="r", 0, 1)
cor.test(mpgfr$hwy, mpgfr$numdr)
```

Let's inspect the code. The code, `mpgKtd[(mpgKtd$drv=="f" | mpgKtd$drv=="r"),]`, subsets the mpgKtd dataset to one with only two categories for drive; front and rear. The "`|`" syntax is an OR relationship.

The code, `ifelse(mpgfr$drv=="r", 0, 1)`, creates a variable in which “0” is assigned to rear drive and “1” to front drive. This is a new numeric variable with which correlation with another numeric variable can be computed.

The third line applies the `cor.test` function to compute the correlation, significance and confidence interval. Output 5-16 returns the analysis which is interpreted as explained in Section 4.3.4. The output shows that, compared to rear-wheel drive, there is a strong correlation of front-wheel drive to highway mileage.



```
> cor.test(mpgfr$hwy, mpgfr$numdr)

        Pearson's product-moment correlation

data:  mpgfr$hwy and mpgfr$numdr
t = 7.8336, df = 129, p-value = 1.529e-12
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.4389907 0.6736827
sample estimates:
      cor
0.5677646
```

Output 5-16: Correlation to front drive to highway mileage.

The visualization of correlation is the most common purpose of a scatter or cross plot chart. Such plots are provided for the pairs in Output 5-15. Of course, we can replicate them as individual charts. However, as shown they are traditional simplistic perspectives.

We will use the capability of ggplot2 to reach much deeper perspectives. The difference is our heightened ability to understand our data when we can recognize subsets within the scatter plot.

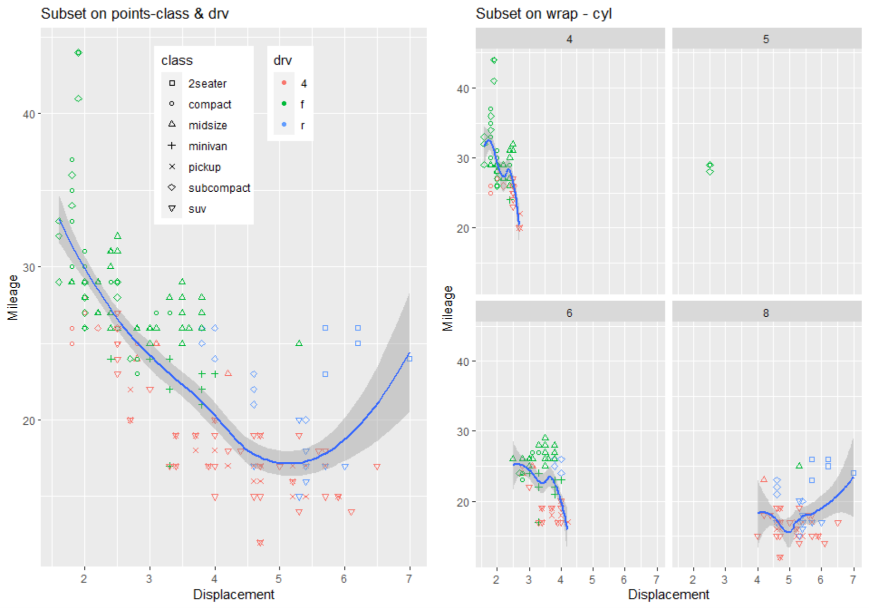
There are two ways to subset cross-plot visualizations with ggplot2. We can show discrete and categorical variables as subsets to the basic plot—e.g., scatter points. Alternatively, we can use facets. Better yet, we can simultaneously apply both methods.

Output 5-17 shows the two possibilities in action. The left-most chart shows a scatter plot being subset upon its points. The right-most chart adds a facet perspective.

The left-most chart begins with the scatter points. Thence, it subsets the points with respect to two categorical variables—class and drive. Additionally, a smooth line and its error zone have been layered over the points.

When we inspect the smooth plot, we can see that the correlation between mileage and displacement changes direction. Here we are discovering something to be investigated because we suspect that we should not see such a pattern.

We should also note that, given the universal love of linear fit, had we chosen to place a linear plot over the points, we might have not noticed the strange pattern. Just as bad, the 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.


Subset on points-class & drv

Subset on wrap - cyl

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

##Alernate 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 `rlm` 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.

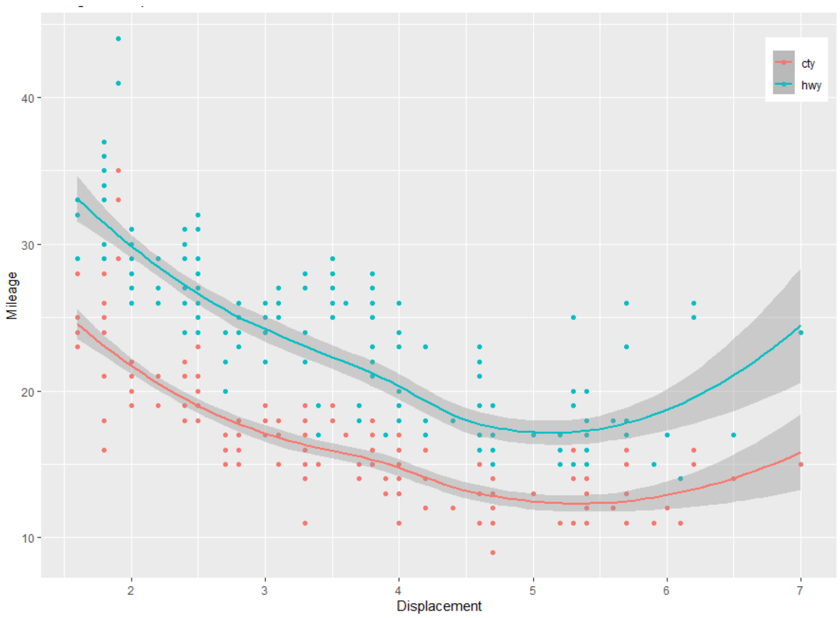
Notice that we have assigned the output to an object; scat1x2. We are using the ggarrange function of the ggpubr package. The charts are designated by their assigned name as objects. Thence, we have specified a single row and two columns.

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 ggarange 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.

The code also allows the combined plots to return a legend. The argument, `color =` in each of the geoms makes the legend happen for the case of multiple base plots.

Know that the two geoms for the displ-hwy visual are taking their specified variables from the `ggplot` function. Thus, they need not include x-y variables in their `aes` functions.

However, the geoms for the displ-cty visual must include them. This causes the geoms to override the base plot variables and instead generate on the displ and cty variables.

Also notice that we have coded to omit a title to the legend. It is left to the reader to search the internet for the code to name the legend other than the default title—color.

There is a final point to make. The base graph supported two graphs. We mentioned different x-y variables to create them—displ-hwy and displ-cty.

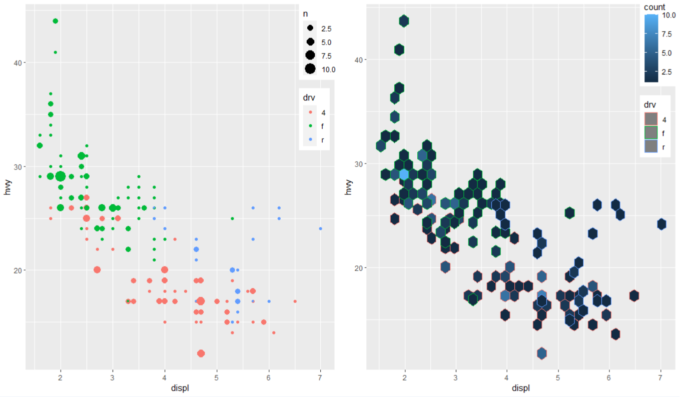
We should also note that a graph need not be limited to a single dataset. In this case, the highway and city mileage were recorded in the same dataset. But what if they were not.

For the respective geoms, we would have simply identified the associated datasets. The only requirement is that the respective charts have the same axes scales.

Finally let's deal with a natural problem to point charts; over plotting. In the graphs to this point, the reality is that not all of the 234 observations of the full mpgKtd dataset will be visible to us. Some will overlap or hide others.

One method to overcome the problem is the `jitter` geom. It will be presented in the next section. It works for continuous or discrete data grouped in a graph by categorical or discrete levels. However, a jittered perspective can be misinformation for continuous variables. This is because each point is moved slightly on the chart.

There are other ways to deal with over plotting. Output 5-19 shows two. They use the geoms `count` and `hex`. Others, not shown, are three-dimensional such as `geom_contour` and `geom_raster`.



Output 5-19: Area count and hex methods to deal with over plotting points.

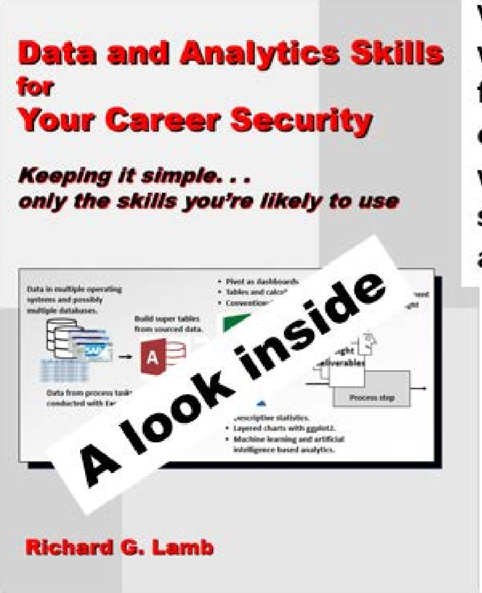
In both cases, the charts are expressing the number of points falling in a plotted area. The legends are given as a reference of comparative magnitude. The code to the respective charts are as follows:

```
#Left panel
#Methods for Over lapping and plotting
##Count in area
scatAreaCnt<- ggplot(mpgKtd, aes(displ, hwy,
        color = drv)) +
   geom_point() +
   geom_count() +
   theme(legend.position = c(.95, .8))
scatAreaCnt

#Right panel
#Hex chart
scatHex<- ggplot(mpgKtd, aes(displ, hwy,
        color = drv)) +
   geom_hex() +
   theme(legend.position = c(.95, .8))
scatHex

#Side-by-side
#Plot count and hex charts
overPlot1x2<- ggarrange(scatAreaCnt, scatHex,
   ncol = 2, nrow = 1)
overPlot1x2
```

The first block of code adds the `count` geom to the scatter plot. The second block replaces the `point` geom with the hex geom. The third block is the boiler plate code to return the graphics of both methods as a side-by-side chart.



Data and Analytics Skills for Your Career Security

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

A look inside

Richard G. Lamb

Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Data from process task conducted with
Pivot as dashboards
Tables and calc
Convention
ght
deliverables
Process step
descriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

Cleansing data entails a series of assessments. Can we hang our hats on it? If not, which variables of our dataset 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? Some bad data are apparent while others can only be unearthed with statistic measures, machine learning and artificial intelligence.

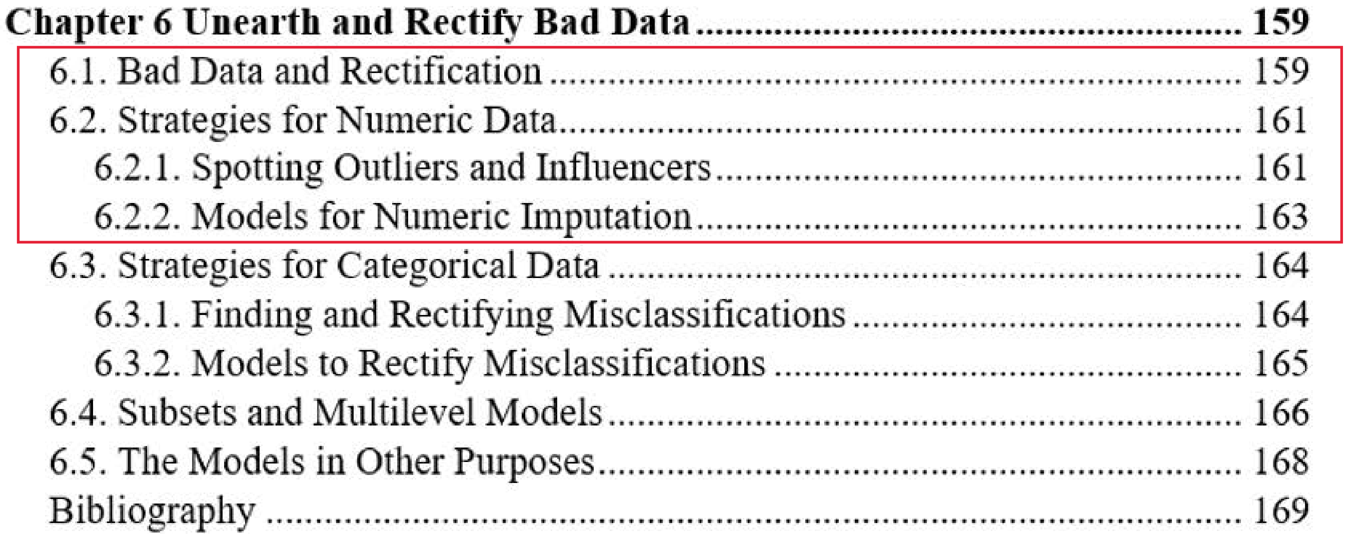
Chapter 6: Unearth and Rectify Bad Data

Excerpts:

6.1. Bad Data and Rectification

6.2. Strategies for Numeric Data

Additional “Look Inside” at https://analytics4strategy.com/book-look-inside
Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds



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.

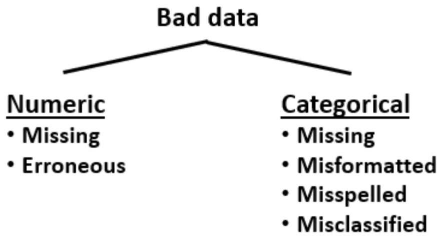


Figure 6-1: Types of bad data by type of variable.

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 will be. ML-AE models as the strategy for rectification are the subject of the next section. The search is the subject of this section.

Chapter 3 to build super tables provided one of a hierarchy of methods to unearth outliers. It was to compute Z-score (section 3.4.4) and student T-score (section 3.4.5) to numeric variables. The tests work best when we subset the numeric variable with respect to significantly related categorical variables and "binned" numeric variables.

Along with the calculated Z-score and T-score, we can utilize graphic methods to identify outliers in the context of all data points. Section 5.2.4 and 5.2.6 demonstrated the graphic methods to inspect the centrality and spread of numeric variables. With the graphic methods, we can subset the numeric variable with respect to categorical variables. This increases our chance of spotting outliers and influencers. Otherwise, the outliers to a subset can hide in an aggregation.

A third and more powerful method moves into the realm of machine learning but with a twist. The twist is that we use the learned fit of a linear regression model to unearth any outlier cases the model would not have accurately predicted and cases that would have changed the model through its parameters.

Here we will introduce the method. Chapter 7 will explain and demonstrate the method in the context of explaining and demonstrating linear regression.

We do not typically realize that the mean and variance to a numeric variable is a model. It is called the null model because it has no predicter variables.

The objective of any model is to learn a fit of one or more predictor variables that maximally explain the variance to the null model. As we build a model by variously including predictors, we evaluate each variation with respect to its increased ability to explain the variance of the null model.

Once the best explaining model is arrived at, we can measure all observations for whether they are incongruent to the learned outcome. We seek them out as the two situations shown in Figure 6-2. They are called leveraging and influencing cases.

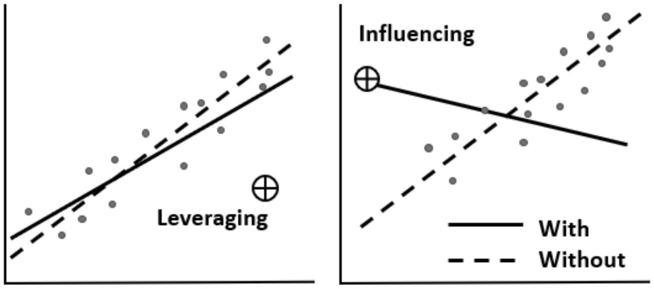


Figure 6-2: Leveraging and influencing cases change a models intercept and slope.

We can see the principle in the depictions of a one-predictor linear regression model. As demonstrated in Chapter 6, we could use a layered graphic of a liner regression over a cross plot to spot the cases. However, once we move to two or more predictor variables, we must utilize statistical analytics to reveal outliers and influencers.

Let's begin with the left-most graphic depiction of a leveraging case. The regression is the solid line of intercept and slope. An outlier is shown marked as a large circled point. There could be multiple outliers, but we will make the point with a single outlier.

The point is that the model with the leveraging case would experience some degree of significant shift and rotation of model's intercept and slope. Notice that the leveraged case is within the horizontal range of points.

In contrast is the right-most graphic of influencing case. We see that one case is high jacking the model while the fit is largely reflective of ignoring the obvious pattern of most observations. In contrast to leveraging cases, notice that the case is well outside the horizontal range of points and well away from the linear fit.

The procedure to build a linear regression model includes tests to evaluate if a model's accuracy is affected by outliers and influencers. It follows that we can build a model with variables of our dataset as the dependent variable and test for the cases. The methods to test for them are explained by demonstration in Chapter 7.

6.2.2. Models for Numeric Imputation

It was previously mentioned that we do not automatically decide to rectify bad data. Not all bad data may have the same ramification to any one insight. If our decision is to rectify rather than eliminate or ignore, then our strategy is to impute our best estimate to be a corrected entry.

This calls for machine learning and artificial intelligence (ML-AI) for which there are considerable models to choose from. They range from straight forward to esoteric, directed to undirected, and transparent to black box.

Of the models, linear regression provides the most information generally as well as specifically for cleansing. Linear regression will be fully explained by demonstration in Chapter 7.

Accordingly, the book will dedicate its attention to linear regression because of its deserved ubiquity. However, the reader is recommended to explore model trees and regression trees as a practical methodology. They are explained and demonstrated in the book, Machine Learning with R, by Brett Lantz.

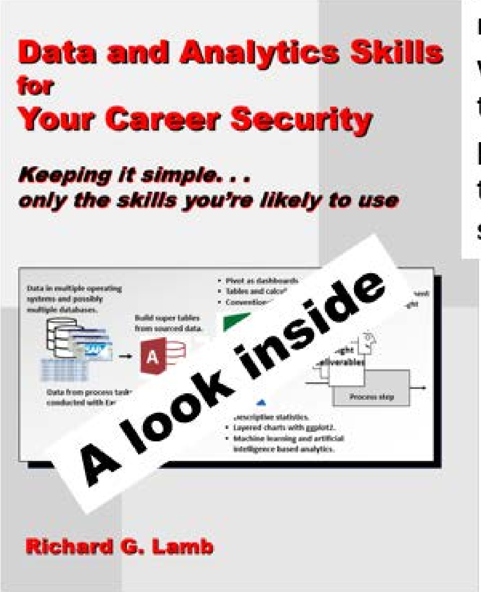
For imputation, the ML stage is the action of fitting a regression model to the data tables from which we have removed the records containing the previously identified bad data. In turn, the AI stage is the action of feeding the predictor variables of our bad records to the learned model. The returned estimates are imputed to replace the bad data.

With linear regression we determine, by cycles, which predictor variables are best and statistically significant to estimating the bad numeric variable. It falls to us as subject 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.

The fourth step is to return to the first step. However, we append the newly validated or corrected data to the growing block of records with good data. We would repeat the loop until we have the block of history we need for the insight deliverable we are planning to build.

Because our records must necessarily be practicably 100 percent correct rather than substantially correct, the method is potentially a substantial task albeit not nearly as prohibitive as the manual approach. For this reason, the action to recover the past should be parallel to actions to improve the process that would otherwise allow a future of continuing to collect bad classifications and having to labor through the rectifying process as a step to each reporting period.



Data and Analytics Skills for Your Career Security

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

A look inside

Richard G. Lamb

Rather than linear regression models to predict outcomes, usually the biggest operational ramifications are from learning which operational variables are meaningful to prediction? Do any of the variables need to be transformed to sharpen predictability? Are there subsets within variables that must be recognized? Does the model tell the story universally or only for our circumstances?

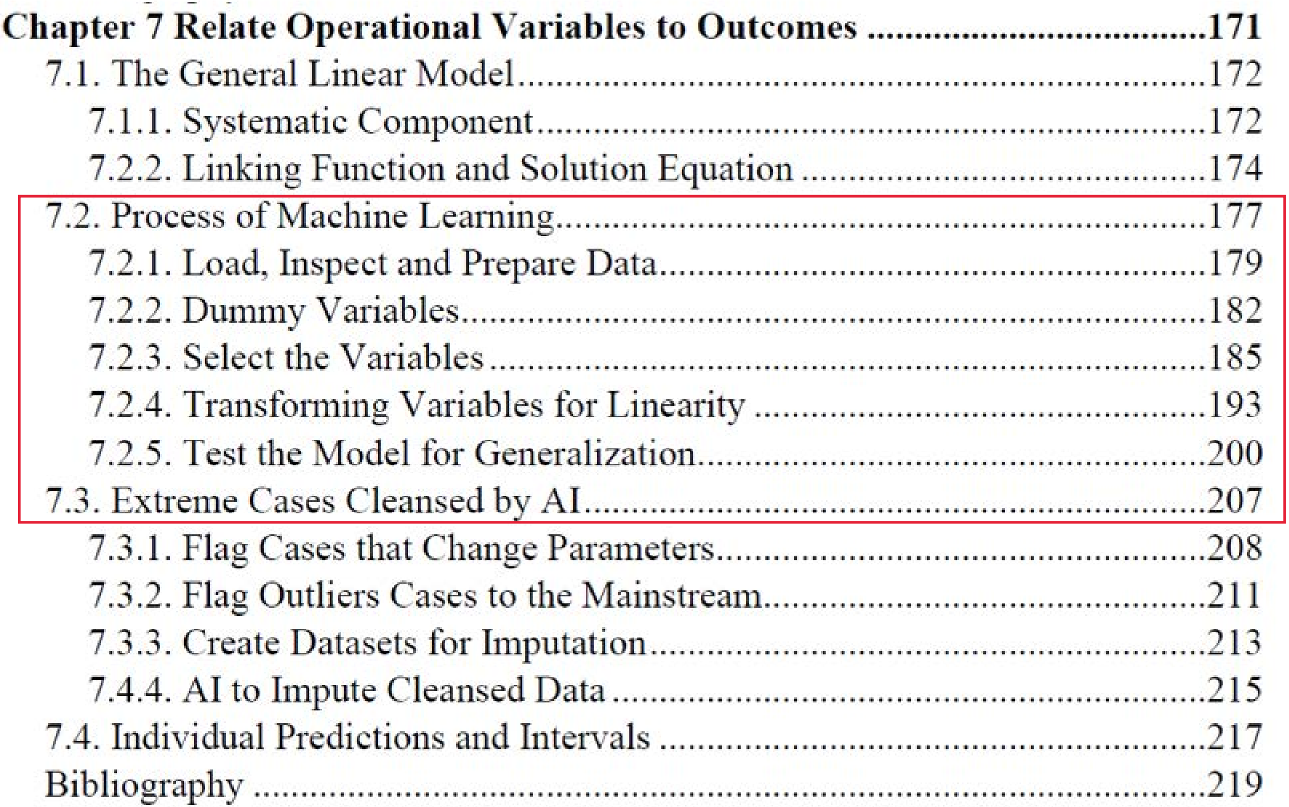
Chapter 7: Relate Operational Variables to Outcome

Excerpts:

7.2. Process of Machine Learning

7.3. Extreme Cases Cleansed by AI (Introduction)

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside
Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds



Note:

The advance graphics created with the ggplot2 package of the R software depend heavily on color to communicate the insight they are coded to give us. Some such graphics appear in the excerpts.

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.

For convenience, the herein excerpts will present the color version.

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.

7.2. 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.

Along with expenses, all other variables are included in the dataset as potential predictors of medical expenses. Just as for any enterprise operation, they are both numeric and categorical data. The numeric variables are age, BMI and number of children. The categorical variables are sex, region and smoker.

We can easily imagine expenses such as cost per work order and predictors as a host of numeric and categorical variables such as metrics, maintenance type, priority, etc. Alternatively, we can imagine a domain expert using the data to plan, organize and control some sub operation to an insurance enterprise.

The sections to come only require that the readers have the knowledge and explained by demonstration in Chapters 3, 4 and 5. For those who wish to reach more deeply into the theory and mathematics, two texts are recommended in order of difficulty.

First, Chapter 9 of the text, "Discovering Statistics with R," by Field and Miles provides a non-advanced explanation of linear regression. Furthermore, the text explains and demonstrates linear regression with the R software.

Second, are Chapters 11 through 13 of the book, "Introduction to Statistical Methods and Data Analysis," by Ott and Longnecker. The chapters provide an advanced and graduate level explanation of linear regression. The chapters will reference commercial software rather than R. However, the reader will know the equivalents in R by virtue of this chapter and the Field and Miles text.

The R functions to play will require that the reader pull the packages into the session per the code below with the `library` function. That assumes the packages have been installed with the `install.packages` function as demonstrated in Chapter 3.

```
library(xlsx); library(psych); library(ggplot2)
library(qqplotr); library(ggpubr); library(mice)
library(MASS); library(QuantPsyc); library(nlme)
library(car); library(dplyr)
```

7.2.1. Load, Inspect and Prepare Data

The first task is to download the dataset. From previous chapters, the code below is familiar to us. The data is assigned to the data frame object named `insure.1`. The reader will have to insert as `<path>` the location at which they have stored the dataset on their computer or server.

```
insure.1<- read.xlsx(
   "C:\\<path>\\SkillsForCareerSecurity_Datasets.xlsx",
   sheetName="insurance", header=TRUE)
```

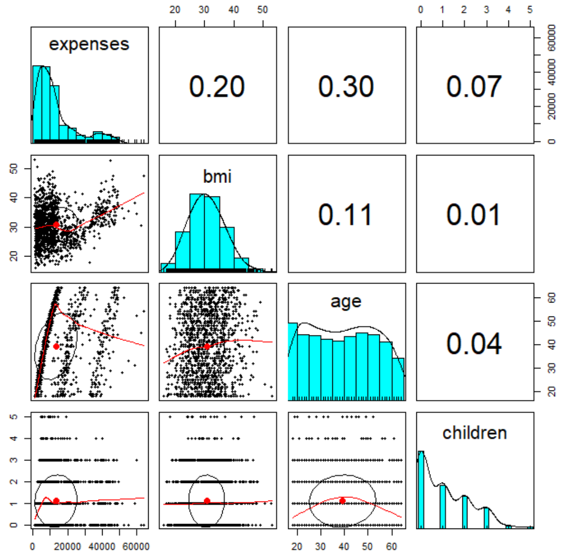
Next, we should inspect our data with the functions below. The functions were explained and demonstrated in Chapter 5. Better yet, we should apply the chapter's full suite of practices to know our data.

```
#Inspect dataset
head(insure.1)
str(insure.1)
summary(insure.1)
describe(insure.1)
#Check for missing data
md.pattern(insure.1)
```

We should also inspect the correlations between the numeric predictor variables of the dataset. We can do that with the code below. It will return the visuals of Output 7-1.

```
pairs.panels(insure.1[c("expenses",  "bmi", "age", "children")])
```

The most important information in the returned panel is the correlation between predictor variables. Lower is better than higher. This is because high collinearity undermines the ability to determine which of two highly correlated variables are most strongly related to a model's outcome. Such variables will split the effect. Correlation above 90 percent is a red flag. Output 7-1 shows that collinearity is not an issue to the model. This will be confirmed by statistical test in section 7.2.5.



Output 7-1. Pairs grid to inspect correlations and histograms.

We can see the correlation between each predictor variable and the outcome variable, `expenses`. However, this does not accurately depict the true strength of relationship. As explained in section 4.3.5, paired variables share correlation with other variables. A paired correlation will be weaker when we consider their shared correlation with a third or more other variables to the dataset.

It is also helpful that the output returns the histograms of each variable. The expenses variable is highly skewed to the right and is obviously not normally distributed. The BMI variable appears to be normally distributed. The age variable suggests multiple shapes combined as one. Finally, the children variable is a discrete variable with a non-normal distribution.

Next, for our example, we want to add three variables to the dataset that will be created upon several of the original variables. However, we will cheat for the sake of explanation by adding variables for needs we would not yet know that we need. We would, instead, recognize them later in the progression to a final model.

The variables `age2`, `bmiCl` and `bmiObese` are added by the code below.

```
#Assign insure.1 to insure.2
insure.2<- insure.1

#Add an age quadratic variable
insure.2$age2<- insure.2$age^2

#Add BMI categories
insure.2$bmiCl<- as.factor(ifelse(insure.2$bmi >= 30, "Obese",
   ifelse(insure.2$bmi <= 24.9, "Healthy", "OverWt")))

#Add bmi greater or less than obese
insure.2$bmiObese<- as.factor(ifelse(insure.2$bmi >= 30, "Over", "Under"))

#Check insure.2 for desired outcomes
str(insure.2)
```

The first block of code assigns the `assure.1` dataset to `insure.2`. This is not necessary but is my preference. This allows us to do things with the data while leaving `insure.1` in tack. If I were to distort or damage the data, I only need to return to this point rather than download the source dataset and rerun everything between the initial download to my present location in the code. Furthermore, I can always be comfortable that at all things done before the assignment still run true.

The second block of code creates and places a variable that is age squared. The element `insure.2$age` identifies the variable to be squared and the "`^2`" element is the calculation. The code `insure.2$Age2` receives the calculated variable in our dataset as `Age2`.

The third block of code creates the variable with levels by which BMI scores are classified. A BMI greater than or equal to 30 is classified as "Obese." Accordingly, "Obese" is assigned as true to the `ifelse` function and an ifelse function to not true. The second `ifelse` assigns 24.9 or less as true to be classified as healthy and all that remains as false is classified as overweight.

The fourth block of code creates the two-level variable `bmiObese`. The true condition to the `ifelse` function is a BMI that is over or equal 30 and is named `Over`. The false condition is named `Under`.

Notice the `as.factor` function in the third and fourth blocks. Without it, the variables may be recognized by R as character variables rather than a factor variables. As will be seen later, we will use the designation to set up what are called dummy variables.

Finally, it is always a good idea to inspect the dataset rather than assume our code gave us what we want. That is the purpose of the final block of code.

My preference for a quick check is the `str` function because it confirms that the variables, their type and total count of cases are what I expected. Additionally, the `head` function allows a view of the dataset as a table.

7.2.2. Dummy Variables

The loaded dataset includes three categorical variables: `sex`, `smoker` and `region`. We created two additional categorical variables to the dataset; `bmiCl` and `bmiObese`.

Although most are text-like, categorical variables are made to be numeric to modeling and analytics. The secret handshake is "dummy" variables. We will use the created variables. `bmiCl` and `bmiObese`, to explain and demonstrate them.

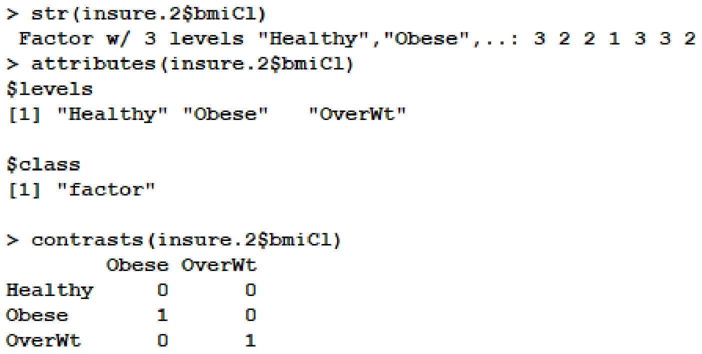
In the code below, notice that we are looking at a single variable, `bmiCl`, rather than the entire dataset.

```
str(insure.2$bmiCl)
attributes(insure.2$bmiCl)
contrasts(insure.2$bmiCl)
```

In Output 7-2, the `str` function reveals that the variable has three levels but the function only enumerates two. The `attributes` function enumerates all levels.

The `contrasts` function returns the dummy variables to the `bmiCl` variable. They are the columns. Notice that there are three levels but only two dummy variables; `Obese` and

`OverWt`. The important point is that they are treated as numeric in models and analytics by virtue of being either “0” or “1.”



```
> str(insure.2$bmiCl)
 Factor w/ 3 levels "Healthy","Obese",..: 3 2 2 1 3 3 2
> attributes(insure.2$bmiCl)
$levels
[1] "Healthy" "Obese"   "OverWt"

$class
[1] "factor"

> contrasts(insure.2$bmiCl)
        Obese OverWt
Healthy     0      0
Obese       1      0
OverWt      0      1
```

Output 7-2. Levels and dummy variables to the bmiCl variable.

The reason for two dummies, one short of all levels to the variable, is that each level is in contrast to the third as a reference. The ramification will be explained later in the context of interpreting a returned regression model.

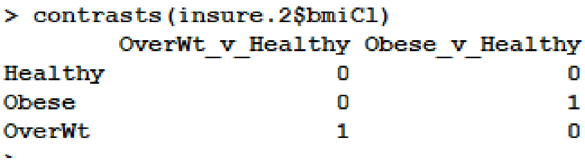
Let’s look closer at the output of the `contrast` function in Output 7-2. As already noted, the dummy variables to the variable present as columns. For the dummy, `Obese`, the value of “1” is assigned for all Obese cases in the dataset and “0” for all others. Similarly, for the dummy `OverWt`, the value “1” is assigned for all `OverWt` cases and all else as “0.” Finally, notice across both dummies, the value of “0” is common to all `Healthy` cases. This creates the reference effect.

Setting up the dummy variables is automatic in R. Shown in Output 7-2 is what happens automatically. The reference is set alphabetically in ascending order. In turn, each dummy will remain named according to its original classification in the dataset; `Obese` and `Healthy`.

We can control which level is the reference variable and name the dummy variables. This is done with the code below for which the returned output is shown in Output 7-3.

```
OverWt_v_Healthy<- c(0,0,1)
Obese_v_Healthy<- c(0,1,0)
contrasts(insure.2$bmiCl)<- cbind(OverWt_v_Healthy, Obese_v_Healthy)
contrasts(insure.2$bmiCl)
```

The first two lines of code creates individual vectors that define and name the two dummy variables from the three levels. By omission as a vector, the code concurrently causes Healthy to be the reference level. Notice, that for clarity, we have given the dummies a name that defines them relative to the reference level. We next bind them together as an object with the cbind function. Finally, we assign them to the bmicl variable with the contrasts function.



```
> contrasts(insure.2$bmiCl)
        OverWt_v_Healthy Obese_v_Healthy
Healthy                0               0
Obese                  0               1
OverWt                 1               0
```

Output 7-3: The dummy variables as named and setting the reference level.

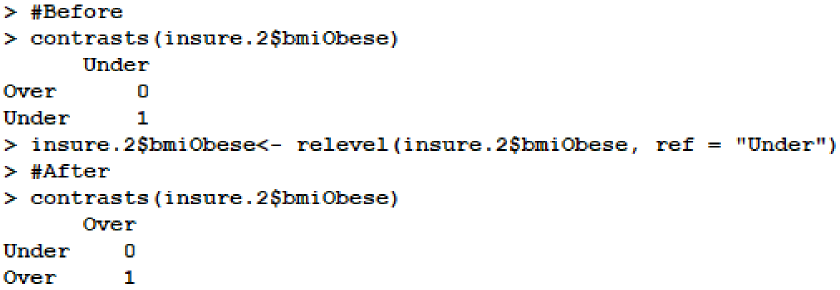
When we call the contrasts function, as shown in Output 7-3, we see that our named dummies are now attributes to the bmicl variable. The two dummies have taken on the name we gave them. Our named columns, rather than dataset names for the levels, will appear in our model.

If we do not choose to rename the dummy variables to our model, we can still control which level is to be the reference. The code below returns a before and after example. The consequences are shown in Output 7-4. The bmiObese variable will be the example.

```
#Before
contrasts(insure.2$bmiObese)
insure.2$bmiObese<- relevel(insure.2$bmiObese,
  ref = "Under")
#After
contrasts(insure.2$bmiObese)
```

Notice in the upper part of Output 7-4 that it is automatic for "Over" be the reference due to its alphabetic order. However, we want to relate our model to "Under" as the baseline against which to compare to "Over."

The relevel function allows us to make the change. The ref = argument makes the change and the relevel function assigns the choice to the bmiObese variable.



```
> #Before
> contrasts(insure.2$bmiObese)
      Under
Over      0
Under     1
> insure.2$bmiObese<- relevel(insure.2$bmiObese, ref = "Under")
> #After
> contrasts(insure.2$bmiObese)
      Over
Under    0
Over     1
```

Output 7-4: Before and after to resetting the reference level from automatic.

Actually, what is explained for the two variables as examples is the case for all categorical variables. Behind the curtains, all are a set of dummy variables. And for all, we have the choice to shape them or leave them as they are. It is left to the readers to inspect the various categorical variables with the `contrasts` function.

7.2.3. Select the Variables

The next step is to select variables to the model. We are unlimited in the variables we can include. Placed in a dataset, they can be extracted from management systems and be metrics computed upon operational performance. Variables can also be created along the way.

For the demonstration we have the numeric variables of age, BMI and number of children. We have categorical data of smoker, sex and region. We have created two additional categorical variables of bmiCl and bmiObese.

If we were to build models to explore what matters most to various outcomes to our maintenance operation, we would have the same types of variables. We would have pulled them together in a super table as demonstrated in Chapter 3. Most would come from our CMMS and others from the specializes systems that capture them such as payroll, human resources and scheduling. And from both we would tease others with the sought insight in mind.

Our numeric variables may be dollars of labor and material, craft hours, productivity, etc. Our categorical variables may include cost center, maintenance type, priority, lead craft, etc. We may tease out numeric variables such as percent scheduled, percent schedule compliance day of order, percent proactive and reactive work, actual as percent of planned cost, etc. We may also tease out categorical variables such as if planned, if scheduled, if completed as scheduled, if carry over, etc.

However, a regression built upon a data dump is not particularly meaningful. We would not model on every variable in our super table just because we can. Instead, we draw upon our domain expertise to select the best parsimonious set for the insight we seek. Furthermore, we would follow the procedure that will now be explained by demonstration.

A first step for considering inclusion is to measure the standardized effect of each numeric variable to a numeric-only model. The measure is the change in standard deviation of the output variable caused by a standard deviation change of each predictor variable.

The code below creates a liner regression with only our numeric variables. Then we use the `lm.beta` function to make the comparisons.

```
#Model for effects
model.stdEffect<- lm(expenses ~ age + bmi + children, data = insure.2)
```

```
summary(model.stdEffect)
#View comparative effect of 1 std deviation
lm.beta(model.stdEffect)
```

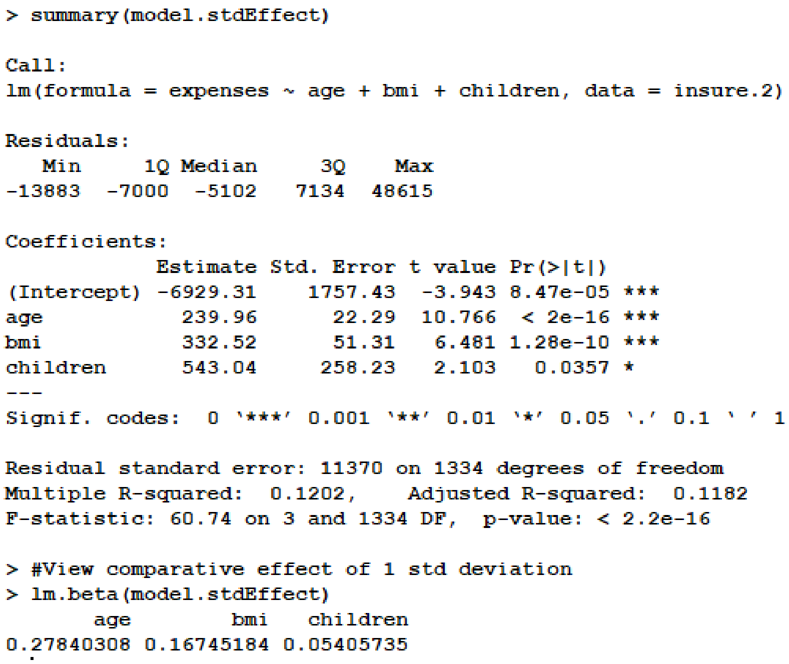
In the code, `expenses ~ age + bmi + children`, we see the form that all models take. The variable expenses followed by "~" and then a body (systematic component) of predictor variables. Section 7.1.1 showed and explained permutations of bodies, but they all set up from the foundational pattern.

Output 7-5 shows what is returned by the code. The `summary` function calls up the model. The `lm.beta` function returns the measure of standardized effect for each variable.

We will look deeply at the interpretation of the regression in later sections. However, note the p-values in the coefficients table as the column `Pr(>|t|)`. The very small values for `age` and `bmi` tell us that they are strongly related to the prediction of expenses. The variable, `children,` has effect but comparatively much less.

However, we want to better understand the relative effect in a form we can make sense of in our minds. The `lm.beta` function gives us that sense.

We can see in lower Output 7-5 that a standard deviation of change in `age` causes a 27 percent of standard deviation change in expenses. In contrast, `bmi` and `children` have standardized effect of 16 and 5 percent respectively, Obviously, we want to begin our selection of variables with age.



```
> summary(model.stdEffect)

Call:
lm(formula = expenses ~ age + bmi + children, data = insure.2)

Residuals:
   Min     1Q Median     3Q    Max 
-13883  -7000  -5102   7134  48615 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -6929.31    1757.43  -3.943 8.47e-05 ***
age           239.96      22.29  10.766  < 2e-16 ***
bmi           332.52      51.31   6.481 1.28e-10 ***
children      543.04     258.23   2.103   0.0357 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11370 on 1334 degrees of freedom
Multiple R-squared:  0.1202,	Adjusted R-squared:  0.1182 
F-statistic: 60.74 on 3 and 1334 DF,  p-value: < 2.2e-16

> #View comparative effect of 1 std deviation
> lm.beta(model.stdEffect)
       age        bmi   children 
0.27840308 0.16745184 0.05405735 
```

Output 7-5: Model for effects and analysis of standardized effects.

The model's low R-squared (0.12) suggests that one or more of the categorical variables should be evaluated for achieving a much greater effect. If they do not, we are not able to build a reasonable model with our given and created variables. This is because the numeric model explains only 12 percent of the variance to the null model; mean of expenses.

Model.1: Age and categorical variables. We will start with a model that includes `age` and the four categorical variables: `bmicl`, `sex`, `smoker` and `region`. The code to create the model and return its output and the confidence intervals to the coefficients is given below. The returned output is shown in Output 7-6.

```
#Age with all categorical variables
model.1<- lm(expenses ~ age + bmicl + sex +
   smoker + region, data = insure.2)
summary(model.1)
confint(model.1)
```

Output 7-6 shows the model that is called up with the `summary` function and the confidence intervals, to the coefficients. Let's interpret the model beginning with the line titled F-statistic. The null hypothesis is that the model does no better than the mean of expenses. The F-statistic is the test of the hypothesis.

The point of interest is the p-value of 2.2e-16. If equal or less than our acceptance criterion, say 5 percent, we reject the null hypothesis and accept that the model is better than the null model.

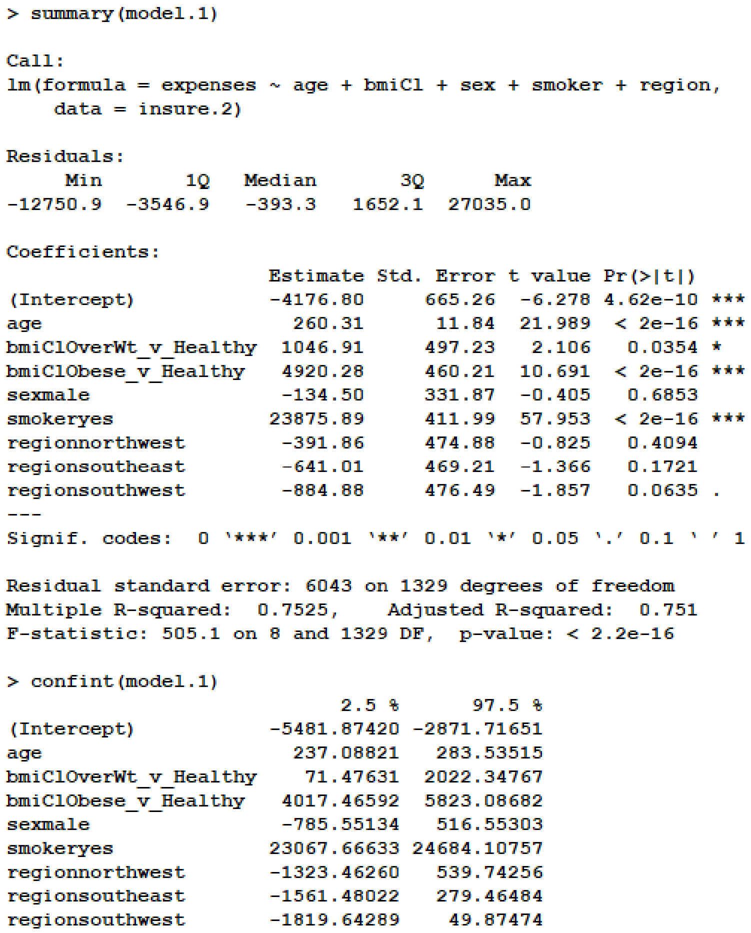
That takes us to the next question. How much of the variance around the mean does the model explain? The R-squared statistic tells us that the model explains 75.25 percent of the variance. Not bad.

Recall that the numeric-only model (Output 7-5) explained just 12 percent of the variance. Therefore, we have improved the model hugely by adding the categorical variables `bmicl`, `sex`, `smoker` and `region`.

The residual standard error to the predicted output is $6,043. It is the standard deviation of the residuals of each observations to its fitted prediction. Checking back to the numeric-only model, the residual standard error was $11,370.

Even though `model.1` is a much better model, we must seek to improve it. We begin by inspecting the p-values for the regression coefficient to each predictor variable. Once again, the smaller the p-value, the stronger a variables plays in the modeled outcome.

It is typical to remove any variable with a p-value greater than 5 percent. However, we may want to set our own threshold for work such as six sigma.



```
> summary(model.1)

Call:
lm(formula = expenses ~ age + bmiCl + sex + smoker + region,
    data = insure.2)

Residuals:
     Min       1Q   Median       3Q      Max
-12750.9  -3546.9   -393.3   1652.1  27035.0

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           -4176.80     665.26  -6.278 4.62e-10 ***
age                     260.31      11.84  21.989  < 2e-16 ***
bmiClOverWt_v_Healthy  1046.91     497.23   2.106   0.0354 *
bmiClObese_v_Healthy   4920.28     460.21  10.691  < 2e-16 ***
sexmale                -134.50     331.87  -0.405   0.6853
smokeryes             23875.89     411.99  57.953  < 2e-16 ***
regionnorthwest        -391.86     474.88  -0.825   0.4094
regionsoutheast        -641.01     469.21  -1.366   0.1721
regionsouthwest        -884.88     476.49  -1.857   0.0635 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6043 on 1329 degrees of freedom
Multiple R-squared:  0.7525,    Adjusted R-squared:  0.751
F-statistic: 505.1 on 8 and 1329 DF,  p-value: < 2.2e-16

> confint(model.1)
                            2.5 %      97.5 %
(Intercept)           -5481.87420 -2871.71651
age                     237.08821   283.53515
bmiClOverWt_v_Healthy    71.47631  2022.34767
bmiClObese_v_Healthy   4017.46592  5823.08682
sexmale                -785.55134   516.55303
smokeryes             23067.66633 24684.10757
regionnorthwest       -1323.46260   539.74256
regionsoutheast       -1561.48022   279.46484
regionsouthwest       -1819.64289    49.87474
```

Output 7-6: Model with age and categorical variables and confidence intervals.

The `age` variable is important to our model. The p-value is very small; well less than 0.001. With each 1-year increase in age, annual expenses increase by $260.

Now we will see for the first time how to interpret categorical variables. We see the two of three levels to the `bmicl` variable. Let's look at the `obese_vs_Healthy` variable in the output.

Recall that `Healthy` is our reference level. However, it will not be directly apparent amongst variables. In prediction, healthy is only in play for cases when both associated dummies are "0" because the case individual is healthy. In other words, we want to predict the expenses for a healthy case rather than an overweight or obese case. It is non-sensical that either dummy would be "1" because the three levels to the `bmicl` variable are mutually exclusive within any case.

Therefore, as compared to a healthy person, one who is obese will experience a $4,920 greater medical expense each year. Once again, the p-value is much less than 0.05, thus, making the `bmiCl` variable a keeper.

In contrast, the strength of relationship of overweight to healthy for effect on outcome is must less robust, but still significant. The p-value, 0.0354, is much larger than the value for obese, thus, less strongly related to expenses.

Next, we have the sexmale line in the output. It is telling us that males spend $134 dollars less than females on medical expenses. The p-value is greater than 5 percent. Therefore, its effect is insignificant. We will remove the variable from the model.

The variable for smokers looms large in medical expenses. The p-value is also very small. Compared to nonsmokers, smokers incur $23,875 greater medical expenses.

Finally, there are the three levels to the four-level region variable. By inspection of the p-values for its dummies, we can see that none of the regions incur a significant difference in expenses. We will remove the variable. Also notice that the reference region is northeast based on ascending alphabetical order.

It is good to inspect the confidence interval to the regression coefficients. The `confint` function allows us to inspect the intervals for width. Narrower is better than wider.

The intervals are presented in the lower portion of Output 7-6. For expediency we will only inspect them for the current model because other than width the p-values give us a sense of what we would find in an intervals table.

The default confidence interval is 95 percent. Therefore, not shown is how to set our own interval with the `confint` function. We would insert the argument `level =` with confidence as a decimal.

The table of confidence intervals show upper and lower as 5 percent divided by 2. However, notice what happens for any variable for which the p-value is greater than 0.05. Within its interval, we will observe a reversal of sign. In other words, zero effect is within the upper and lower limits. We would normally remove such a variable.

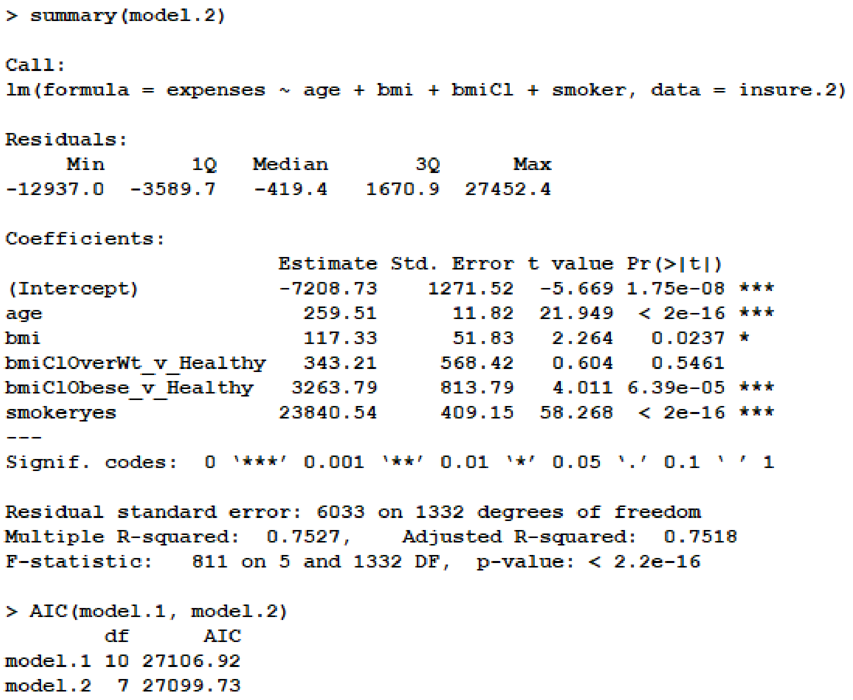
Model.2: BMI added, and sex and region removed. Recall that the analysis of standardized effects found that one standard deviation increase of BMI begets a 0.16 standard deviation increase of medical expenses. Therefore, the next model will add the variable to see if it matters in the overall operational context. Additionally, upon the insight of `model.1`, we will remove the `sex` and `region` variables.

The code below will return `model.2`. The returned model is shown in Output 7-7. Also notice the `AIC` function; for the first time.

```
#Add bmi and remove sex and region
model.2<- lm(expenses ~ age + bmi + bmiCl + smoker, data = insure.2)
```

```
summary(model.2)
AIC(model.1, model.2)
```

The output shows only minor improvement over `model.1`. `Model.2` explains a slightly greater amount of the overall variance to mean expenses. The residual standard error is only reduced by $10. The variable `bmi` adds $117 of expenses for an increase of one BMI. However, `bmi` is significant with a p-value of 0.025. We will leave it in the model.



```
> summary(model.2)

Call:
lm(formula = expenses ~ age + bmi + bmiCl + smoker, data = insure.2)

Residuals:
     Min       1Q   Median       3Q      Max 
-12937.0  -3589.7   -419.4   1670.9  27452.4 

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)    
(Intercept)           -7208.73    1271.52  -5.669 1.75e-08 ***
age                     259.51      11.82  21.949  < 2e-16 ***
bmi                     117.33      51.83   2.264   0.0237 *  
bmiClOverWt_v_Healthy   343.21     568.42   0.604   0.5461    
bmiClObese_v_Healthy   3263.79     813.79   4.011 6.39e-05 ***
smokeryes             23840.54     409.15  58.268  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6033 on 1332 degrees of freedom
Multiple R-squared:  0.7527,    Adjusted R-squared:  0.7518 
F-statistic:   811 on 5 and 1332 DF,  p-value: < 2.2e-16

> AIC(model.1, model.2)
        df      AIC
model.1 10 27106.92
model.2  7 27099.73
```

Output 7-7: Model.2 and the AIC score show small improvement over model.1.

Notice that the variable `overWt_v_Healthy` shows no significant cost increase over healthy. If we did a confidence interval with the `confint` function, we would find "0" between the upper and lower limits to a 95 percent confidence interval. Of course, we cannot surgically remove the dummy from the next model. Instead, we will try a different classification scheme; `bmiObese`.

Now to the `AIC` function. The Akaike information criterion (AIC) is an estimator of prediction error and, thereby, a relative comparison of statistical models from the same dataset. AIC estimates the relative amount of information lost by a given model. The less lost, the better the model. It follows that the lower the AIC, the comparatively better the model.

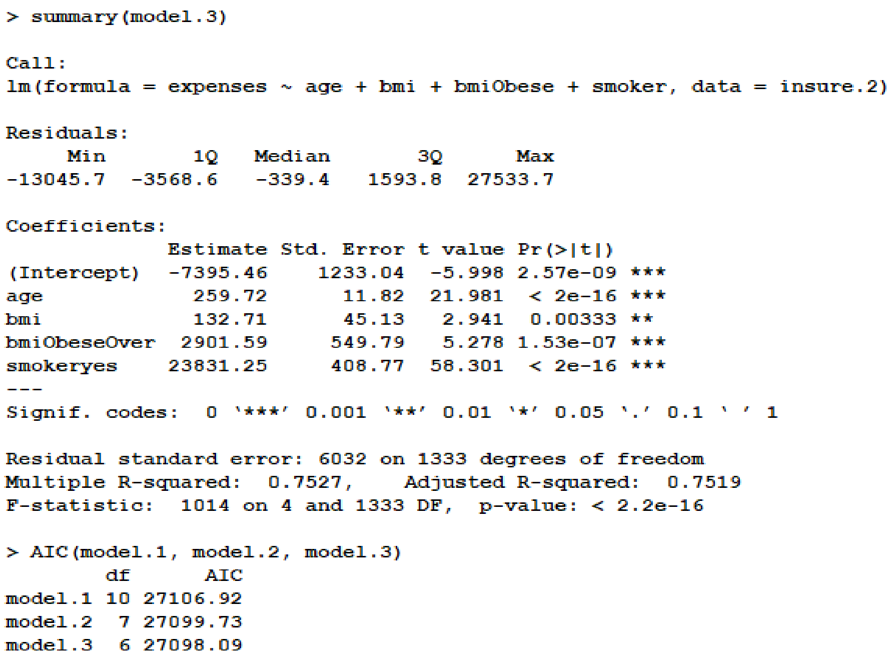
We can use the AIC to compare models. We can read the AIC score in the bottom section to the readout of Output 7-7. `Model.2` is an improvement over `model.1` as indicated by a reduction of AIC from 27106 to 27099. One reason is that the removed variables reduced the model's degrees of freedom from 10 to 7 while concurrently explaining a bit more of the total variance.

Model.3: Replace bmiCl with bmiObese. The three classification of the `bmiCl` variable do not seem to have much to contribute to `Model.2`. This suggests that the most indicative split is over and under a BMI of 30. For this reason, the `bimObese` variable was earlier created in the `insure.2` dataset. In real life, we would have arrived at that recognition and created the variable at this stage of trial and error toward the best model. However, our timing served the purpose of explaining dummy variables.

The code below is `model.3` with the `bmiObese` as our alternative to `bmiCl`. The `AIC` function is also included so that we can compare the model to previous models. The returned output is shown in Output 7-8.

```
#Replace bmiCl with bmiObese
model.3<- lm(expenses ~ age + bmi + bmiObese + smoker, data = insure.2)
summary(model.3)
AIC(model.1, model.2, model.3)
```

`Model.3` offers no additional predictive ability as indicated by R-squared. The important point is that we have reduced the number of variables without a collateral loss of predictability.



```
> summary(model.3)

Call:
lm(formula = expenses ~ age + bmi + bmiObese + smoker, data = insure.2)

Residuals:
     Min       1Q   Median       3Q      Max 
-13045.7  -3568.6   -339.4   1593.8  27533.7 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -7395.46    1233.04  -5.998 2.57e-09 ***
age            259.72      11.82  21.981  < 2e-16 ***
bmi            132.71      45.13   2.941  0.00333 ** 
bmiObeseOver  2901.59     549.79   5.278 1.53e-07 ***
smokeryes    23831.25     408.77  58.301  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6032 on 1333 degrees of freedom
Multiple R-squared:  0.7527,	Adjusted R-squared:  0.7519 
F-statistic:  1014 on 4 and 1333 DF,  p-value: < 2.2e-16

> AIC(model.1, model.2, model.3)
        df      AIC
model.1 10 27106.92
model.2  7 27099.73
model.3  6 27098.09
```

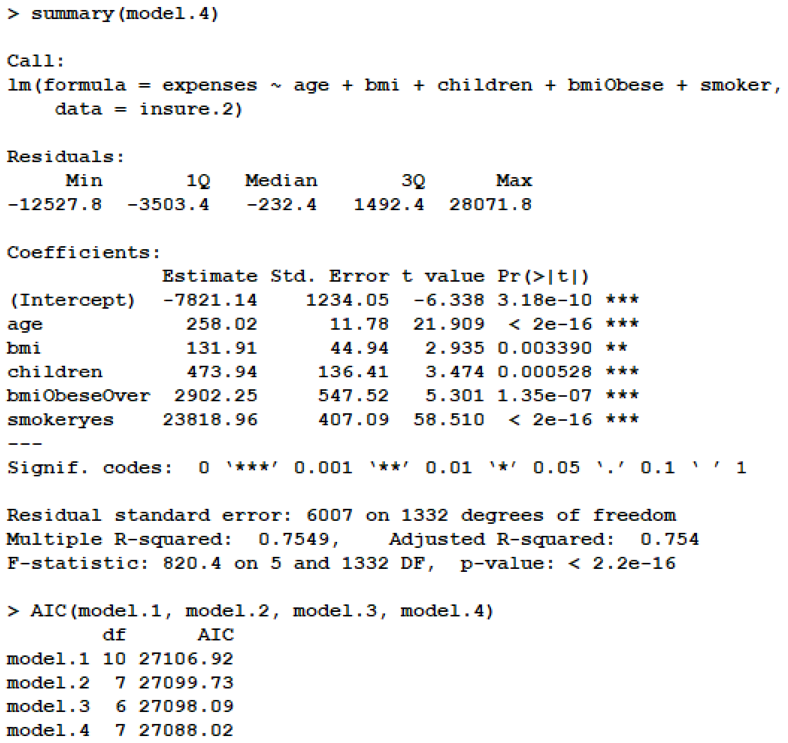
Output 7-8: Model.3 with a two-level distinction for obesity versus overweight and healthy combined.

Model.4: Add children. It was earlier found with the standardized effects test that the number of children is a potentially significant predictor of medical expenses. Not large but some. The code below includes the variable. The output is shown in Output 7-9.

```
#Add children
model.4<- lm(expenses ~ age + bmi + children +
   bmiObese + smoker, data = insure.2)
summary(model.4)
AIC(model.1, model.2, model.3, model.4)
```

The output shows that for each child, a family experiences $473 greater medical expense. The p-value of 0.0033 shows a strong relationship to expenses.

To summarize, we have added a variable but still caused a decrease in AIC after being penalized for the addition. Notice that R-squared has increased and residual standard error has decreased; neither impressively. Thus, the model explains a bit more of the overall variance. We will stop with the model.



```
> summary(model.4)

Call:
lm(formula = expenses ~ age + bmi + children + bmiObese + smoker,
    data = insure.2)

Residuals:
     Min       1Q   Median       3Q      Max
-12527.8  -3503.4   -232.4   1492.4  28071.8

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -7821.14    1234.05  -6.338 3.18e-10 ***
age            258.02      11.78  21.909  < 2e-16 ***
bmi            131.91      44.94   2.935 0.003390 **
children       473.94     136.41   3.474 0.000528 ***
bmiObeseOver  2902.25     547.52   5.301 1.35e-07 ***
smokeryes    23818.96     407.09  58.510  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6007 on 1332 degrees of freedom
Multiple R-squared:  0.7549,    Adjusted R-squared:  0.754
F-statistic: 820.4 on 5 and 1332 DF,  p-value: < 2.2e-16

> AIC(model.1, model.2, model.3, model.4)
        df      AIC
model.1 10 27106.92
model.2  7 27099.73
model.3  6 27098.09
model.4  7 27088.02
```

Output 7-9: Adding number of children improves the model with respect to R-squared and AIC.

7.2.4. Transforming Variables for Linearity

The assumption of linear regression is that the relationship between each numeric predictor and the model’s outcome is linear and that the variances of the residuals are evenly dispersed (homogenous) along the predictor variable. If not, we may be able to make them so with a transformation. We can apply transformations to the predictors or the outcome variables.

A transformation is found by trial and effects. We transform the variable in our dataset, run the model and form a scatter plot of the base transformed variable as the x axis versus the model’s residuals on the y axis. For a complete discussion of transformation see Ott and Longnecker [11.1 and 13.4].

This section will inspect the numeric variables for linearity and seek appropriate transformations if necessary. The mainstream possibilities are as follows with respect to patterns observed in a residuals-to-variable plots:

- Plot is a pattern increasing at a decreasing rate with roughly constant variance. Try transforming x to a square root, logarithm or inverse.

- Plot is a pattern increasing at an increasing rate with roughly constant variance. Try x and x^2 as predictors.
- Plot is a pattern that increases to a maximum and then decreases with roughly constant variance. Try x and x^2 as predictors. The transformation can be extended to higher orders for patterns with multiple changes of direction.
- Plot is a pattern increasing at a decreasing rate and variance around the curve increases as y increases. Try y^2 as the outcome variable.
- Plot is a pattern increasing at an increasing rate and variance to the curve increases as y increases. Try ln(y) for the outcome variable or predictor variable.

We will start our inspection for necessary transformations with the code below. The assignment of `insure.2` to `insure.3` is the analyst's preference. The idea is to lock in the dataset such that if subsequent actions damage the feedstock dataset, we only need to return and rerun the line of code.

```
#Assign insure.2 to insure.3
insure.3<- insure.2
```

The residuals are extracted from `model.4` and standardized with the code below. The standardized residuals are the residuals to the fitted outcome divided by the residual standard error. The `rstandard` function does the calculation for us.

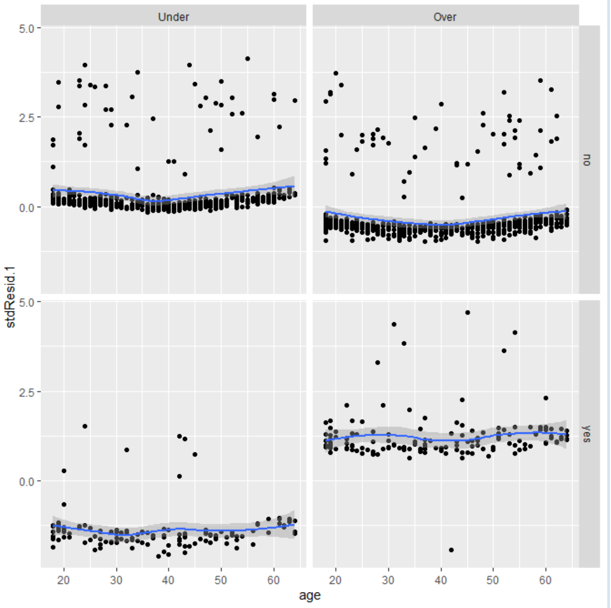
```
#Extract standardized residuals from model.4
#Residuals placed in insure.3
insure.3$stdResid.1<- rstandard(model.4)
```

We will first inspect the age variable for linearity and homogeneous variances. Recall that we need a scatter plot of standardized residuals as y and age as x. To visually test linear relationship, we want to fit a loess line to the plot. A straight-line linear fit can be misleading because it bridges rather than senses the effects of disturbing forces. The code below, using ggplot2, will return the plot of Output 7-10.

```
residAge<- ggplot(insure.3, aes(x = age,
   y = stdResid.1)) +
   geom_point() + geom_smooth() +
   facet_grid(smoker~bmiObese)
residAge
```

Notice that the code calls for subsetting our perspective on `smoker` and `bmiObese`. It is done with `facet_grid` function.

The returned plot shows curvature to a valley and upward again. The pattern is most pronounced for nonsmokers. The overall variance appears somewhat homogeneous but with a considerable number of outliers.



Output 7-10: Check for linearity of the age variable subset on smoker and bmiObese.

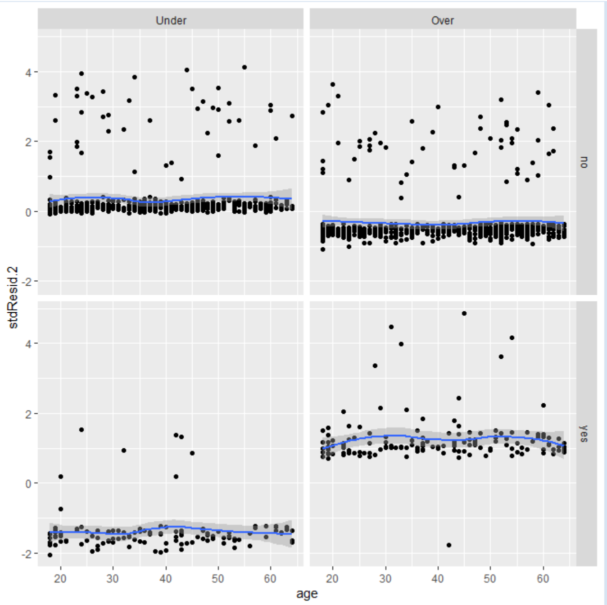
The code below adds the variable `age2` to `model.4` and renames it `model.5`. Again, we extract the standardized residuals from the model and place them in our dataset as `stdResid.2`. The third block returns the age-to-residuals plot shown in Output 7-11.

```
model.5<- lm(expenses ~ age + age2 + bmi + children +
   bmiObese + smoker, data = insure.3)
summary(model.5)

insure.3$stdResid.2<- rstandard(model.5)

residAgePol<- ggplot(insure.3, aes(x = age,
   y = stdResid.2)) +
   geom_point() + geom_smooth() +
   facet_grid(smoker~bmiObese)
residAgePol
```

Upon inspection we can see that our strategy for the age variable works well. The loess fits are now close to linear.



Output 7-11: Linearity is achieved by adding the age squared variable to the model.

However, we should also test, with the code below, whether the addition of the variable returns a better model. We will not show the returns, but they indicate an improvement. It is left to the reader to inspect what the code returns and compare the elements within the linear models.

```
#Compare models
AIC(model.4, model.5)
anova(model.4, model.5)
```

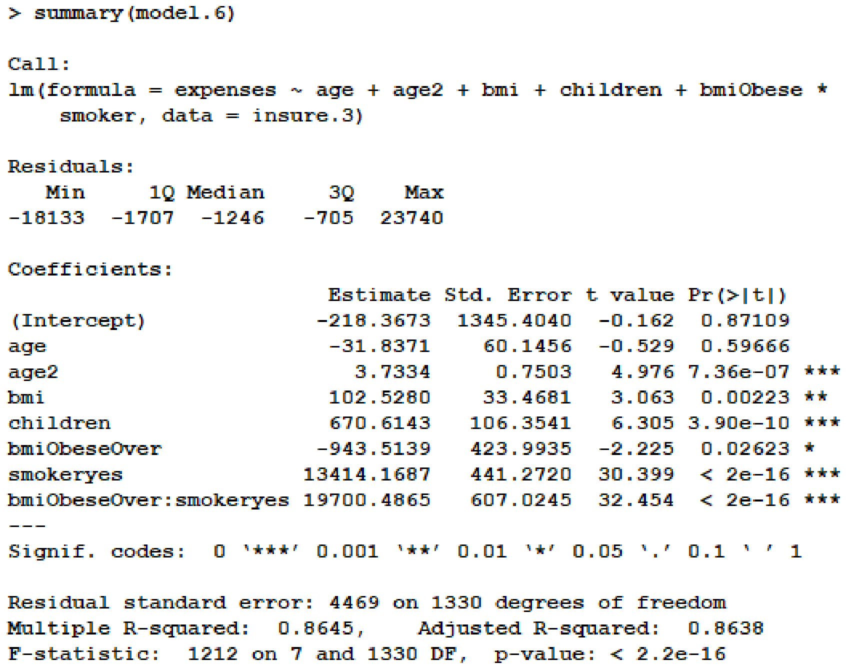
However, there is something else to notice in the plots. Notice that they are not centered on zero as standardized residuals should be. This may suggest that the `smoker` and `bmiObese` variables are interacting with each other. Interaction is defined as the effect of a variable on an outcome depends on the state of a second variables.

The code below creates a two-way interaction between `bmiObese` and `smoker` with the "*" in `bmiObese*smoker`. The new model is named model.6 and its results are shown in Output 7-12.

```
#Insert bmiObese and smoker interaction
model.6<- lm(expenses ~ age + age2 + bmi + children +
   bmiObese*smoker, data = insure.3)
summary(model.6)
```

This is the first time we have seen an interaction in play. If we wanted to start with an interaction between all variables, we would have written our model as expenses ~ `age*age*bmi*children*bmiObese*smoker`.

That would have created a six-way interaction of all orders down to two-way and main effects. Thence our modeling process, would eliminate insignificant interactions to the model from the highest order down. Any lower order interaction or main effect variable to a significant higher order interaction must be retained. If the six-way interaction proved to be significant, we would leave all else in the model as it is. In this case, we would have arrived at a two-way interaction between the `bmiObese` and `smoker` variables.



```
> summary(model.6)

Call:
lm(formula = expenses ~ age + age2 + bmi + children + bmiObese *
    smoker, data = insure.3)

Residuals:
   Min     1Q Median     3Q    Max
-18133  -1707  -1246   -705  23740

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)             -218.3673  1345.4040  -0.162  0.87109
age                      -31.8371    60.1456  -0.529  0.59666
age2                       3.7334     0.7503   4.976 7.36e-07 ***
bmi                      102.5280    33.4681   3.063  0.00223 **
children                 670.6143   106.3541   6.305 3.90e-10 ***
bmiObeseOver            -943.5139   423.9935  -2.225  0.02623 *
smokeryes              13414.1687   441.2720  30.399  < 2e-16 ***
bmiObeseOver:smokeryes 19700.4865   607.0245  32.454  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4469 on 1330 degrees of freedom
Multiple R-squared:  0.8645,	Adjusted R-squared:  0.8638
F-statistic:  1212 on 7 and 1330 DF,  p-value: < 2.2e-16
```

Output 7-12: Model.6 with an interaction between bmiObese and smoker.

Output 7-12 reports that there is a tremendous improvement. The model now explains 86 percent of variance to the null model; up from 75 percent. The residual standard error is now $4,469 down from $5,980.

Let's inspect the interaction in the model and how we interpret it in our evaluation. The interaction, `bmiObeseOver:smokeryes`, is strongly related to expenses. This is indicated by its p-value.

However, let's make another point. The main effect variables `bmiObese` and `smoker` cannot be given meaning for the strength of their relationship to outcome. The rule is that the significance of any main effect variable should not be interpreted if included in an interaction.

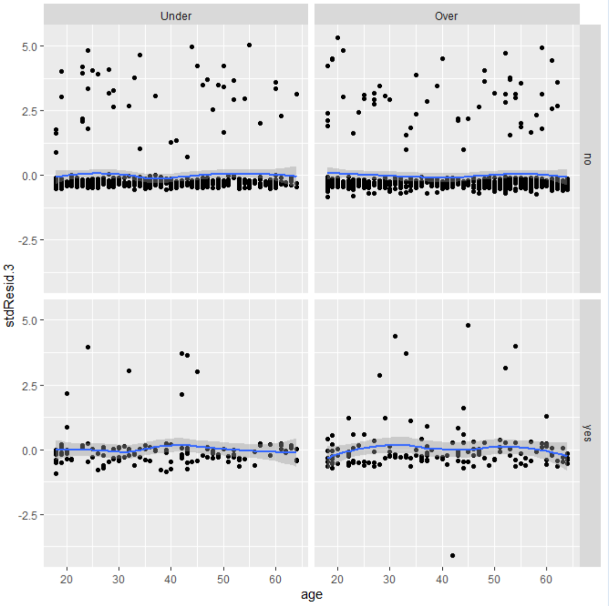
Also note that age is shown as insignificant. Why do we not remove it? We do not because its effects are shared with age2 as an element of the polynomial f(age) = age + age^2.

Now to review the effect of the interactions on the model. The code below extracts the standardized residuals to model.6 and places them as stdResid.3 in our dataset. In turn, the ggplot2 code will return the residuals-to-age plot shown in Output 7-13.

```
insure.3$stdResid.3<- rstandard(model.6)

residIntr<- ggplot(insure.3, aes(x = age,
   y = stdResid.3)) +
   geom_point() + geom_smooth() +
   facet_grid(smoker~bmiObese)
residIntr
```

The residuals to a model must center on zero because the mean of standardized residuals is zero. As we saw in Output 7-12, that was not the case without the interaction. As can be seen in Output 7-13, that has been remedied by including the interaction of variables. All is now pinned to zero.



Output 7-13: Residual plot for age now centered on residuals with mean equal zero.

We can also see another issue that will be spoken to in a later section. There seems to be outlier and possibly influencing cases to the outcome. We will need to make distinctions in their nature.

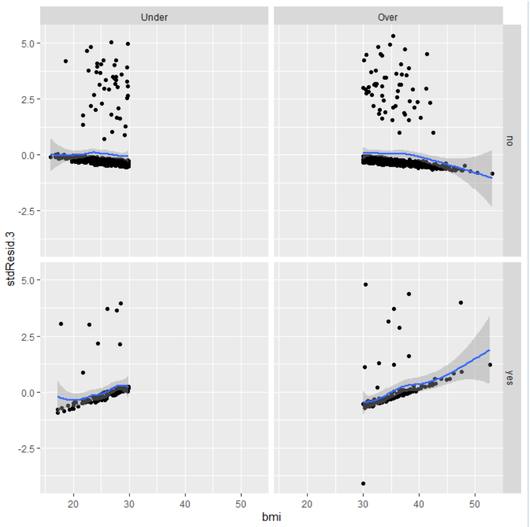
The code below will compare the three models from age to age2 to interaction of bmiObese and smoker. It is left to the reader to run the code. The reader should also run the models and inspect the improvement from model to model.

```
AIC(model.4, model.5, model.6)
anova(model.4, model.5, model.6)
```

We will next evaluate the `bmi` variable for necessary transformation. The code below creates the visual. Because we are continuing with `model.6`, we do not need to call for a new standardized residuals variable.

```
#Evaluate bmi
residBmi<- ggplot(insure.3, aes(x = bmi,
   y = stdResid.3)) +
   geom_point() + geom_smooth() +
   facet_grid(smoker~bmiObese)
residBmi
```

Output 7-14 indicates linearity. Because `bmi` is colinear to `bmiObese` we can see the influence of the interaction. However, notice the loess fit seems to be pulled and curved a bit by the outlier points above the tightly linear group of cases.



Output 7-14: Residuals to bmi show linearity, but outliers pull the plot northward.

Finally, we will inspect the children variable. The code below will generate the chart upon which to make the evaluation. The findings are that the variable can remain untransformed. It is left to the reader to run the code for their own confirmation.

```
residChlInt<- ggplot(insure.3, aes(x = children,
   y = stdResid.3)) +
   geom_point() + geom_smooth(method = loess) +
   facet_grid(smoker~bmiObese)
residChlInt
```

Now we have explored our model from basic development through to whether we can improve it by making transformations. We made age to be a polynomial and added an interaction to the model. The polynomial was a significant improvement as compared by AIC and the anova test. The interaction was a huge improvement because the fit revealed that expenses change to different levels of the interacting variables.

7.2.5. Test the Model for Generalization

Next is to test the model for generalization with respect to assumptions of independence of errors, collinearity, linearity of predictions to residuals, constancy of variance and normal distribution of residuals.

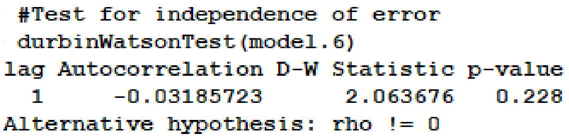
Upon the findings we will introduce the ramifications of pass-fail with respect to "generalization." It is incorrectly assumed that a "passing" model is mandatory. The typical expression, "test for accuracy," contributes to the misperception.

Let's begin with the assumption of independence of errors. We are testing for what is called autocorrelation. It is defined as the outcome of the current period influenced by one or more previous periods. That would not be the case for the data to this chapter. If we were domain experts, we would know that our dataset does not hold chronological variables.

Autocorrelation and testing for it will be a topic to Chapter 10. For a complete explanation of regression, we will still see how the test is conducted with the `durbinWatsonTest` function as coded below. The output is Output 7-15.

```
#Test for independence of error
durbinWatsonTest(model.6)
```

The null hypothesis of the test is that there is no autocorrelation in the outcome variable. Notice in the Output that the p-value is greater than 0.05. Therefore, we accept the hypothesis. Our data are independent rather than autocorrelated.



```
 #Test for independence of error
 durbinWatsonTest(model.6)
lag Autocorrelation D-W Statistic p-value
  1     -0.03185723      2.063676   0.228
Alternative hypothesis: rho != 0
```

Output 7-15: The data is independent as shown by a p-value greater than 0.05.

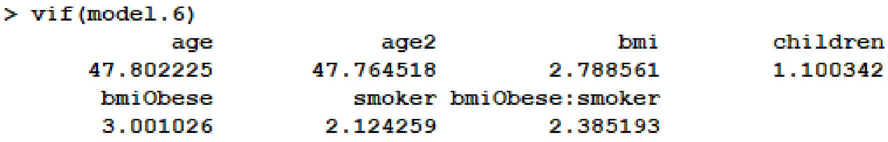
At this stage, we also test the model with respect to variance inflation factor (VIF). It measures multicollinearity in regression analysis. Multicollinearity is when there is correlation between predictors in a model. The VIF estimates how much the variance of a regression coefficient is inflated due to multicollinearity in the model.

Of course, multicollinearity is the expectation rather than the exception. However, the presence of extreme multicollinearity can adversely affect the model through the regression coefficients. When there is a presence, we rely upon our domain expertise to decide which variables to remove without undermining our quest for a particular insight.

We measure VIF with the `vif` function. The code is below. Output 7-16 is the returned output.

```
#Test model.6 for multicolinearity
vif(model.6)
```

Exactly how large a VIF must be before it causes a problem is a subject of debate. It seems to be accepted that a VIF above ten indicates high correlation and is cause for concern. The findings are that, except of `age` and `age2`, the choice of variables do not cause a factor beyond ten.



```
> vif(model.6)
          age            age2             bmi        children
    47.802225       47.764518        2.788561        1.100342
     bmiObese          smoker bmiObese:smoker
     3.001026        2.124259        2.385193
```

Output 7-16: The VIF returned show no multicollinearity.

Sometimes a high VIF is not a concern such as for `age` and `age2`. We can get a high VIF when including products or powers of variables. An example is x and x^2, as we did for age.

Another example is that high VIFs may occur for dummy variables representing categorical variables with three or more levels. They too are usually not a problem.

Next, we will inspect the plot of residuals to predicted expenses. The objective is to assess the linearity of the outcome variable and the constancy (homoscedasticity) of residuals to the outcome variable.

To begin, we extract the fitted values from `model.6` and place them in our dataset. The following code does that. The newly created variable is named `fitted`.

```
#add the fitted points of model.6 to insure.3
insure.3$fitted<- model.6$fitted.values
```

Let's give some attention to the `fitted.values` variable. If we run the code `str(model.6)` we would discover that everything to the model is actually a data object among many objects. As compared to a data frame or matrix it is called a list object. This is because any type of object can be included without regard to type. In other words, we are looking at every aspect of the model as data. One object is a vector of fitted values.

The point is that we can reach into any object to the list, pull out specific content and place it in another object as we just did with the code. Another example would be to reference the element as an argument to another analytic.

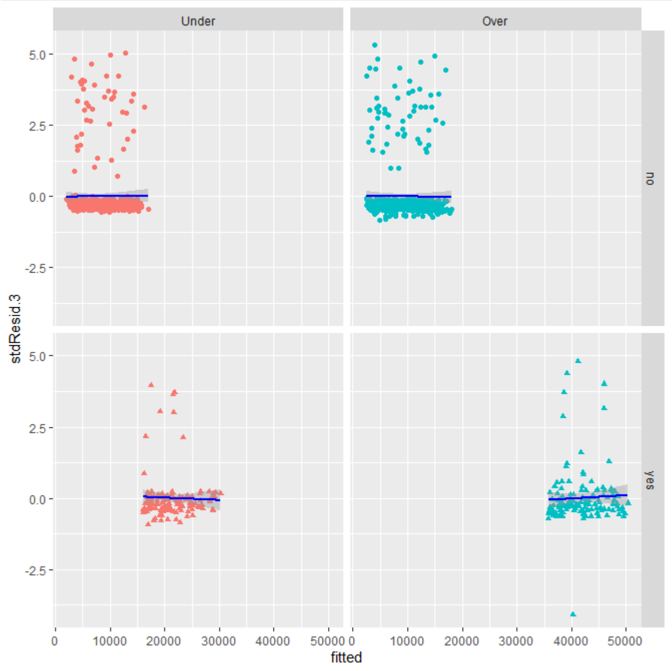
We will use the fitted variable to generate a scatter plot of standardized residuals to the fitted predictions. We are looking for linearity with respect to the model's outcome just as we did for each variable in the previous stage to look for necessary transformations.

The chart of Output 7-17 is the outcome of the code below.

```
#Student stdResid.3 is from model.6
residModel.6<- ggplot(insure.3,
   aes(fitted, stdResid.3, )) +
   geom_point(aes(color = bmiObese, shape = smoker)) +
   geom_smooth(method = "lm", color = "Blue") +
   facet_grid(smoker~bmiObese) +
   theme(legend.position = "none")
residModel.6
```

In the output, the loess plot confirms linearity between expenses and model residuals. We can also see that the residuals are roughly constant but skewed along the plot. We can also see that our non-smoker subsets are more likely to be found as not normally distributed due to substantial numbers of outliers.

Next, we will test the model through the normal distribution of its residuals. Of course, we hope to find that they are but Output 7-17 suggests that they are not. We will use a statistical test and two visualizations to evaluate the distributions.

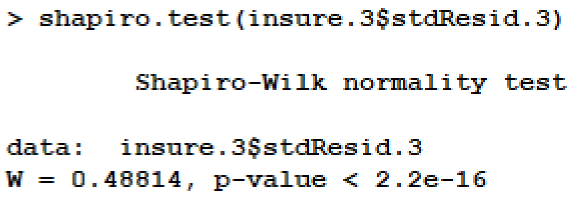


Output 7-17: Expenses, as fitted variable, against standard residuals to model.6.

We will first statistically test for normality with the `shapiro.test` function as coded below. The returned output is shown in Output 7-18.

```
#Test for normal distribution
shapiro.test(insure.3$stdResid.3)
```

The null hypothesis of the Shapiro test is that there is no difference between the variable, `stdResid.3`, and a normal distribution. The p-value that is much smaller than 0.05 would cause us to reject the hypothesis. Our data is not normally distributed as will be dramatically confirmed by graphic tests to follow.



```
> shapiro.test(insure.3$stdResid.3)

        Shapiro-Wilk normality test

data:  insure.3$stdResid.3
W = 0.48814, p-value < 2.2e-16
```

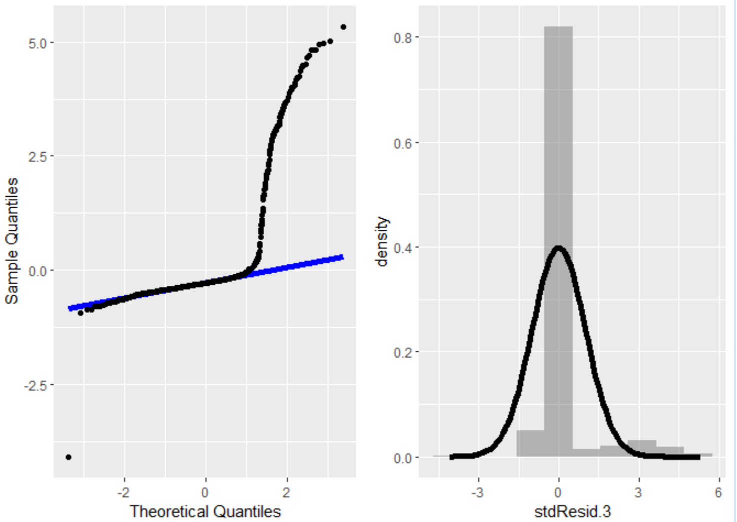
Output 7-18: The test shows to reject the null hypothesis that the distribution is normal.

We will use two plots to further inspect the normality of residuals. They are the Q-Q and density plots. They were explained in section 5.2.2. The code below will present them side by side in Output 7-19.

```
#Q-Q test
qqModel.6<- ggplot(data = insure.3,
        mapping = aes(sample = stdResid.3)) +
        stat_qq_line(color = "Blue", size = 2) +
        stat_qq_point() +
        labs(x = "Theoretical Quantiles",
        y = "Sample Quantiles")
qqModel.6

#Density plots.
hstModel.6<- ggplot(insure.3, aes(stdResid.3)) +
   geom_histogram(aes(y = ..density..), bins=10, alpha = .4) +
   stat_function(fun = dnorm, args = list(mean = mean(insure.3$stdResid.3,
   na.rm = TRUE),
   sd =sd(insure.3$stdResid.3, na.rm = TRUE)), size = 2)
hstModel.6

#Figure Q-Q plot and Histogram to model.6
QqHstModel.6<- ggarrange(qqModel.6, hstModel.6,
   ncol = 2, nrow = 1)
QqHstModel.6
```



Output 7-19: Q-Q and density plots to inspect for normal distribution.

Recall from earlier chapters, a normal distribution of fitted predictions will track the straight line of the Q-Q plot. It does not. We can see why in the layered density plot. The normal distribution for data with the mean and variance of the expenses variable are layered on the density plot to expenses.

We can see that the left tail is heavy and the data are strongly skewed to the right. The distribution is also highly peaked. This is the consequence of the many high-side outliers to the model and of the mass of points clustered tightly along the loess fit.

Therefore, our model does not pass the full set of tests for generalization. However, we have gathered many insights as we built it. That does not mean the model is not valuable, or as some would say, “not accurate.”

The conclusion is that, although the model fits our data very well, we cannot generalize the model. We cannot apply the model to make predictions regarding similar operations. The model is only applicable to the operation and the data to the model. Furthermore, the model will likely “remain good” to the operation by periodically updating its feed data.

Nor is the inability for generalization problematic such that we pine for it. As is apparent in this chapter, once an SME builds one model the experience teaches them to build any model. Furthermore, the insight out of the design of one model will likely travel to equivalent operations. In other words, the issue is insight. It is not that machine learning is difficult and that generalized AI is our grail.

We can guess that the model’s ability to generalize is undermined by the outliers. However, rather than “messing up our model” they will be either cases of interesting or bad data. Discovering which is worth the price of admission.

If we investigate each outlier, we may find the record is legitimate. Therefore, our question is why and how we should act on with what is found. Or we may realize, as domain experts, that there are variables we should seek out or create to include in the model to better fit the cases as part of our operation’s reality. If we find the case data is bad, we can eliminate the record or impute an estimate.

We may also want to break the model into sub-models to get to normal distribution. Let’s subset and explore the density plots for smoker and bmiObese. Recall from Chapter 5 that density plots allow us to compare the shape of the data, unaffected by the centering and spread characteristics of the compared variables.

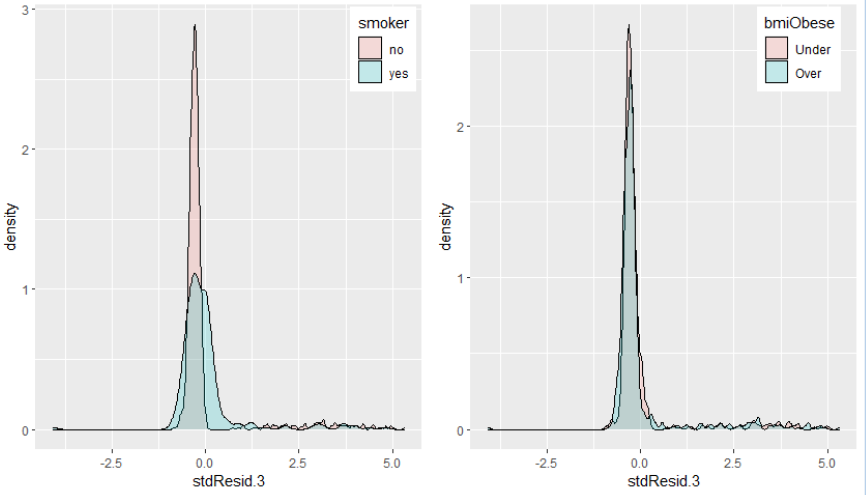
The code below will return visuals to both variables side by side in Output 7-20.

```
#Density model to compare smoker distributions
polyDen.Smoke<- ggplot(insure.3, aes(stdResid.3, fill=smoker),
bins = 10) +
      geom_density( alpha = 0.2) +
      theme(legend.position = c(.9, .9))
   polyDen.Smoke

   #Density model to compare bmiObese distributions
   polyDen.Obese<- ggplot(insure.3, aes(stdResid.3, fill=bmiObese),
   bins = 10) +
      geom_density( alpha = 0.2) +
      theme(legend.position = c(.85, .9))
   polyDen.Obese

   #Figure histograms to model.6 on smoker bmiObese
   DenModelInter<- ggarrange(polyDen.Smoke, polyDen.Obese,
      ncol = 2, nrow = 1)
   DenModelInter
```

The dual figures suggest that we may find the biggest concentration, skew and heavy tail in the non-smoker subset. Output 7-17 showed many more outliers for nonsmokers than for smokers. Note that the bmiObese levels have a stronger match. Accordingly, we may want to deepen our exploration by building models subdivided by the levels of the smoker variable. That is left to the reader.



Output 7-20: The distributions are distinctive for smokers but not for bmiObese.

By now, we can see that we will likely not follow a direct path from load, build and evaluate a regression. Instead, we will take cycles of investigation and discovery. It is very likely that our whole purpose is to see and learn from what emerges as a model is built 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.

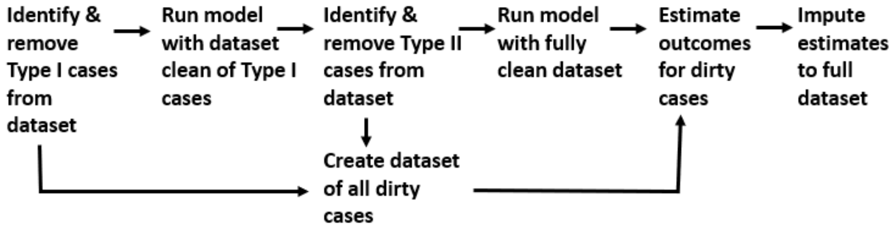

Identify & remove Type I cases from dataset
Run model with dataset clean of Type I cases
Identify & remove Type II cases from dataset
Run model with fully clean dataset
Estimate outcomes for dirty cases
Impute estimates to full dataset
Create dataset of all dirty cases

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.

Data and Analytics Skills for Your Career Security

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

A look inside

Richard G. Lamb

Every act of planning, organizing, execution and control leaves a data trail. In turn, the data are feedstock to every aspect of data-driven operations. Until this chapter in the explanation of data-driven operations, all has been on the presumption of entirety of data. Entirety is defined as all captured data are layered, current to capture and accessible on demand. Few operations have achieved the standard.

Chapter 8: Achieve Entirety of Data

Excerpt:

8.1. Layered Structure

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside
Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds

Note:

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 grpahics to the excerpt are in their original color

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.

- A table is tabular in its variables. For example, rather than a column for each shift, the day and night shifts are categories to single variable, e.g., Shift.
- Along the left side there are no titles for row, sections and subsections, and subtotals and totals.
- There are no separation (empty) rows.
- Only horizontal calculations are allowed and the results are a variable to the table.
- A table is continuous.

Let's expand on the criterion of "continuous." Imagine an Excel file with individual sheets for a group such as periods, cost centers, etc. Instead, the group of the sheets are made a variable, thus, making the table continuous. Many sheets become a single sheet.

Chapter 3 not only showed this format but demonstrated why it matters. Figures 8-3 and 8-4 are examples of data formatted as tables.

In contrast, Figure 8-1 is a spreadsheet to an operational task conducted with Excel. We can see in it that there are no rules to a spreadsheet format other than the integrity of its information.

The following rules of a table format are broken in the shown spreadsheet:

- There are multiple levels to the column headers.
- Each location and order type are a variable, rather than two tabular variables.
- Along the left side, titles are given to every row.
- There are separation rows.
- There are vertical calculations to variables: totals and subtotals.
- The spreadsheet is not one in a continuous spreadsheet but a single day to a daily report.

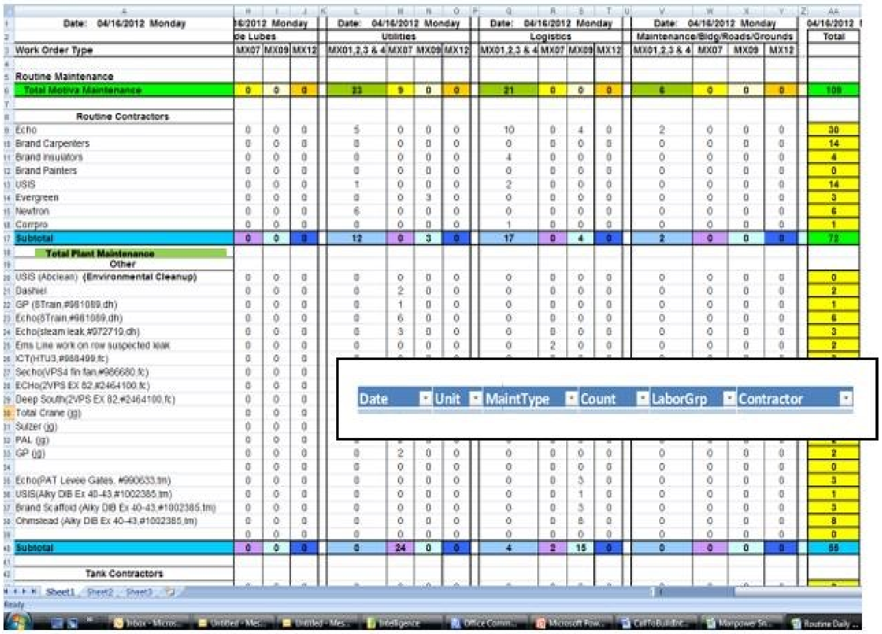
What we see in the spreadsheet is also defining. Data, calculation, filtering and presentation are fused into a single static object. If we want to utilize the data embedded in the spreadsheet, we must extract it manually. At the least, extraction is a substantial task. At the most, as it often is, it is a prohibitively laborious task.

To apply some nontechnical terminology, the report is a hot mess. It was the bane of daily existence to dozens of role holders in its daily data gathering and preparation.

Now to define what is a layered structure and why we care so much. As mentioned all operating systems work on the principle of layers, no exceptions.

The principle of layers is simple. Data and report are separate objects. The report is a layer on top of a data table as a separate layer. In contrast, and as already mentioned, data and report are "fused" in a spreadsheet as single static object.

How we move from a spreadsheet to a layered structure is easy to see. As the insert demonstrates, we would find that the fused spreadsheet of Figure 8-1 entails six variables: Date, Unit, MaintType, LaborGrp and Contractor. All fused spreadsheets are constructed upon data that are, in fact, an identifiable set of variables.



	A	H	I	J	L	M	N	O	Q	R	S	T	V	W	X	Y	AA
1	Date: 04/16/2012 Monday	16/2012 Monday			Date: 04/16/2012 Monday				Date: 04/16/2012 Monday				Date: 04/16/2012 Monday				04/16/2012 I
2		pe Lubes			Utilities				Logistics				Maintenance/Bldg/Roads/Grounds				Total
3	Work Order Type	MX07	MX09	MX12	MX01,2,3 & 4	MX07	MX09	MX12	MX01,2,3 & 4	MX07	MX09	MX12	MX01,2,3 & 4	MX07	MX09	MX12	
4																	
5	Routine Maintenance																
6	Total Motiva Maintenance	0	0	0	23	9	0	0	21	0	0	0	6	0	0	0	109
7																	
8	Routine Contractors																
9	Echo	0	0	0	5	0	0	0	10	0	4	0	2	0	0	0	30
10	Brand Carpenters	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14
11	Brand Insulators	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	4
12	Brand Painters	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	USIS	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	14
14	Evergreen	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	3
15	Newtron	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	6
16	Corrpro	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
17	Subtotal	0	0	0	12	0	3	0	17	0	4	0	2	0	0	0	72
18	Total Plant Maintenance																
19	Other																
20	USIS (Abclean) (Environmental Cleanup)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	Dashiel	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	2
22	GP (8Train,#981089.dh)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
23	Echo(8Train,#981089.dh)	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	6
24	Echo(steam leak,#972719.dh)	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	3
25	Ems Line work on row suspected leak	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	2
26	ICT(HTU3,#988499.fc)	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
27	Secho(VPS4 fin fan,#986680.fc)	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
28	ECHo(2VPS EX 82,#2464100.fc)	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
29	Deep South(2VPS EX 82,#2464100.fc)	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
30	Total Crane (jg)	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
31	Sulzer (jg)	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
32	PAL (jg)	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
33	GP (jg)	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	2
34		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	Echo(PAT Levee Gates, #990633.tm)	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	3
36	USIS(Alky DIB Ex 40-43,#1002385.tm)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
37	Brand Scaffold (Alky DIB Ex 40-43,#1002385.tm)	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	3
38	Ohmstead (Alky DIB Ex 40-43,#1002385.tm)	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	8
39		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	Subtotal	0	0	0	0	24	0	0	4	2	15	0	0	0	0	0	55
41																	
42	Tank Contractors																

Date	Unit	MaintType	Count	LaborGrp	Contractor

Figure 8-1: Spreadsheets violate every rule of a data table.

Therefore, the "first layer" is a table designed to capture and store the six variables for each record (row) upon which the spreadsheet is constructed. Of the six, the date variable makes the table continuous.

For each record, those who submit information to the process merely fill in a row to the table with the data. Done and done!

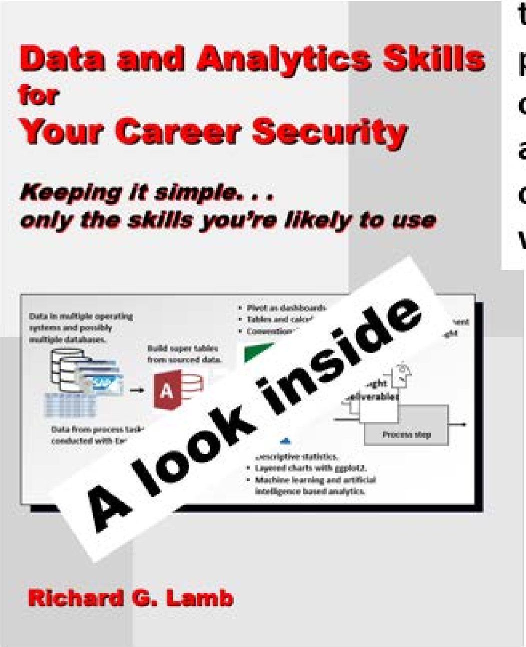
In contrast, entering data in a fused spreadsheet is akin to completing a form by tediously inserting each item of the data in the appropriate cell. Furthermore, any task that requires thought and concentration invites errors in both the data and report.

In the "second layer" of a layered structure are the task standard reports and data analytics. They reach into the table layer and return their deliverables. As time passes, records are added to the table. Rather than a significant task, the reports and analytics are instantaneously updated on command.

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.



Data and Analytics Skills for Your Career Security

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

A look inside

Richard G. Lamb

Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Data from process tasks conducted with Ex
Pivot as dashboards
Tables and calc
Convention
Insight deliverables
Process step
Descriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

Super tables are not just feed to analytics, Excel pivots and slick dashboards. Just as importantly they can be final insight deliverables from a process of queries fed by tables extracted from our operating systems. The chapter flowcharts and demonstrates a complex monthly operational plan and control process conducted with the MS Access software.

Chapter 9: Workload-Based Budget and Variance

Excerpts:

9.1. Framework for Budget and Variance

9.2. From History to Budget to Variance

9.3. Variance as Systemic or by Outliers

9.4. Maintenance Craft Capacity

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside
Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds

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 availably 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.

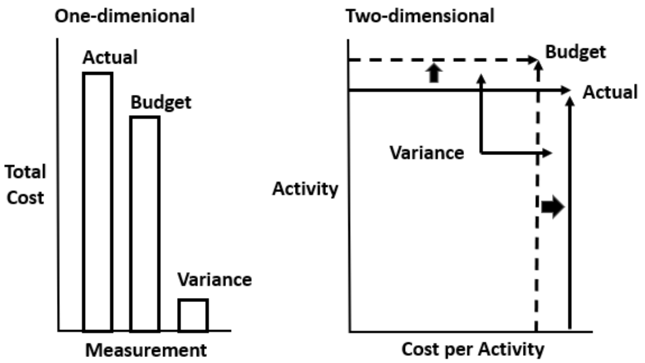


Figure 9-1: The contrast between one- and two-dimensional structures is insight.

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.

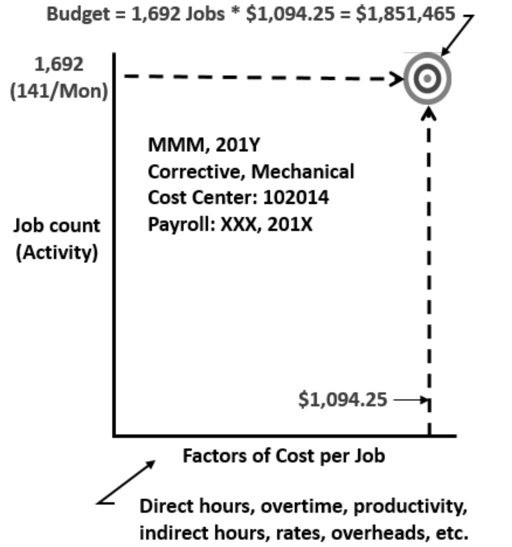


Figure 9-2: To be meaningful, budgets must be the intersection of two dimensions.

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.

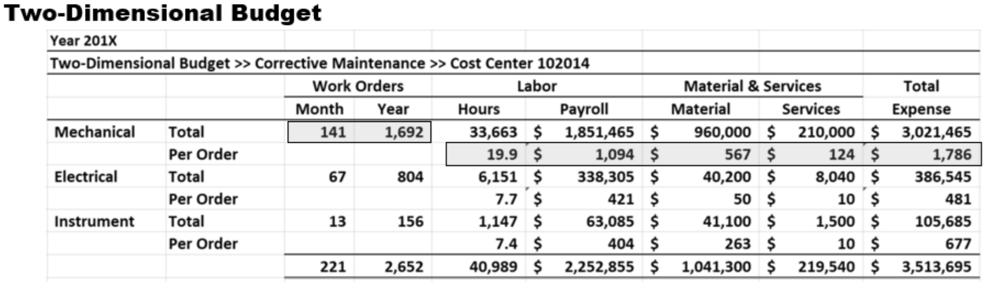
Whereas, the activity dimension ties back to the plant's aggregate production plan and staying abreast of its deterioration, the factors-of-cost dimension has its own connections to profit and enterprise ROI. First is the indirect connection to revenue through the adequacy of resources available to deliver 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.

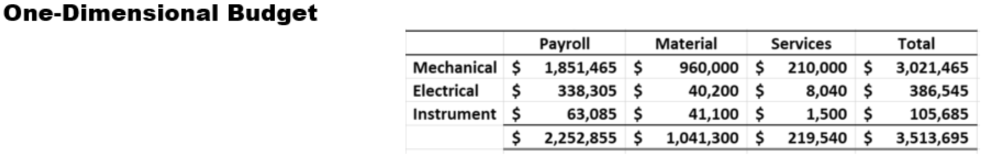


Two-Dimensional Budget

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.

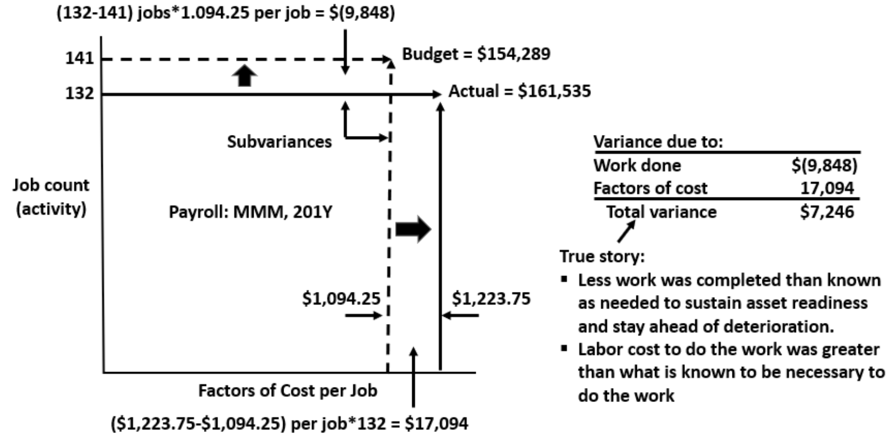

(132-141) jobs*1.094.25 per job = $(9,848)
141
132
Budget = $154,289
Actual = $161,535
Subvariances
Job count
(activity)
Payroll: MMM, 201Y
$1,094.25
$1,223.75
Factors of Cost per Job
($1,223.75-$1,094.25) per job*132 = $17,094
Variance due to:
Work done $(9,848)
Factors of cost 17,094
Total variance $7,246
True story:
▪ Less work was completed than known as needed to sustain asset readiness and stay ahead of deterioration.
▪ Labor cost to do the work was greater than what is known to be necessary to do the work

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:

Variance due to count = (AJC – BJC)*BHJ (9-1)
Variance due to hours = (AHJ – BHJ)*AJC (9-2)
Where:
BJC = Budget job count.
AJC = Actual job count.
BHJ = Budget hours per job.
AHJ = Actual hours per job.

It is also important to note that the calculation of the "due to" variances cannot be made on the back of a one-dimensional budget. There is no baseline upon which to do so.

The figure's table of calculated variances dramatically demonstrates what is forfeited with a one-dimensional structure. The capability to make decisions and take actions in response to the true story is lost. Instead, plants are making decisions and taking actions on the misinformation that one-dimensional variances almost always are.

The table of payroll variances for the budget group tells the true story. In this case, actual payroll overran budget by $7,246. Note that this is the totality of all that can be known out of a one-dimensional structure.

The total net overrun is not huge. It is approximately 4.7 percent of the month's budget. However, when we pull back the two-dimensional curtain, the revealed true bad is 17.5 percent of the month's spending. The variance of both, in a bad direction, is significantly greater than the net variance.

The revealed true bad shows that total overrun is misinformation. Instead, we would find that the subvariance for activity bodes poorly for the plant's ability to deliver its planned aggregate production and stay abreast of deterioration. Less work was done then is known by budget to be necessary during the period. We would also find that the cost to do the work that was completed is much greater than it should have been. This would suggest our activity did not fall short for lack of craft force capacity.

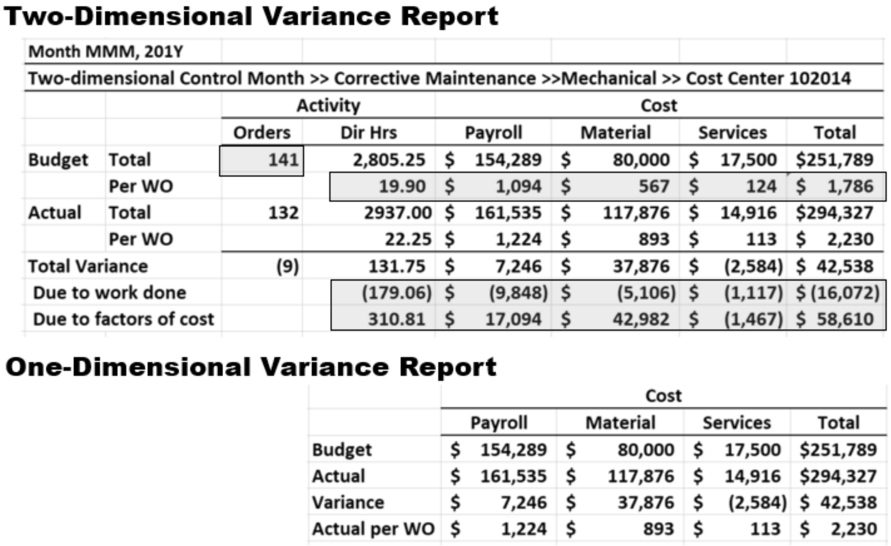
Now let's step from the concept of dimensions to an actual working report. Figure 9-5 shows, as contrasts, the two- and one-dimensional monthly variance report for a single budget group.

As seen in Figure 9-5, the one-dimensional variance report only reports the differences between the month's total budgeted and actual spending. However, the true, actionable story can only be known if the variances are reported as their dimensional sub-variances: variance due to work done and variance due to factors of cost.

In the figure, the subvariances are reported in the two-dimensional variance report. In contrast, they are conspicuously absent from the one-dimensional variance report.

It is easy to see what is gained when all variances are presented in the two-dimensional variance report. We can see in the report, the scenario of insight instead of misinformation that was described and shown in its Figure 9-4. Without a two-dimensional variance report, maintenance professionals can remember times that their plants' management inappropriately responded to one-dimensional totals. They zigged when they should have zagged. Later, the innocent were punished and the guilty were rewarded.

Two-dimensional variance reporting is doubly exciting. It allows so much more than just to report subvariances. Because each variance is tied to an analytical workup, it follows 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 duel 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 defendable 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.

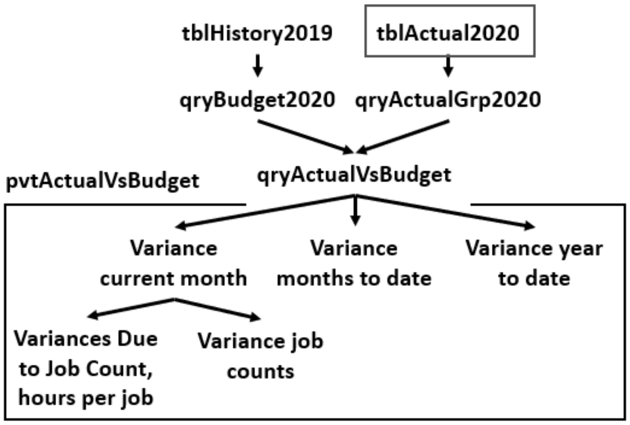


Figure 9-6: Schematic of the budget and variance process.

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. Variance as Systemic or by Outliers

Is the variance systemic, due to outliers or both? We essentially back into the assessment. Of course, we always want to know of our outliers, but are there outliers with too much influence on the group average? After accounting for outliers, are we still systemically off target?

This section will demonstrate the method to locate outliers. The section will then describe approaches to evaluate for systemic variance once the outliers are accounted for.

Chapter 3 introduced Student T and Z-score as boundary measures with which to flag outliers. We will use the Student T approach rather than the Z-Score. The reason is to adjust the confidence limits by which an order is considered as an outlier. The Student T adjusts the measure to reflect the count of jobs in each budget group. The Z-score does not.

When the count is infinity, Student T and Z-scores are equal. Recall that to a normal distribution, 68 percent of all observations will fall within one standard deviation. The statistical reality is that, short of infinity, less than 68 percent of the observations will fall within one standard deviation.

This can be seen by contrasting the measures at 95 percent confidence. The Z-score is +/- 1.96. In contrast, the Student T for 20 observations is +/- 2.08.

Let's make a final note of the importance of being able to seek true outliers. Without a method to seek true outliers, some plants investigate the top ten most costly orders. The problem is that, more times than not, they are legitimately the costliest orders rather than outliers. However, the purpose of budget and variance is to find the orders that are. The human difference is that we waste our time investigating what does not call for investigation.

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.

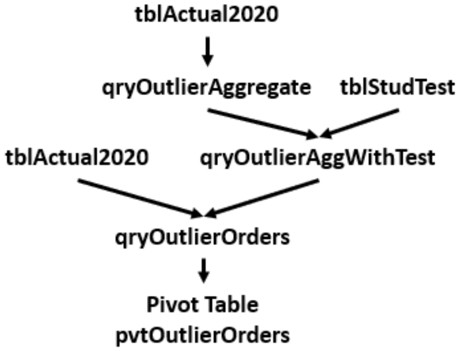


Figure 9-8: Schematic to return table of outlier orders for year to date.

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.

9.4. 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.

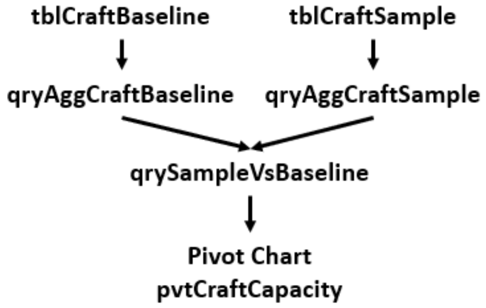
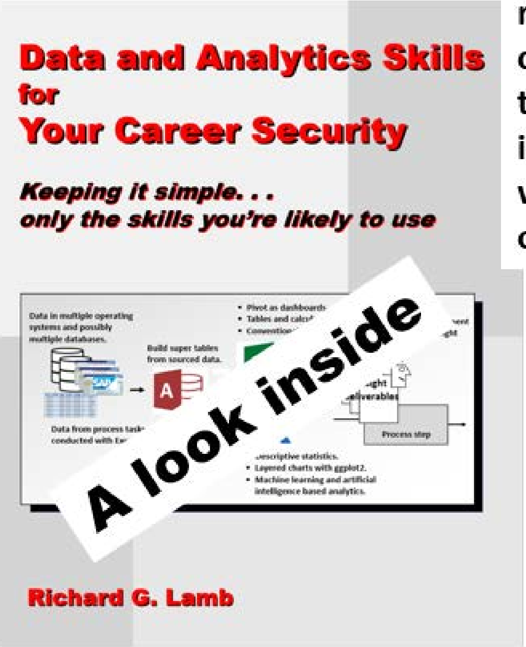


Figure 9-11: Tables, queries and pivot to analyze craft force maintenance capacity.

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.

At the mention of time series, we think of line charts. It is hard to imagine life without our many routinely distributed line charts. However, line charts are marginal in contrast with the analytics that are now so doable with the R software. This is because what we see as line charts have buried within them a trove of essential insights that are otherwise lost to us without time series analytics.



Data and Analytics Skills for Your Career Security

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

A look inside

Richard G. Lamb

Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Data from process tasks conducted with Ex
Pivot as dashboards
Tables and calc
Convention
Process step
Descriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

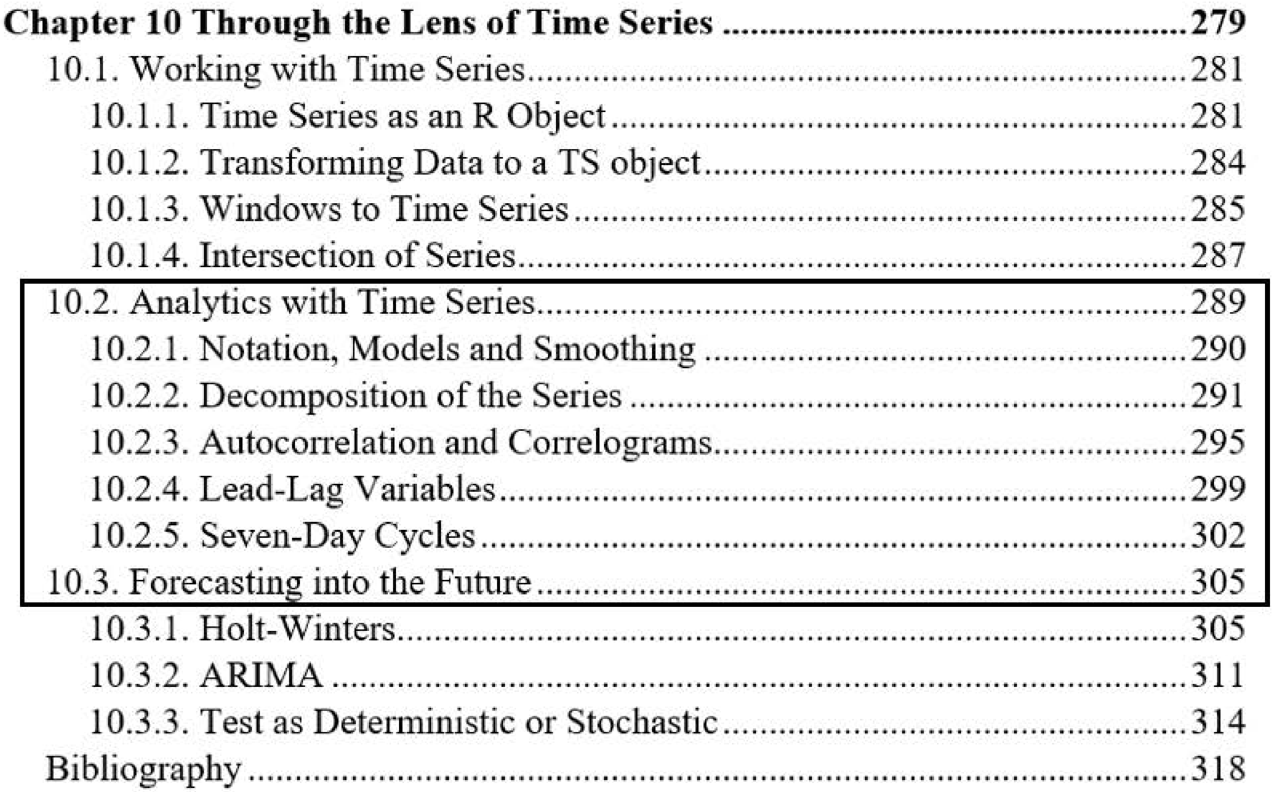
Chapter 10: Through the Lens of Time Series

Excerpts:

10.2. Analytics with Time Series

10.3. Forecasting into the Future

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside
Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds



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.

10.2. Analytics with Time Series

The previous section described times series as an R object just as are data frames, matrices, vectors and many others. The section described how to work with them. Now to shift gears to the analytic possibilities.

10.2.1. Notation, Models and Smoothing

We have defined a time series object and how to plot, slice-dice and combine it with others. However, that is the tip of the iceberg and we now need to move to fitting models to time series.

Accordingly, we need to set some notation. A time series is a series of length *n*. It is depicted as $\{x_t$: t = 1...n$\}$ = $x_1 \ldots x_n$. The notation is abbreviated as $\{x_t\}$ when it is not necessary to specify n. The time series model is regarded as a sequence of random variables of x_i.

The mean of the series, xbar, is the summation of the series divided by the number of observations, n. The "hat" in the notation represents prediction. It also represents a forecast for some time t + k where k is a number of periods from t.

There are two fundamental models to reflect whether or not the cycle is increasing. Additive and multiplicative, they are depicted respectively with the following equations.

Additive:

$$x_t = m_t + s_t + z_t \quad (10\text{-}1)$$

Multiplicative:

$$x_t = m_t * s_t + z_t \quad (10\text{-}2)$$

Where:

x_t = Observation at time t.

m_t = Trend at x_t.

s_t = Cycle or seasonal pattern at x_t.

z_t = Error at x_t.

There are ways to calculate m_t at time t. A simple method is with a moving average. The idea is to average the cycle effect centered on x_t.

There is a matter to be aware of for month and quarter as even numbered cycles in a fixed period. This is because an average speaks to time t with even number cycle periods to each side of t. For the year of months, we need six periods to both sides. For the quarters we need two. However, the problem is that we end up with one period entering the average twice and, for a year, x_t would be month 6.5.

The solution is equation 10-3. Notice that to create 12 periods in the cycle the end-most observations are given a weight of ½ (0.5).

$$\text{mhat}_t = (0.5x_{t-6} + x_{t-5} \ldots x_{t-1} + x_{t=1} \ldots x_{t=5} + 0.5x_{t+6})/12 \quad (10\text{-}3)$$

If we were working with a time series of quarters the same form would apply. The extreme terms would reflect as two rather than six.

We can extract the cycle effects for months or quarters with the following equations for the additive and multiplicative cases:

Additive:
$\mathrm{shat}_t = x_t - \mathrm{mhat}_t$ (10-4)
Multiplicative:
$\mathrm{shat}_t = x_t/\mathrm{mhat}_t$ (10-5)

Thence, we compute the predicted or forecasted x_t as xhat_t. How is shown in the following equations for the additive and multiplicative cases:

Additive:
$\mathrm{xhat}_t = \mathrm{mhat}_t + \mathrm{shat}_t$ (10-6)
Multiplicative:
$\mathrm{xhat}_t = \mathrm{mhat}_t*\mathrm{shat}_t$ (10-7)

In turn, the series error or random component of the series is the difference between xhat_t and x_t; equation 10-8:

$z_t = x_t - \mathrm{xhat}_t$ (10-8)

The term z_t is an estimate of the residuals to the series. This is because it is downstream from estimated trend and cycle.

10.2.2. Decomposition of the Series

We can use the decompose function to estimate the trend and cyclic effects in a time series. Its internals are built upon the moving average of Equation 10-3.

Our fundamental question of the observed series is what part is cycle and what part is trend. But first let's recall the definition of "trend" because the first thought to mind is a straight line through a series of points. Instead, it is the core underlying pattern to the overall series. As such, it is what remains after removing any cyclical patterns from the time series.

We now know there are two options with respect to the form by which cycle is related to the trend. Additive adds cycle to the trend. Multiplicative multiplies the trend by a factor.

Both were presented as notation and calculation in the previous section and are now demonstrated in the code below.

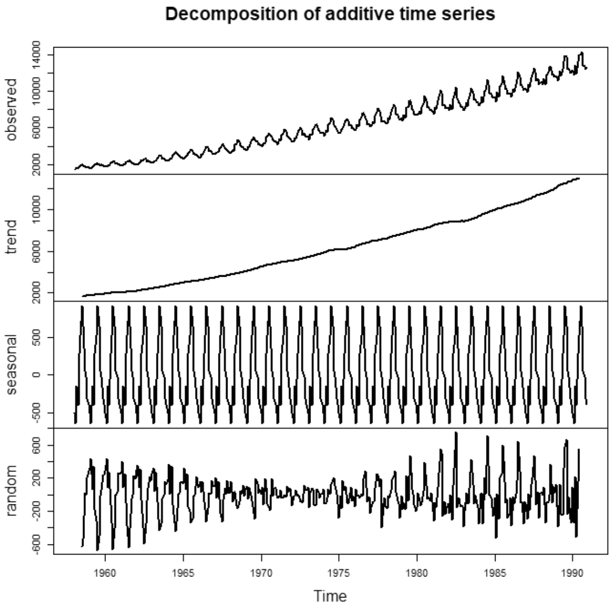
```
#For additive case
Elec.decom.add<- decompose(Elec.ts, type="add")
plot(Elec.decom.add)

#For multiplicative case
Elec.decom.mult<- decompose(Elec.ts, type="mult")
plot(Elec.decom.mult)

str(Elec.decom.add)
str(Elec.decom.mult)
```

In the first two blocks, we see the `decompose` function for each type. The function is fed the time series object to be dissected. The second argument selects additive or multiplicative, however, additive is the default that need not be coded.

Notice that what is returned by the function is assigned to a decomposed.ts object named `Elec.decom.add` and `Elect.decom.mult`. The next line of code returns the graphic of the `decompose` function, Output 10-7, for the additive case.

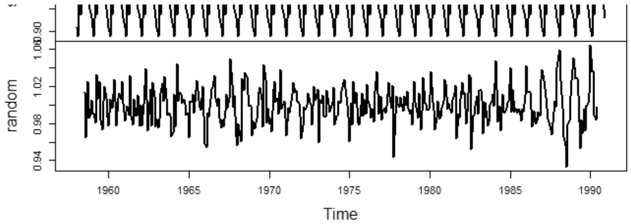


Output 10-7. Chart of decomposed electrical consumption.

The top panel charts the series ($x_{t...n}$) before decomposition. The next panel down is the trend to the series after removing cycles. Notice in the panel that the cycle is additive, swinging upon zero as its centering point.

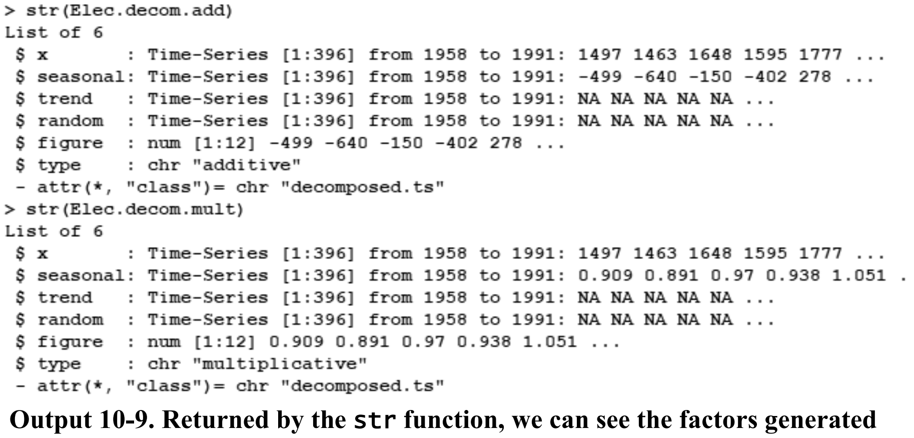
However, also notice the bottom panel. If the estimate of cycle were a good fit to the series, the plot would show the characteristics of randomness. But it does not because fixed adjustments for each period are being added to the trend even though the cycle is swinging wider with time.

The solution lies in the `type=` argument. We change it to "multi" for multiplicative. Now in Output 10-8 we can see the result in the random panel of the decomposition plot. Notice the residuals to the decomposition are generally random. Also, the seasonal panel will show decimal factors for cyclical adjustment rather than gravitation around zero.



Output 10-8. The residuals approach randomness with the multiplicative is the argument.

Notice the third block of code: `str(Elec.decom.mult)`. "Str" is reference to the structure of the object. When run, we get the output of Output 10-9. It allows us to determine what data is available to us by virtue of the decomposition. Each item preceded by "$" can be extracted and worked as a variable. This will be seen in action as the chapter unfolds.



```
> str(Elec.decom.add)
List of 6
 $ x       : Time-Series [1:396] from 1958 to 1991: 1497 1463 1648 1595 1777 ...
 $ seasonal: Time-Series [1:396] from 1958 to 1991: -499 -640 -150 -402 278 ...
 $ trend   : Time-Series [1:396] from 1958 to 1991: NA NA NA NA NA ...
 $ random  : Time-Series [1:396] from 1958 to 1991: NA NA NA NA NA ...
 $ figure  : num [1:12] -499 -640 -150 -402 278 ...
 $ type    : chr "additive"
 - attr(*, "class")= chr "decomposed.ts"
> str(Elec.decom.mult)
List of 6
 $ x       : Time-Series [1:396] from 1958 to 1991: 1497 1463 1648 1595 1777 ...
 $ seasonal: Time-Series [1:396] from 1958 to 1991: 0.909 0.891 0.97 0.938 1.051 .
 $ trend   : Time-Series [1:396] from 1958 to 1991: NA NA NA NA NA ...
 $ random  : Time-Series [1:396] from 1958 to 1991: NA NA NA NA NA ...
 $ figure  : num [1:12] 0.909 0.891 0.97 0.938 1.051 ...
 $ type    : chr "multiplicative"
 - attr(*, "class")= chr "decomposed.ts"
```

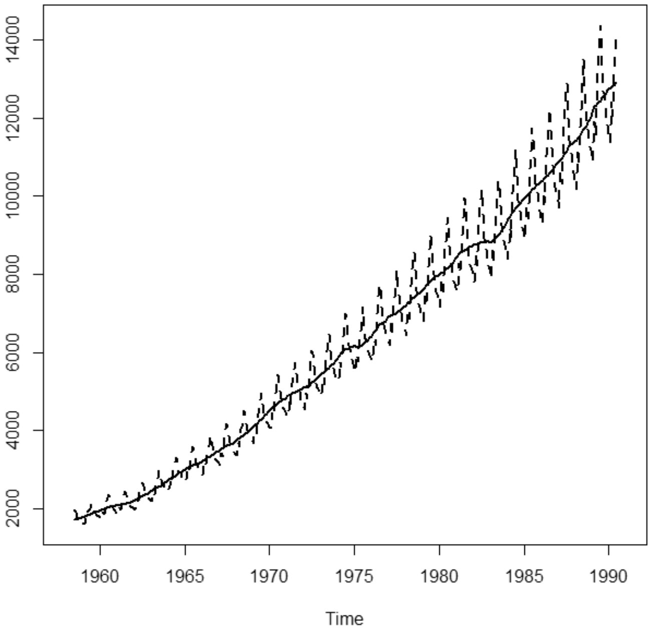
Output 10-9. Returned by the `str` function, we can see the factors generated by the decomposition function.

Plots typified in Output 10-7 show the components of the time series. What if we want to see the overall plot but with the trend distinguished from cycle in a layered chart? The following code will return such a perspective and is shown in Output 10-10.

```
#Trend distinguished from cycle
layout(1:1)
Trend<- Elec.decom.mult$trend
Seasonal<- Elec.decom.mult$seasonal
ts.plot(cbind(Trend, Trend*Seasonal), lty=1:2, lwd=2)
```

In the code, we see being put into play what is extracted from the `str` function. We are assigning the variables for trend and season from the object to be the variables Trend and Seasonal.

The final line of the block returns the layered chart shown in Output 10-10. The objects are combined as one with the `cbind` function. The `ts.plot` function returns the graphic with Seasonal overlaying Trend.



Output 10-10. The trend to electrical with trend and cycle as layers.

Notice the expression `Trend, Trend*Seasonal`. The first variable to the graph is the component trend. The second is the multiplicative cyclical variable times the trend. If our decompose.ts object were additive, the second variable would have been `Trend+Seasonal`.

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:

Variable1 = $x_i \ldots x_n$ and,
Variable2 = $x_{i-1} \ldots x_{n-1}$

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:

Variable1 = $x_i \ldots x_n$ and
Variable2 = $x_{i-2} \ldots x_{n-2}$

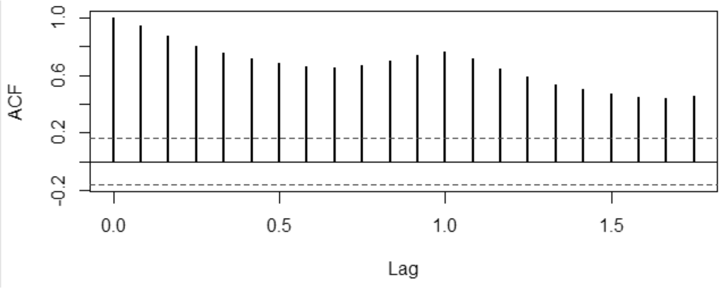
The correlation calculation is made for sets of two lag periods and then for all possible sets of lag.

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)
```

The second block of code subjects the observations of passengers to the autocorrelation function, `acf`. It presents as the correlogram in Output 10-11.



Output 10-11. Correlogram of the air passengers.

The x-axis gives the lag (k); each a month. The y-axis is the autocorrelation at each lag.

For the lag period, k = 0, autocorrelation is always 1. Its presence allows us to visually compare all other periods against a theoretical maximum.

The dotted lines at −1/n ± 1.96/sqrt(n) represent the boundaries for a 95 percent confidence that there is or is not autocorrelation. Stated statistically, they are the boundary at which the null hypotheses is that there is no autocorrelation. If the vertical line extends beyond, there is autocorrelation.

However, we must be cautious in our interpretation. We are interpreting multiple hypothesis tests. If correlation is zero at all lags, we can still expect 5 percent of the tests along the chart to fall outside the boundaries.

Furthermore, we must remain cognizant that correlations at the lags are autocorrelated. If one test shows significant, the adjacent tests are also likely to be statistically significant.

We should also be cautious of what constitutes a practicable autocorrelation. Squaring the autocorrelation at a lag gives guidance. It indicates the linear dependency of x_t and x_{t-1}. If, for example, the test shows 0.2, then it explains only 0.2^2 = 0.04 of the variability in x_t; not much.

Rather than test the time series as observations, we typically test the random component of its dissection. However, this case of the time series is illustrative. We can see the depiction of an increasing series and seasonal cycle. The increase is reflected in the slow overall decline. The cycle is reflected by the wave shape for which valley to valley is 12 months.

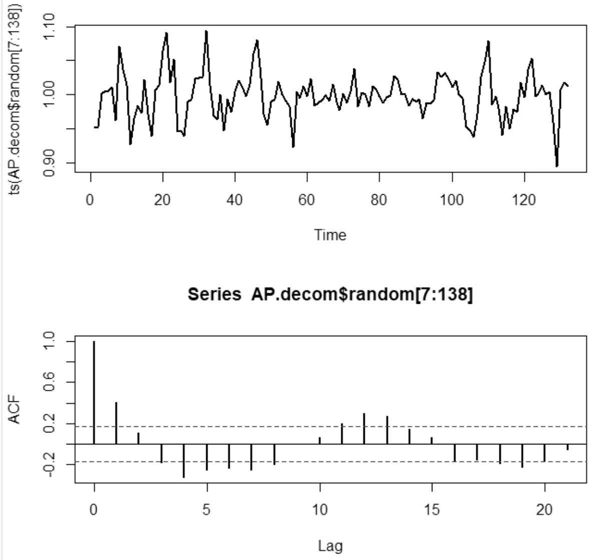
The code below will evaluate the random component. The first block creates the decompose.ts object. The `length` function informs us that there are 144 periods to the series object, a necessary piece of information as we set up the subsequent block of code.

```
#Decompose AP and get count of periods
AP.decom<- decompose(AP, "multiplicative")
length(AP.decom$trend)

#Plot the random component and correlogram
layout(1:2)
plot(ts(AP.decom$random[7:138]), lwd=2)
acf(AP.decom$random[7:138], lwd=2)
```

The second block of code creates two plots. One is the random component of the series. The other is the correlogram. The returned graphics are shown in Output 10-12.

Notice in square brackets the code `7:138`. Recall that the decompose function uses a moving average centered on x_t. Consequently, for this case of 12 months, the first and last six months cannot be calculated. By calling the length function, we remind ourselves that there are 144 periods and, thus, the range of usable data points starts with the 7th month and ends with the 138th month.



Output 10-12. Random component of the passenger series and its correlogram.

The output shows that there is autocorrelation at the first lag period. We could subject the `acf` line of the code to the `str` function if we want the finding as a number. Readers would find the first period to be 0.403.

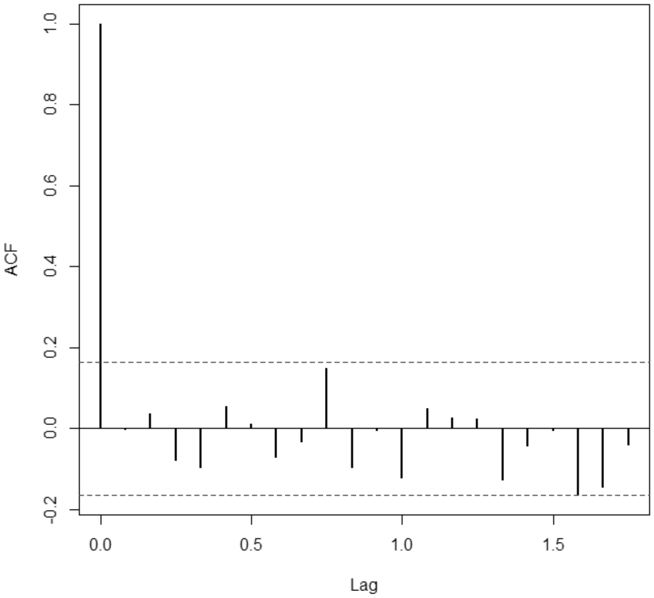
We can also see in the correlogram that the fit of the decompose function does not completely deal with the seasonal component. However, if we applied the standard deviation function, `sd`, to the series of observations and to the trend component of decomposition we would find a considerable reduction in variance.

Let's step in here to demonstrate the ARIMA model for dealing with the seasonal component of air passenger data. As a point of reference, it is one of the most complex of models. The acronym stands for autoregressive interactive moving average. Because of its complex theory and mathematics, here and later in the chapter, the ARIMA model will only be shown and explained in action rather than to full technical depth.

The code below will call up the correlogram shown in Output 10-13.

```
#Correlogram with ARIMA
d.arima <- auto.arima(AirPassengers)
acf(d.arima$residuals, lwd=2)
```

The first line of the code calls up the ARIMA model which requires that the `forecast` package has been loaded into the R session. Fitting the model to the data is done by trial and error for seven parameters. Rather than involve ourselves in a manual process, the `auto.arima` function optimize the integrated fit of the seven parameters. Output 10-13 shows the `acf` function applied to the residuals of the returned model.



Output 10-13. Correlogram of residuals to air passengers using the ARIMA model.

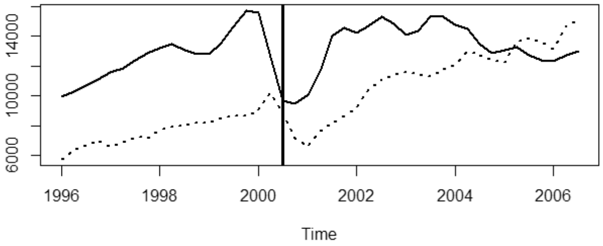
Let's mention why we care. Two fundamental characteristics of a variable is its location (mean) and its spread (variance). If there is positive autocorrelation our dataset, spread will be reported back as a tighter variation than true and the opposite for negative autocorrelation. At times, spread looms 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 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.

$$\text{Index} = (\text{Min} + \text{Observation})/(\text{Max} - \text{Min}) \qquad (10\text{-}9)$$

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.

```
##Extract random components for approve and activity
#Approvals
app.ran<- decompose(App.ts)$random
app.ran.ts<- window(app.ran, start=c(1996,3), end=c(2006,1))
#Activity
act.ran<- decompose(Act.ts)$random
act.ran.ts<- window(act.ran, start=c(1996,3), end=c(2006,1))
```

In the code we do a decomposition of App.ts and Act.ts and then extract the random component for each. Notice that we start at month 3 and end 2 months short of the last period of each series. Recall that the moving average is upon quarters, thus, we must remove the first and final two observations.

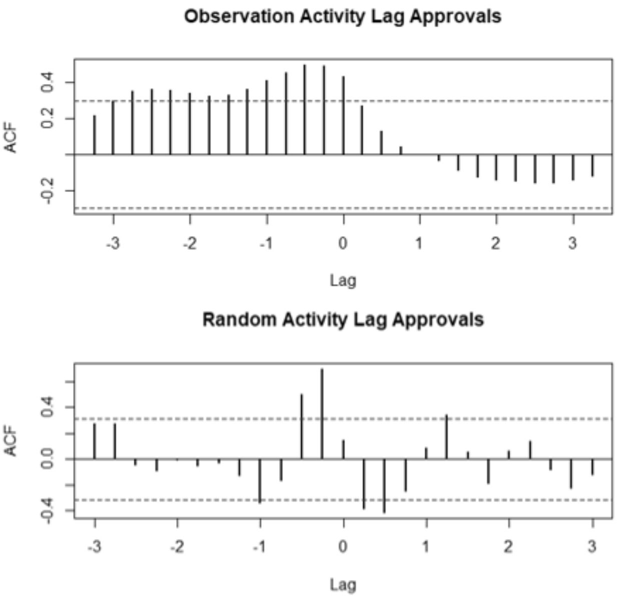
Let's explain what we are about to look at. Autocorrelation was explained in the previous section. Now we shift to cross correlation. The concept is the same except that for one series we are computing correlation to the second series by number of lags.

The code below will conduct the analysis at two levels. The first will demonstrate application upon the observations and why we should not be quick to hang our hats on the apparent conclusion. The second applies the analysis to the random component of the series for a more truthful expectation.

```
#Plot cross correlograms for observations and random
#Cross correlation of observations
layout(1:2)
ccf(App.ts, Act.ts, main="Observation Activity Lag Approvals")
#Cross correlogram of random component
ccf(app.ran.ts, act.ran.ts, main="Random Activity Lag Approvals")
```

The insight and contrast are shown in Output 10-15. The inspection of negative lag in the upper panel indicates that we can foresee activity as far in advance as years. We can expect the strongest indicator to be two quarters in advance.

In the lower panel of Output 10-15 we see a tighter story based on the random component. The negative as lag shows the strongest relationship is at the first quarter. The tell is fading by the second quarter. We can see that cycles after the first quarter quickly lose their indicative power.



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.

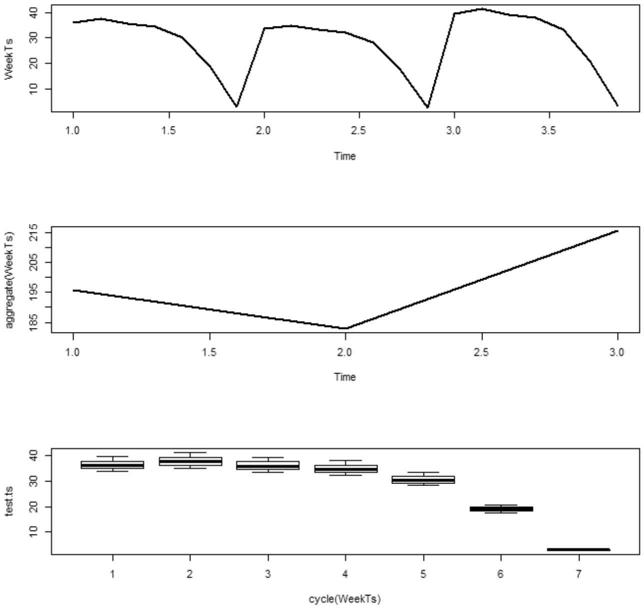
```
#Transform series variable to ts object
WeekTs<- ts(SevenDay$Series, start=c(1,1), freq=7)
```

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))
```

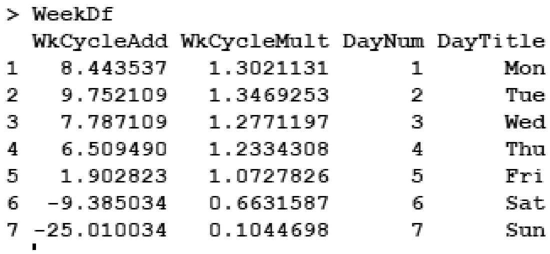
In the top panel of the figure are the daily observations for the three weeks. If for example, the variable was number of requests for new work, there is a pattern. The middle panel shows the total (aggregate) new requests for each week.

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.

More specifically we may want a table in Excel so that other analysts can use the findings in their pondering and scheduling. Output 10-17 is an example of what we may seek.



```
> WeekDf
  WkCycleAdd WkCycleMult DayNum DayTitle
1   8.443537   1.3021131      1      Mon
2   9.752109   1.3469253      2      Tue
3   7.787109   1.2771197      3      Wed
4   6.509490   1.2334308      4      Thu
5   1.902823   1.0727826      5      Fri
6  -9.385034   0.6631587      6      Sat
7 -25.010034   0.1044698      7      Sun
```

Output 10-17. Weekly pattern for additive and multiplicative cases.

The table as a data frame object can be exported to be an Excel file with the `write.xlsx` function. The code below will develop the data frame shown in the output.

```
#Decompose weeks as trend: Additive
DecomPredAdd<- decompose(WeekTs, type="add")
plot(DecomPredAdd, lwd=2)

#Decompose weeks as trend: Multiplicative
DecomPredMult<- decompose(WeekTs, type="mult")
plot(DecomPredMult, lwd=2)

#Extract seasonal component from respective decompositions
WkCycleAdd<- as.numeric(window(DecomPredAdd$seasonal, start=1,
   end=(1+6/7)))
WkCycleMult<- as.numeric(window(DecomPredMult$seasonal, start=1,
   end=(1+6/7)))

#Form data frame of additive and multiplicative
WeekDf<- data.frame(WkCycleAdd, WkCycleMult, DayNum = c(1:7),
   DayTitle = c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"))
WeekDf
```

The first two blocks of code decompose the week for the additive and multiplicative cases. Both return the perspectives that are normal to decomposition. They will not be shown here because they are typical to earlier sections and the reader can generate them if desired.

The third block of code extracts the seven-day pattern. Notice the `window` function in action to slice out a precise segment of the overall series. It commands that we want the first seven days; day six over seven days defines the seven data points in decimal format.

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.

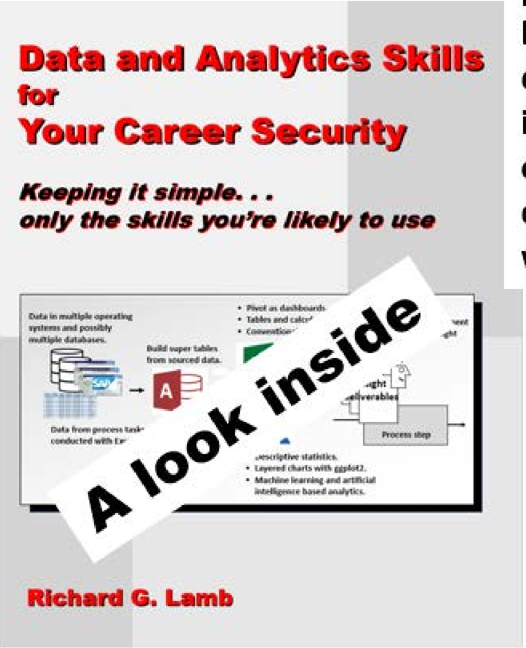
The final block of code creates the data frame we want and was previewed in Output 10-17. It combines the factorial patterns of both decompositions and gives a day number and name to each factor.

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



Data and Analytics Skills for Your Career Security

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

A look inside

Richard G. Lamb

Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Pivot as dashboards
Tables and calc
Convention
deliverables
Process step
Data from process task conducted with
Descriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

The design of a data-driven operational process precedes the choice of data and analytics methods. But process design will reflect our knowledge of data and analytics. A process is recharted at the intersection of process activity and knowledge. The chapter demonstrates the top-down charting of a data-driven operational process, influenced by what we know is possible with MS Access.

Chapter 11: Budget-Based Schedule and Craft Capacity

Excerpts:

11.1. Scheduling Framed Top-Down

11.2. Budget-Based Schedule Cycle

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside
Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds

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.

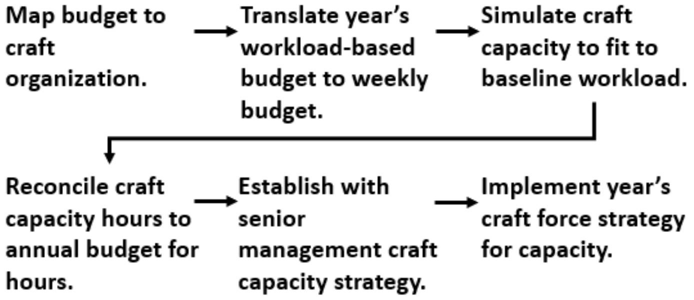

Map budget to craft organization.
Translate year's workload-based budget to weekly budget.
Simulate craft capacity to fit to baseline workload.
Reconcile craft capacity hours to annual budget for hours.
Establish with senior management craft capacity strategy.
Implement year's craft force strategy for capacity.

Figure 11-1: The activities to set weekly budget for sustaining workload and craft capacity to do the work.

Map budget groups to the craft organization. The annual budget has likely not been formed with regard to craft organization. It is structured by budget groups. In some plants they coincide. However, they are typically not an exact match.

Accordingly, the first step is to map the budget groups to schedule groups. We are grouping the workload with respect to the craft groups that are being scheduled to do the work. The craft groups upon which the weekly schedule should be organized are often already in practice and we continue with them.

Translate the annual workload-based budget to a baseline weekly budget. The weekly schedule is a budget as much as it is an action plan. Now to reduce the annual budget to the weekly workload to each schedule group. It is the baseline or starting point for every week's initially proposed schedule. Rather than load ready jobs into the schedule in accordance with available craft hours, the weekly schedule is loaded in accordance with the baseline.

We have presumed that the monthly budget is the annual workload smoothed for each budget group. However, for some plants, average may vary by season to reflect industry, market, weather, etc. For them, the baseline may be set on seasonal cycles.

Simulate craft capacity to fit the baseline workload. The next challenge is to size craft capacity as headcount to deliver the baseline workload. Unfortunately, it is not as easy as dividing total craft-type hours to the baseline weekly workload by standard working hours per week.

We need to simulate the craft body for each self-contained craft schedule group in the maintenance organization. With applied statistics it can be done in sophisticated ways. Fortunately, it can also be done with a more straightforward approach.

The practical method is to build a collection of representative weekly schedules. The scheduler can reach back and select representative weekly schedules. Thence, the scheduler can expand or trim the schedules to match the baseline schedule.

The next step is to slice the weekly schedule into day schedules of generally smoothed workload. To the collective days, the scheduler will fit, by trial and error, a craft force of types and headcounts.

Most scheduling software such as MS Project and Primavera make excellent simulators. For the jobs to a representative series of weekly schedules and standard workdays, schedule software can draw and assign from craft pools the individual crafts that are called for by job plan. If the next job requires already engaged crafts, the task is not started until crafts become available.

What we seek is to arrive at the craft pools that allow all work to happen no later than the end of the schedule week. The scheduler will adjust the pools by trial and error to find the craft counts that work.

It is best that the scheduler size the force on a collection of representative schedules. An alternative is to schedule for multiple weeks in series to a month or longer. In both methods, this will test the pools against varying circumstances. The more representative schedules, the tighter the fit in the face of uncertainty.

The scheduler will also want to in some way adjust for noise. This is because job plans should not be written to include the random unplannable. Accordingly, the supervisory role holders in the work execution stage should collaborate with the scheduler to devise an adjustment for noise in the craft pool.

Reconcile craft capacity to the annual budget. The sizing of craft capacity arrives at the hours to the sustaining workload from two directions. The annual budget is formed on the history of occurring work and average actual or estimated hours per order. The weekly schedule ties hours to orders according to job plan as the best estimate of hours. To that is made an allowance for noise.

At this step we should reconcile the craft capacity hours to budget hours. If there is significant imbalance we should learn where and why.

Collaborate with plant management in decision-making. The level in an organization at which decisions are made reflects how widely the organization is affected. The decision-making for maintenance budget and craft body have far ranging implications for the enterprise and are, thus, senior management's decisions to make. The activities to build the budget and size craft headcount is decision-analysis upon which senior management makes its final call.

Their decisions may seem to be decision-making for maintenance. The reality is that the decisions being made are directly relevant to the firm's market share and net margin. That is why it is not the maintenance operation's place to make budget and capacity decisions. However, it is maintenance management's place to not allow senior management to make their decision without the preceding workload-based analyses to guide them.

Therefore, the decision-analysis and decision-making process should be collaborative. Upon the collaboration, senior management will ultimately make decisions for the enterprise as a whole. If they do not adopt the full-scale version of decision-analyses, they will want to explore the alternative backward from ramifications to the enterprise.

Because it is a substantial business expense, management may decline to retain the fully sustaining craft body, thus, elect to not fulfill the sustaining workload. Rather than 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.

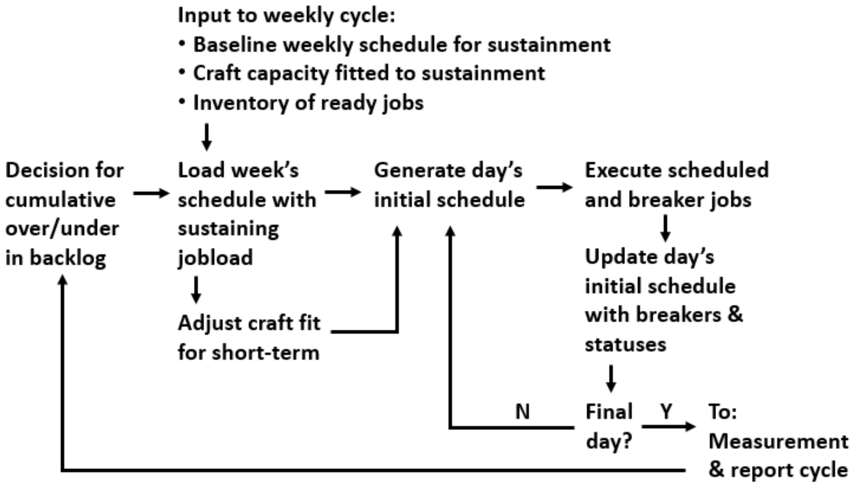

Input to weekly cycle:
• Baseline weekly schedule for sustainment
• Craft capacity fitted to sustainment
• Inventory of ready jobs
Decision for cumulative over/under in backlog
Load week's schedule with sustaining jobload
Adjust craft fit for short-term
Generate day's initial schedule
Execute scheduled and breaker jobs
Update day's initial schedule with breakers & statuses
N
Final day?
Y
To: Measurement & report cycle

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.

Adjust Craft fit for the short-term. The baseline workloads and craft capacities can never be prescient. We hope that more then 90 percent of the time the craft body covers the workload with a spread in variance that is normally reasonable.

Accordingly, an auxiliary task of the scheduling process is short-term craft resourcing. There is a host of strategies and tactics to flexibly stay on beam. The point is that the craft capacity may be a bit dynamic but must ultimately keep the plant on beam.

Generate day's initial schedule. The next step is repetitive throughout the week. An initial schedule is set for the day, usually as a standard activity of the previous day. The mixture and counts to schedule groups are observed, but the overarching goal is to achieve the match on a weekly basis with some over and under from day to day.

The day's schedule can have breakers in progress from the previous day or breakers recognized in a previous day but delayed to the current day. Otherwise, the remainder of the day's 100 percent loading will be taken from the remaining balance of the week's scheduled work.

Execute day's scheduled and breaker jobs. Want to hear a belly laugh, tell God our plan for the day. At this step, we set out to execute the day's "list" of work to be done. We can expect that some of the initially scheduled work on list of work may be displaced by breakers.

At this step, the objective is to execute a fixed sustaining mixture and count of jobs. Therefore, we do not fanatically resist unforeseen breakers. Of course, if frequently excessive we would want to know why.

Update the day's initial schedule for breakers and statuses. It is realistic to expect that occasionally, or more frequently, the day's scheduled jobs will be displaced by breakers. Accordingly, at the end of the day we will want to record the breakers to the day's actual outcome. The jobs that were scheduled but not started, will remain on the updated schedule because the update also serves as the record of the day.

Also updated are the status variables to all jobs. Breakers to the day's schedule should be classified as something other than scheduled. The same for planned and unplanned jobs. Finally, the jobs to the day are classified as not started, started and completed.

Final day of the week? At the end of the day there will be a fork in the road depending upon whether the day is the last of the week. If not, the next day's schedule will reflect all that has happened as the week has unfolded. If the final day, the next steps reside in the measurement and report stage of the overall cycle.

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.

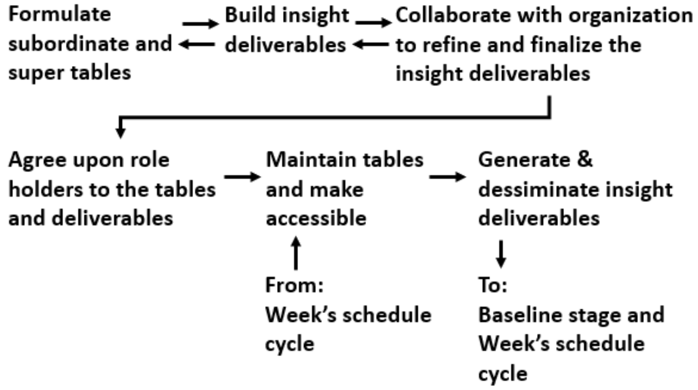

Formulate subordinate and super tables
Build insight deliverables
Collaborate with organization to refine and finalize the insight deliverables
Agree upon role holders to the tables and deliverables
Maintain tables and make accessible
Generate & dessiminate insight deliverables
From: Week's schedule cycle
To: Baseline stage and Week's schedule cycle

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.

The philosophy is to build subordinate and super tables without concern for “cost” of data. This is because, for a maintenance operation, there is no incremental cost for more data unless manually collected. Another reason is that we do not want to limit the operation to the insight it has thought to create. Another reason is that operational data has a universal entirety to it. Rather than think in terms of envisioned insight deliverables, we think in terms of how to place our data in tables of every imaginable variable to the operation.

Build insight deliverables. The developer knowledgeable in maintenance planning and scheduling will build the insight deliverables. Section 11.4 defines and scopes the insight deliverables that will emerge from this step.

Collaborate with the organization to expand, refine and finalize insight deliverables. Some tasks require that an individual engage herself in quiet, thoughtful problem-solving. That was the case for the step to build the insight deliverables.

In contrast, this step will take the creations of the previous two tasks to the organization of role holders and decision-makers. There will be a creative collaboration between the parties.

As such we can expect that the insight deliverables will be expanded, enhanced and refined. We can also expect that the organization will send the developer back to flesh out and build new ideas.

The collaboration has another effect. As the collaborators converge upon common thinking and perception of possibilities, they will tend to broadcast the possibilities outward across the organization. For example, role holders in other processes may be made aware of super tables that they have always wished for but long ago given up on.

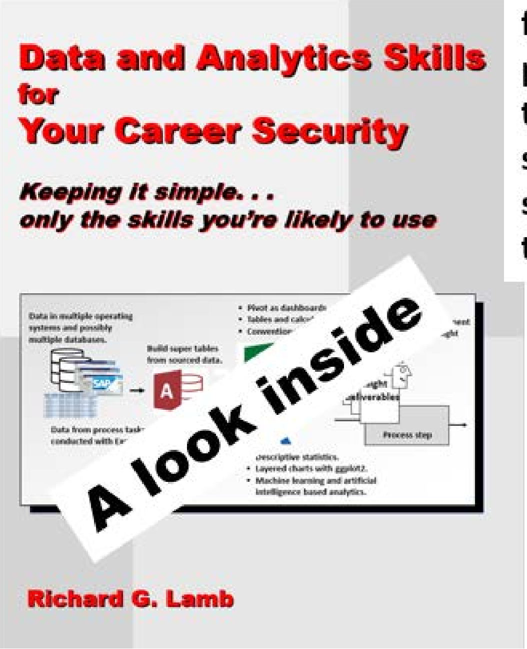
Agree upon role holders to the tables and insight deliverables. It can be expected that the tables and insight deliverables, once built and made operational, can be administered by clerical and analyst roles. This step is to establish who will take on the tasks to routinely administer the tables and insight deliverables.

In some cases, a role holder to a normal operational task should administer the tables and insights as part of their task. In parallel, their administrative task is on behalf of all role holders across the organization that engage the tables and insights in their own work.

Maintain tables and make them accessible. The task is a routine one and can be assigned to anyone for administration. At daily and weekly milestones, the collection of designed tables is updated and returned to accessibility. As seen in Figure 11-3, the input to the administrative tasks are the outcomes of the day’s preparation and execution.

Generate and disseminate insight deliverables. Finally, the insight deliverables are updated and disseminated. Dissemination may be no more than updating the deliverables 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.



Data and Analytics Skills for Your Career Security

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

A look inside

Richard G. Lamb

Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Pivot as dashboards
Tables and calc
Convention
SAP
A
Data from process task conducted with Ex
Process step
deliverables
Descriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

Two essential measure of any operation are what is the probability that specific conditions will hold for some length of time and then what is the probability the condition will end? Additionally, through what operational variables can we significantly change the answers? With the R software, the chapter demonstrates how to answer the questions with retention-exit analytics.

Chapter 12: Elapsed Time Through Stages

Excerpts:

12.1. The Insight We Need

12.2. How It Works

12.3.3. Parametric Modeling

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside

Book is available from Amazon

Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds

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 reactiveness 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.

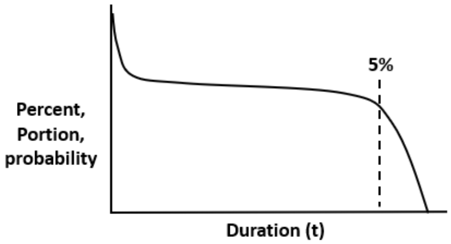


Figure 12-1: General pattern of retention as in a pipeline of work.

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.

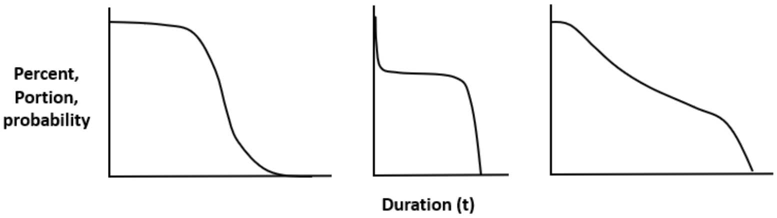


Figure 12-2: Stages to a pipeline of operational activities.

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.

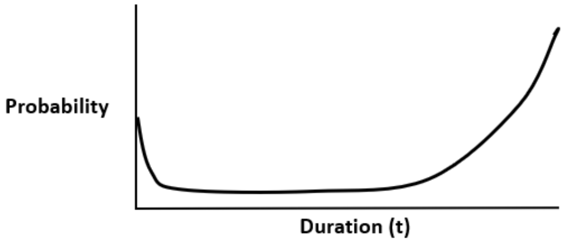


Figure 12-3: Instantaneous probability of the specified exit event.

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.”

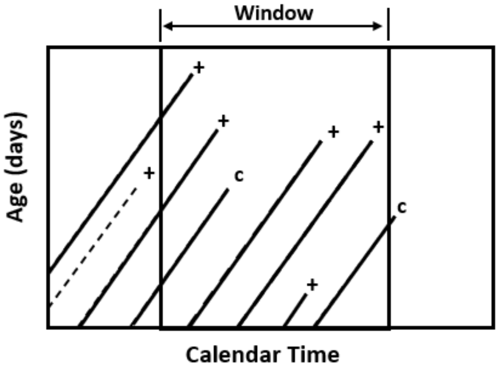


Figure 12-4: Cases as a specified or censored event during the window of the study.

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.

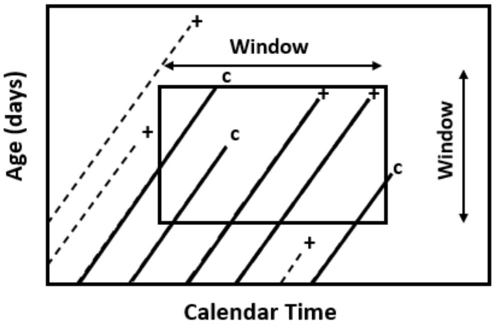


Figure 12-5: Window of a study with age as a dimension.

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-Aelen estimator of probability of a specified exit event (hazard) and the Kaplan-Meir 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.

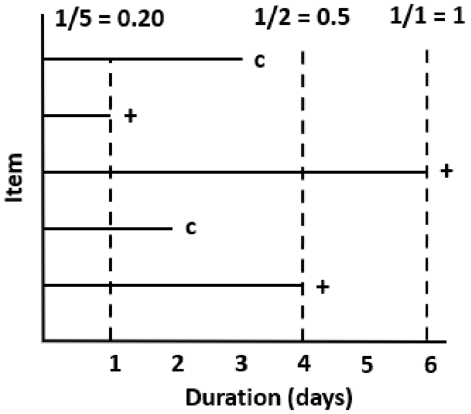


Figure 12-6: Probabilities computed upon specified exit events.

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.

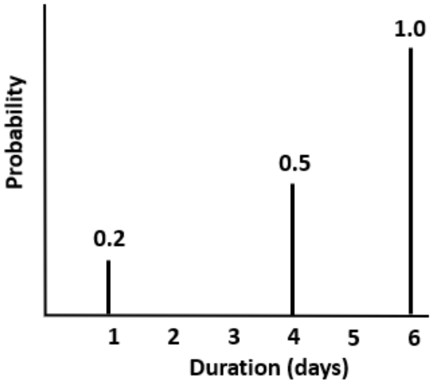


Figure 12-7: Plot of probability for specified exit events.

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.

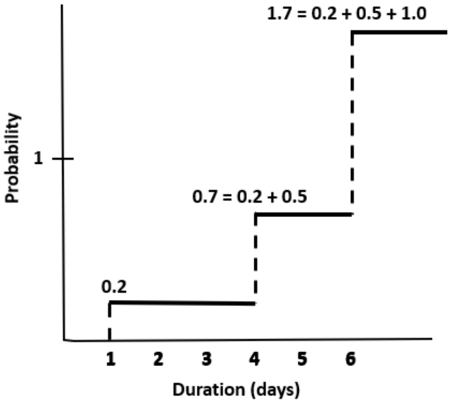


Figure 12-8: Cumulative hazard with time for an exit event.

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.

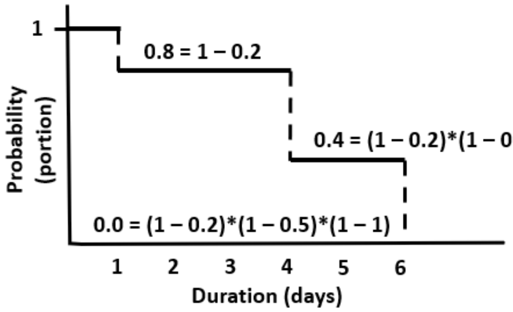


Figure 12-9: Computation of the portion of cases remaining in the stage at day in duration.

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.

12.3.3. 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.

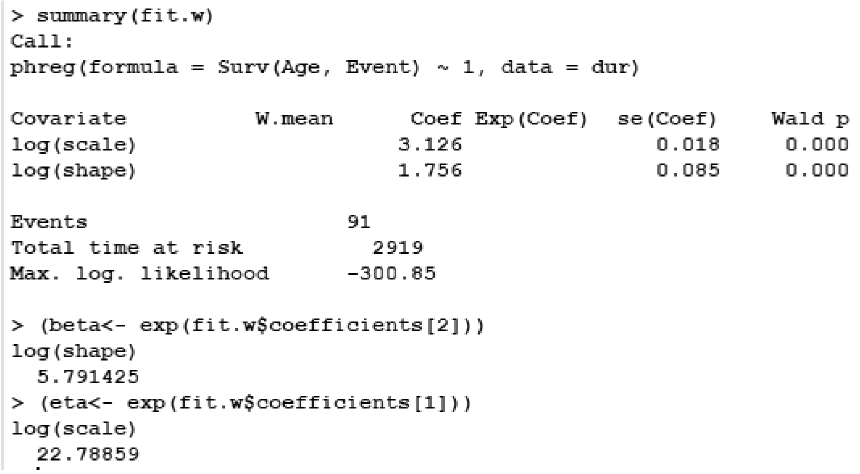
```
#PARAMETRIC FIT FOR BETA AND ETA
fit.w<- phreg(Surv(Age, Event)~ 1, data=dur)
summary(fit.w)
(beta<- exp(fit.w$coefficients[2]))
(eta<- exp(fit.w$coefficients[1]))
#PLOT UPON PARAMETERS
plot(fit.w, lwd=2)
```

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 "coefficients." There are two classifications to the variable. The first is eta and the second is beta.



```
> summary(fit.w)
Call:
phreg(formula = Surv(Age, Event) ~ 1, data = dur)

Covariate          W.mean      Coef Exp(Coef)  se(Coef)    Wald p
log(scale)                    3.126             0.018     0.000
log(shape)                    1.756             0.085     0.000

Events                   91
Total time at risk         2919
Max. log. likelihood      -300.85

> (beta<- exp(fit.w$coefficients[2]))
log(shape)
  5.791425
> (eta<- exp(fit.w$coefficients[1]))
log(scale)
  22.78859
```

Output 12-4: Model fitted to Weibull and the parameters extracted from it.

Recall that square brackets are used to filter vectors and table by row and column. Within the brackets of the code, we specify "2" as the second item in the two item vector for beta and likewise "1" for eta.

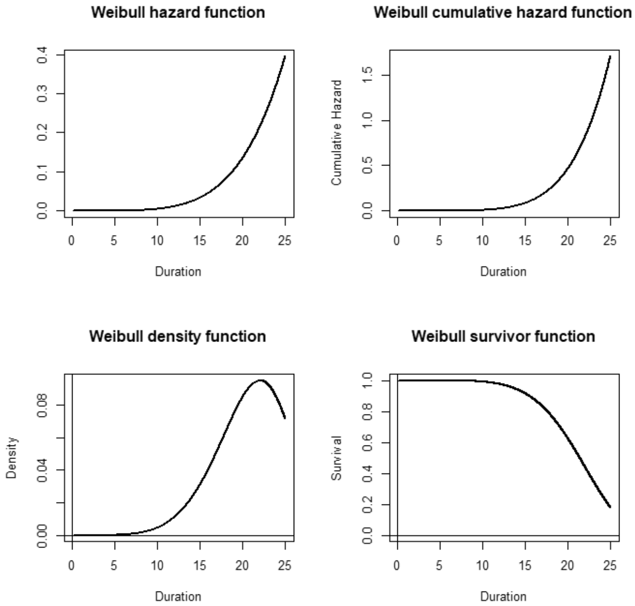
The final line of the block of code will return a chart view for the survival-hazard (retention-event) functions: hazard (exit), cumulative hazard, survival (retention) and Weibull density. Shown in Output 12-5, they present a picture as if the data were continuous rather than the previous discrete plot.

There is a drawback of relying solely on empirical or parametric fits. Notice that the expectation for several early existing cases is not visible in the continuous fit of Output 12-5. They were in the empirical plot of Output 12-3. For this reason, we should run both if we chose to run the fitted model.

The payback of the parametric model, of course, is the returned parameters. With them, our analysis of the retention and exit behavior can be ascertained by the equations of Section 12.2.3.

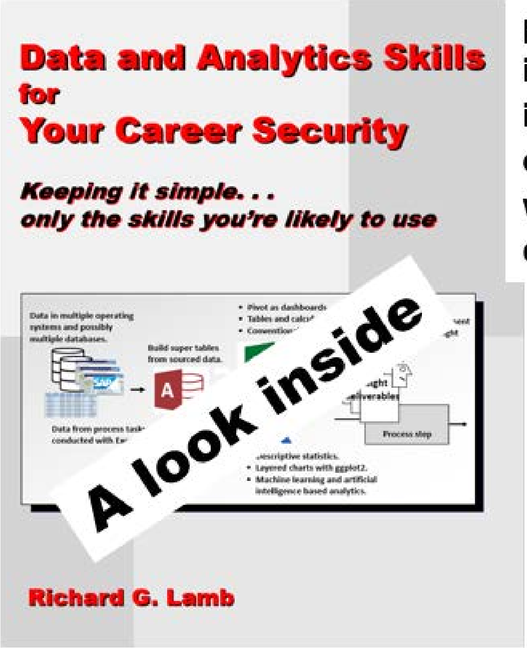
Suppose we do need to get a stronger sense of eras along the overall retention plot. Maybe there are considerable early exit events. Maybe there is a noticeable decline rather than plateau but much less than before the point in retention at which there is accelerating exit.

In the scenario, there are three eras to the Weibull distribution. Unfortunately, the Weibull can only fit one at a time from the family of curves. The solution is to subset the data by age and study each as an era. We would use the empirical plots to delineate the eras.



Output 12-5: The four functions of survival hazard (retention-event) shown as continuous plots.

Section 12.2.1 explained the two dimensions to a study window. The principle would now come into play for partitions upon age that are the three eras of retention.



Data and Analytics Skills
for
Your Career Security

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

A look inside

Richard G. Lamb

Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Data from process task conducted with Ex
Pivot as dashboards
Tables and calc
Convention
ght deliverables
Process step
escriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

It is a simple question. Is there a difference in performance from operational alternatives and improvements? In statistical speak, we are asking if the comparative means in the consequent outcomes are significantly different. The chapter will explain by demonstration how to answer the question.

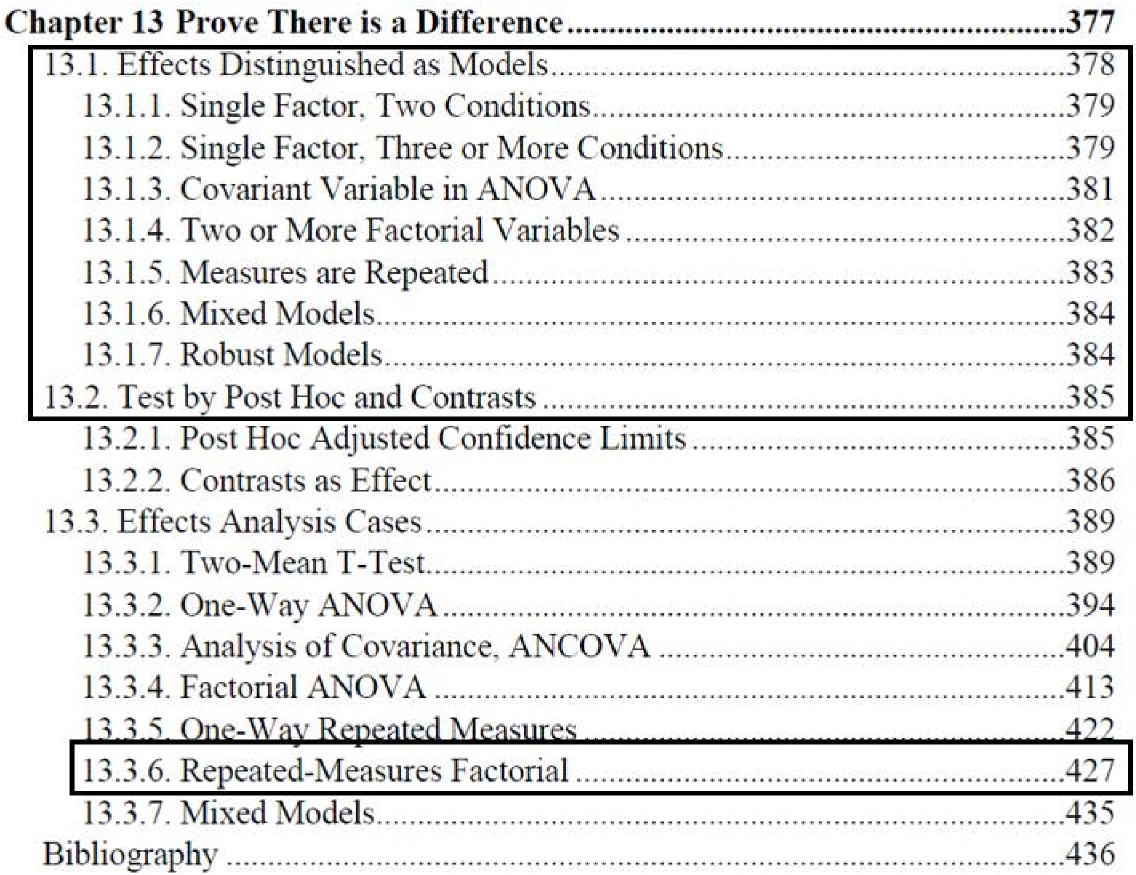
Chapter 13: Prove There is a Difference

Excerpts:

13.1. Effects Distinguished as Models

13.3.4. Factorial ANOVA

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside
Book is available from Amazon
Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds



Chapter 13 Prove There is a Difference 377

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.

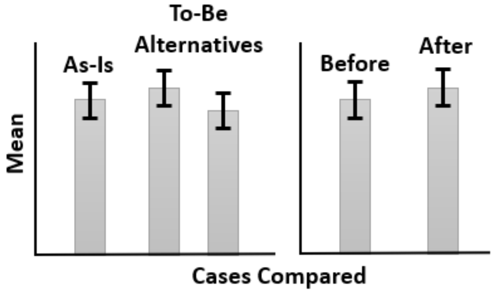


Figure 13-1: We must determine if the mean to each differ from others.

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.

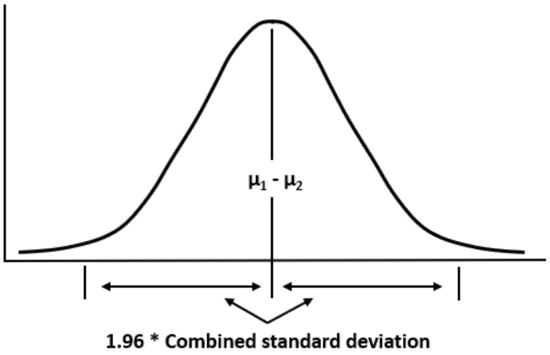


Figure 13-2: T-test for two means.

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.

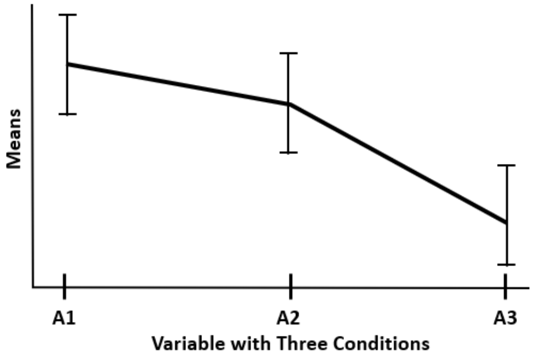


Figure 13-3: Means and confidence intervals to three conditions.

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.

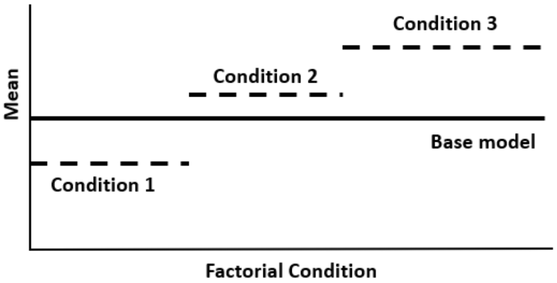


Figure 13-4: The concept of ANOVA as conditions.

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 covariant, 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. The 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.

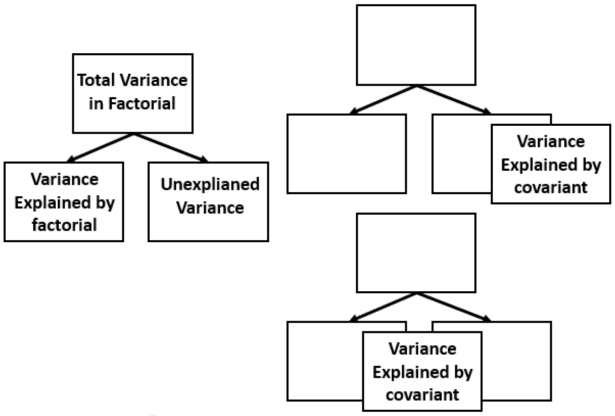

Total Variance in Factorial
Variance Explained by factorial
Unexplianed Variance
Variance Explained by covariant
Variance Explained by covariant

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.

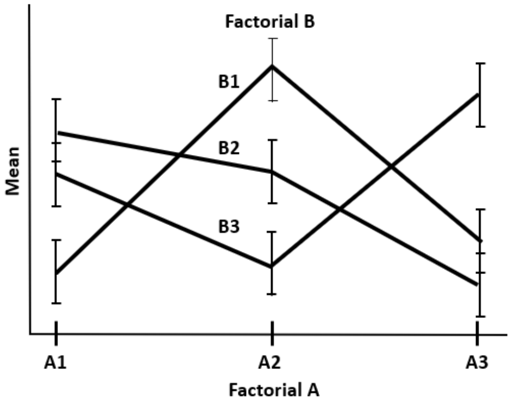


Figure 13-6: Two factorial variables with three conditions to each.

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.

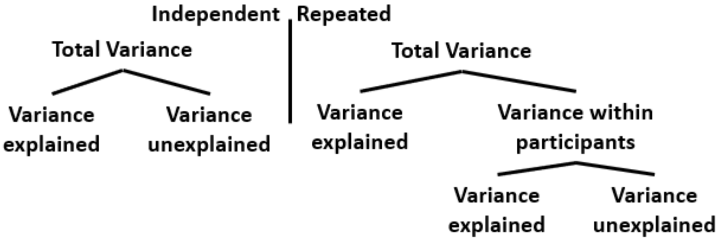


Figure 13-7: Comparison between independent and repeated measures models.

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.

13.3.4. Factorial ANOVA

We are not limited to evaluating conditions to a single factorial variable. We can model two or more factorial variables. This section will demonstrate how with two variables. From what was earlier called one-way ANOVA, we are now stepping out to what is called factorial ANOVA. We could, but do not, include one or more covariates.

This section will demonstrate the method of independent designs. Section 13.3.6 will demonstrate the method for repeated measures.

The code to follow demonstrates the method while extending the explanation of Section 13.1.4. As before, this section will explain the process but not always demonstrate what is repetitious to previous cases.

Just as before, the case will imagine that we are comparing two new schemes to the standing operational procedure and to each other. The conditions, as schemes, to the variable, Strat, are NoChng, DesignA and DesignB. This time we are adding a second factorial variable, Grade, with conditions of Junior and Senior. We seek to understand the degree that they effect the variable, Score, which is some index of plant performance.

The dataset is in the previously identified Excel file as the worksheet, tblBestFactorial. The loaded packages are ggplot2, pastecs, car, multicomp, and xlsx.

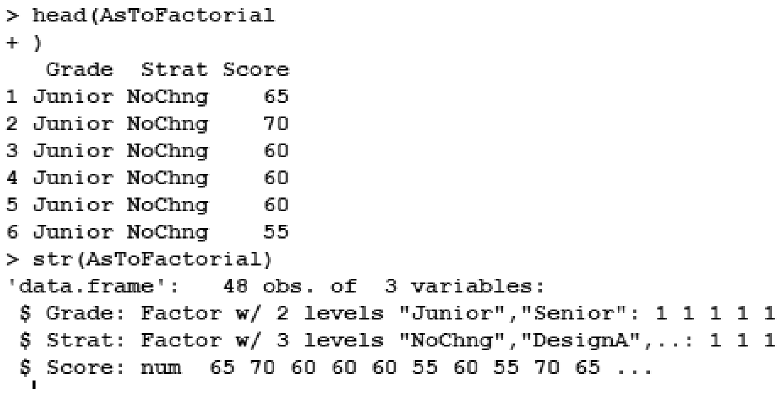
Beginning with the first block of code, we read the dataset into the session. The second block of code is necessary because the alphabetical order of the conditions to the Strat variable do not make NoChng the base condition. We do not need to take the same ordering action for the grade variable because the order of conditions we want are alphabetical.

```
#DOWNLOAD DATA
OrigAsToFactorial<-
   read.xlsx("C:\\<PATH>\\SkillsForCareerSecurity_Datasets.xlsx",
   sheetName="tblBestFactorial", header=TRUE)
AsToFactorial<- OrigAsToFactorial
head(AsToFactorial; str(AsToFactorial)

#SET LEVELS TO STRAT VARIABLE
AsToFactorial$Strat<- factor(AsToFactorial$Strat,
   levels=c("NoChng","DesignA","DesignB"))
str(AsToFactorial$Strat)
```

With the `head` and `str` functions, we can inspect the dataset as shown in Output 13-26. Notice the predictor variables, Grade and Strat, are factor variables with two and three conditions, respectively. The output variable, Score, is numeric. There are 48 observations in the dataset.

As before, we explore the dataset and, as before, we presume the analyst has already explored the dataset to the standard demonstrated in Chapter 5.



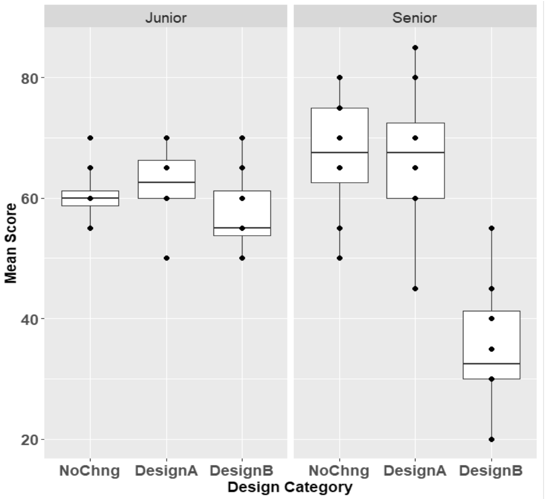
```
> head(AsToFactorial
+ )
   Grade  Strat Score
1 Junior NoChng    65
2 Junior NoChng    70
3 Junior NoChng    60
4 Junior NoChng    60
5 Junior NoChng    60
6 Junior NoChng    55
> str(AsToFactorial)
'data.frame':   48 obs. of  3 variables:
 $ Grade: Factor w/ 2 levels "Junior","Senior": 1 1 1 1 1
 $ Strat: Factor w/ 3 levels "NoChng","DesignA",..: 1 1 1
 $ Score: num  65 70 60 60 60 55 60 55 70 65 ...
```

Output 13-26: Head and str view of the dataset.

We should begin with a sense of relationship of the two factorial variables to score and each other. The code below will produce the boxplot insight shown in Output 13-27.

```
#EXPLORE DATA
#GRAPHIC
FactBox<- ggplot(AsToFactorial, aes(Strat, Score))
FactBox + geom_boxplot() + facet_wrap(~Grade) +  geom_point(size = 2) +
   labs(x = "Design Category", y = "Mean Score") +
   theme(plot.title=element_text(size=14,face="bold"),
        axis.text=element_text(size=14,face="bold"),
        axis.title=element_text(size=14,face="bold"),
        strip.text = element_text(size=14))
```

Output 13-27 shows that a pattern seems to persist across the design variables. For both grades, the median score for DesignA is close or above NoChng and DesignB below. The scores for NoChng and DesignA are greater for Senior over Junior. Interestingly, the median scores for DesignB flip the pattern. The relative gaps between DesignA and DesignB with respect to grade is also noteworthy.



Output 13-27: Boxplot of Design to Score, subset by Grade.

With the following code we also inspect the data with descriptive statistics. The three uses of the `by` function present the descriptive insight to the two variables and their permutations. The first two command lines of the code are as we have done in the previous models. The third is new to us.

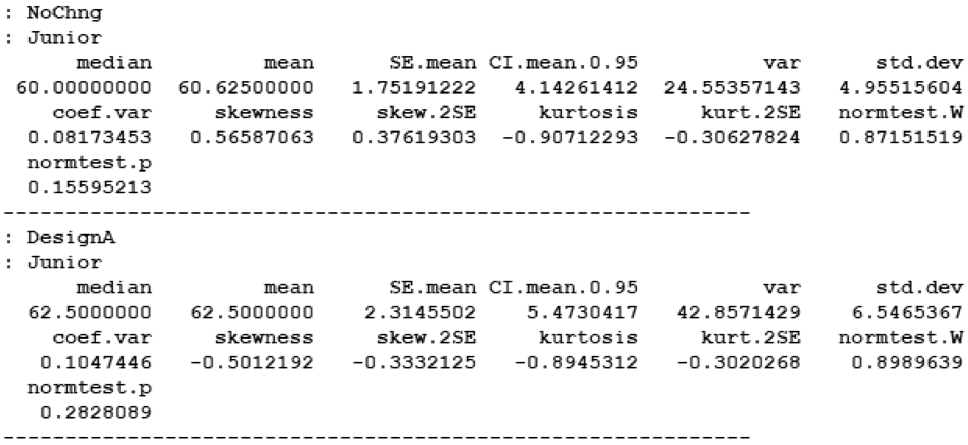
```
#DISCRIPTIVE
#TABLE INDIVIDUAL FACTORIALS
by(AsToFactorial$Score, AsToFactorial$Strat, stat.desc, basic=FALSE,
   norm=TRUE)
by(AsToFactorial$Score, AsToFactorial$Grade, stat.desc, basic=FALSE,
   norm=TRUE)
#TABLE OF FACTORIALS AS INTERACTIVE
by(AsToFactorial$Score, list(AsToFactorial$Strat, AsToFactorial$Grade),
   stat.desc, basic=FALSE, norm=TRUE)
```

If run, the first `by` command would reveal that scores to the conditions of the Strat variable have normal distributions. However, the variance for DesignB is far different than the other two conditions.

If run, the second `by` command would find that the distribution of score for craft is not normal for the Junior craft. We would also notice a large difference in variance between Junior and Senior conditions.

We will run the third `by` command because it is new to the chapter. It shows the factorial variables as permutations of the Strat and Craft variables. An excerpt of two of the six possible interactions is shown in Output 13-28. The reader should pull up and inspect the entire table.

An inspection of the excerpt finds the same statistics as before. The difference is that each is one of the six possible combinations. As we inspect the entire table, we would be especially interested in variance and normal distribution.



```
: NoChng
: Junior
      median         mean      SE.mean CI.mean.0.95          var      std.dev
 60.00000000  60.62500000   1.75191222   4.14261412  24.55357143   4.95515604
    coef.var     skewness     skew.2SE     kurtosis     kurt.2SE   normtest.W
  0.08173453   0.56587063   0.37619303  -0.90712293  -0.30627824   0.87151519
  normtest.p
  0.15595213
------------------------------------------------------------
: DesignA
: Junior
      median         mean      SE.mean CI.mean.0.95          var      std.dev
  62.5000000   62.5000000    2.3145502    5.4730417   42.8571429    6.5465367
    coef.var     skewness     skew.2SE     kurtosis     kurt.2SE   normtest.W
   0.1047446   -0.5012192   -0.3332125   -0.8945312   -0.3020268    0.8989639
  normtest.p
   0.2828089
------------------------------------------------------------
```

Output 13-28: Example of two of six combinations to interactive descriptive statistics.

With the block of code, we were able to inspect for normal distribution to each condition as standalone and as an interaction. However, we must also test for homogeneity of variance. The next block of code does that for the Strat and Grade variables and their interactions.

```
#LEVENE TEST OF HOMOGENIOUS VARIANCE
#TEST IF VARIANCE BETWEEN SCORE AND EACH FACTORICAL VARIABLE
leveneTest(AsToFactorial$Score, AsToFactorial$Strat, center=median)
leveneTest(AsToFactorial$Score, AsToFactorial$Grade, center=median)
#TEST OF VARIANCE ACROSS ALL SIX GROUPS
leveneTest(AsToFactorial$Score,
   interaction(AsToFactorial$Strat, AsToFactorial$Grade), center=median)
```

The reader would find that the code reports that the Strat variable passes the test of homogeneity with a p-value of 0.1095. The Craft variable fails the test with a p-value of 5.08e-05. For the interaction of variables, the p-value of 0.2351, reports there is homogeneity among permutations.

Just as for one-way ANOVA, we build contrasts of our choosing. The only difference is that we have more than one variable for which we must build sets of contrasts. The code is shown below. It is left to the reader to inspect them with `AsToFactorial$Strat` and `AsToFactorial$Grade` commands..

```
#CHOOSING CONTRASTS
#CONSTRASTS FOR DESIGN
contrasts(AsToFactorial$Strat)<- cbind(c(-2,1,1),
   c(0,-1,1))
AsToFactorial$Strat
#CONTRASTS FOR GRADE
contrasts(AsToFactorial$Grade)<- c(-1,1)
AsToFactorial$Grade
```

The contrasts for Strat are the same as before. By the assignment of negative weights in the respective contrasts, NoChng and DesignA are the base conditions.

The contrast for Grade, as two conditions, simply compares the two. By virtue of assigning a negative weight, Junior is the base condition to Craft.

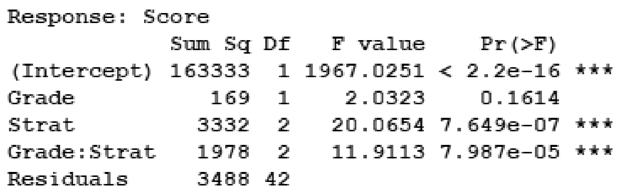
Now to fit an ANOVA model to the factorial variables. The following code does that. The meat of its output is shown in Output 13-29.

```
#FITTING THE FACTORIAL ANOVA MODEL
FactorialModel<- aov(Score~Grade+Strat+Grade:Strat, data=AsToFactorial)
#Alternative code
#FactorialModel.Alt<- aov(Score~Grade*Strat, data=AsToFactorial)
#Run the model omnibus ANOVA
Anova(FactorialModel, type="III")
```

Notice that the model equation, `Score ~ Grade + Strat + Grade:Strat`, is coded full out for the model, `FactorialModel`. The alternative is to write the model as the shorthand variable1*variable2 and so on, depending upon the number of variables to be inspected for interaction.

In Output 13-29, we see the output of the factorial ANOVA. We see the same structure as for one-way ANOVA but regarding more than a single term. The p-value is determined by the ratio of explained to unexplained variance for each variable and the interaction of the variables. More specifically, the F ratio and p-value are built on the ratio of Residuals each of the three.

The interpretation of the model lies in the interaction of variable, Grade:Strat. The p-value less than or equal 0.05 reveals that there is interaction. As explained in Chapter 7, the rule is that when this is the case, the p-value to the "main" effects, Design and Grade, should not be interpreted.

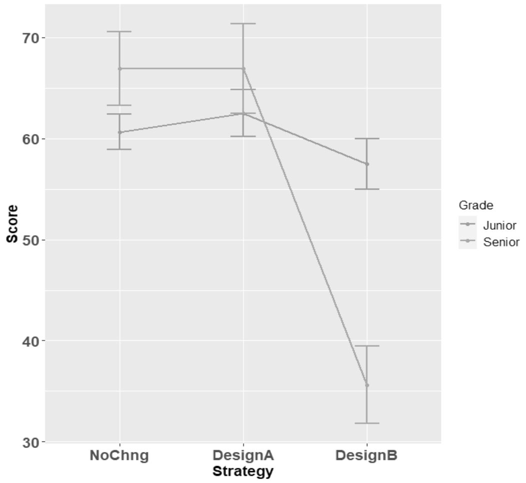


```
Response: Score
            Sum Sq Df   F value    Pr(>F)    
(Intercept) 163333  1 1967.0251 < 2.2e-16 ***
Grade          169  1    2.0323    0.1614    
Strat         3332  2   20.0654 7.649e-07 ***
Grade:Strat   1978  2   11.9113 7.987e-05 ***
Residuals     3488 42                        
```

Output 13-29: The ANOVA model shows there are significant differences in effects.

We can plot the means and confidence limits of the factorials such that the interactions are visible. Such a plot is constructed with the following code and shown in Output 13-30.

```
#GRAPHIC OF FINDINGS BY MODEL
InterLine<- ggplot(AsToFactorial, aes(Strat, Score, color = Grade))
InterLine + stat_summary(fun = mean, geom = "point") +
    stat_summary(fun = mean, geom = "line",
        aes(group= Grade), lwd=1) +
    stat_summary(fun.Grade = mean_cl_boot,
        geom = "errorbar", width = 0.2, lwd=1) +
    labs(x = "Strategy", y= "Score", colour= "Grade") +
    theme(plot.title=element_text(size=14,face="bold"),
        axis.text=element_text(size=14,face="bold"),
        axis.title=element_text(size=14,face="bold"),
        legend.title = element_text(size = 12),
        legend.text = element_text(size = 12))
```



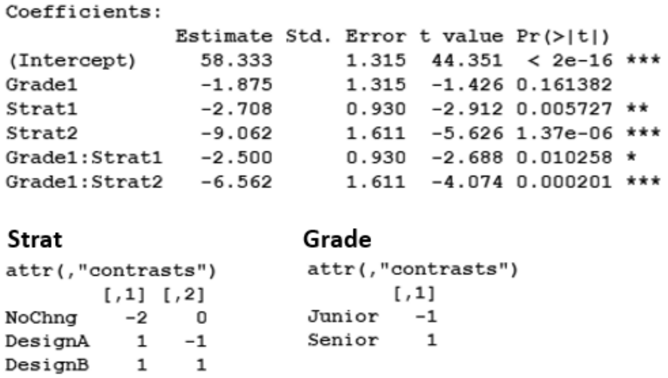
Output 13-30: Plot showing interactions in the ANOVA.

In the returned chart, we can clearly see an interaction when the conditions of the factorial variables are not parallel. If the readers inspect the charted means and confidence limits, they will find that they are constructed with the descriptive statistics, means and confidence limits, of Output 13-28.

Now to ferret out significant effects by interpreting the contrasts. We begin with the code below. The meat of the output is shown in the upper part of Output 13-31. The lower portion is inserted for reference while explaining the model's output.

```
##INTERPRETING CONTRASTS FOR DIFFERENCE
summary.lm(FactorialModel)
```

The output presents the p-values to the main and interactive effects of the contrasts. As already noted, only the two interaction terms can be interpreted. The general rule is that any main effect term or contrast involved in a significant interaction term must not be interpreted.



```
Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)      58.333      1.315  44.351  < 2e-16 ***
Grade1           -1.875      1.315  -1.426 0.161382    
Strat1           -2.708      0.930  -2.912 0.005727 ** 
Strat2           -9.062      1.611  -5.626 1.37e-06 ***
Grade1:Strat1    -2.500      0.930  -2.688 0.010258 *  
Grade1:Strat2    -6.562      1.611  -4.074 0.000201 ***
```

```
Strat                     Grade
attr(,"contrasts")        attr(,"contrasts")
        [,1] [,2]                [,1]
NoChng    -2    0         Junior   -1
DesignA    1   -1         Senior    1
DesignB    1    1
```

Output 13-31: Output of the contrasts to the model.

Just as for the previous two demonstrations, the terms to the linear model are the contrasts rather than the dataset variables.

The contrasts, Grade1, Strat1 and Strat2, are interpreted in the same manner as explained in Section 13.3.2. However, because the interaction terms are showing significance, we should not draw conclusions from them.

Both interaction terms are significant. The significance is telling us that there is interaction between the contrasts of the respective factorial variables.

The interaction, Grade1:Strat1, is telling us that the difference in the NoChng and the combined DesignA and DesignB varies depending on the condition of the Grade variable. The expression can also be stated in reverse logic.

The interaction, Grade1:Strat2, is telling us the difference between the DesignA and DesignB varies in magnitude upon the condition of Grade. The expression can be stated in reverse logic.

By overlay on Output 13-30, the interactions are visible. For the Grade1:Strat1 interactions, average the means of DesignA and DesignB for the two Grade lines. For Grade1:Strat2, the contrast is the output with the NoChng condition removed.

Let's visualize a step up in complexity. Imagine if Grade was a factorial of three conditions. Three additional contrasts, Grade2, Grade2:Strat1 and Grade2:Strat2, would appear in the upper portion of the output. However, we would interpret them in the same manner.

Now imagine additional factorial variables. Visual interpretations quickly get out of hand. When we get a little past the complexity demonstrated, we must rely on models to interpret the main and interaction effects. Models set us free to pursue searches into complex comparisons and effects.

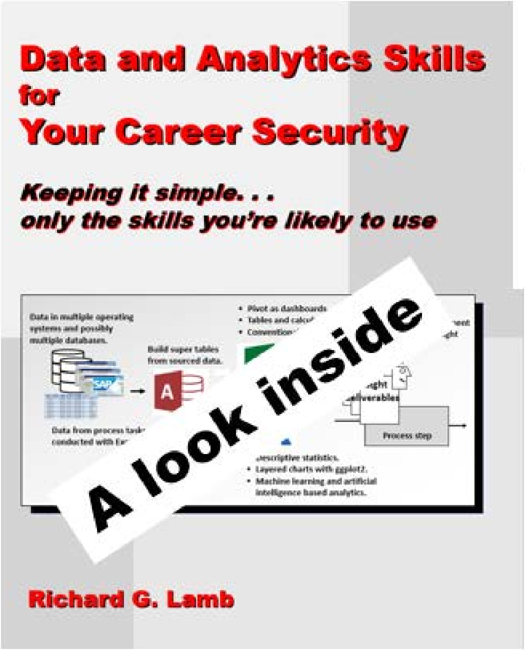
For the one-way ANOVA, we also drew upon post hoc tests to find pairs that were significantly different. However, the method is impossible for factorial ANOVA.

If we wanted to do a post hoc analysis, we would create a new factorial variable as the permutation of all conditions. In the dataset, we would concatenate the conditions to each observation as a new variable e.g., CraftDesign. Thence, we would conduct the one-way ANOVA with the CraftDesign variable. The Grade and Strat variables would not be included in the model.

As for the one-way ANOVA and ANCOVA, to validate the model, we would generate the visual of residuals with the `plot` function below. The reader will find that the plot shows, for variance, the residuals take on a blatant funnel shape. They are heteroscedastic. The other plot shows the residuals are shown by the Q-Q plot to be somewhat, but loosely, normally distributed.

```
#PLOTS IN FACTORIAL ANOVA
par(mfrow=c(2,2))
plot(FactorialModel)
par(mfrow=c(1,1))
```

The finding of the plots imply that we should conduct a robust factorial ANOVA. How is presented by Section 12.7 of Field.



Data and Analytics Skills for Your Career Security

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

Richard G. Lamb

A look inside
Data in multiple operating systems and possibly multiple databases.
Build super tables from sourced data.
Pivot as dashboards
Tables and calc
Data from process task conducted with Ex
Process step
Descriptive statistics.
Layered charts with ggplot2.
Machine learning and artificial intelligence based analytics.

Inherent to operational data are the classifications captured in its management systems. If we comply with our operational procedures, the consequent quality of all classifications is almost unavoidably perfect. The cold reality is that compliance is problematic for classifications that are not automatic. The chapter presents the two favorite, most practical methods to recover classifications: logistic regression and naïve Bayes.

Chapter 14: Recover Lost Classifications

Excerpts:

14.1. Generalized Procedure

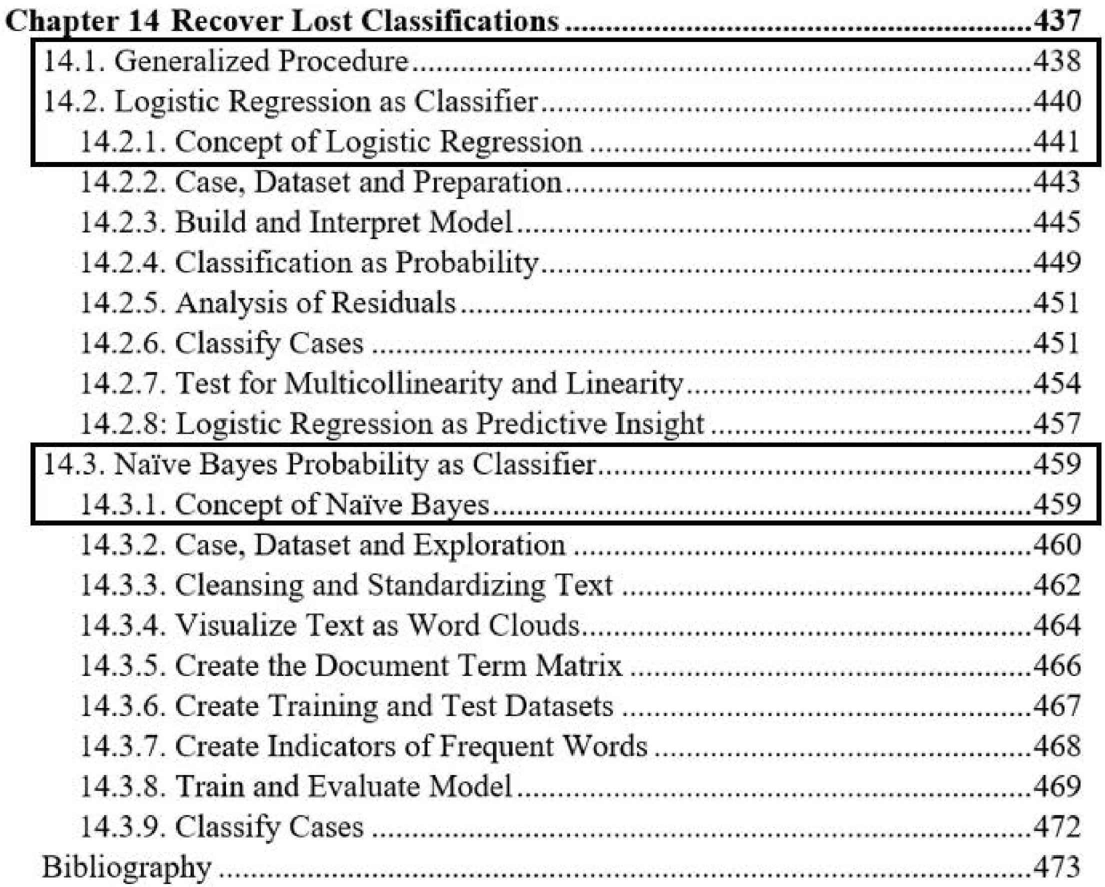
14.2. Logistic Regression as Classifier

14.3. Naïve Bayes Probability as Classifier

Additional "Look Inside" at https://analytics4strategy.com/book-look-inside

Book is available from Amazon

Paperback, 508 pages, 300 figures and outputs, 7.5 x 1.15 x 9.25 inches, 2.37 pounds



Chapter 14 Recover Lost Classifications

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 the 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.

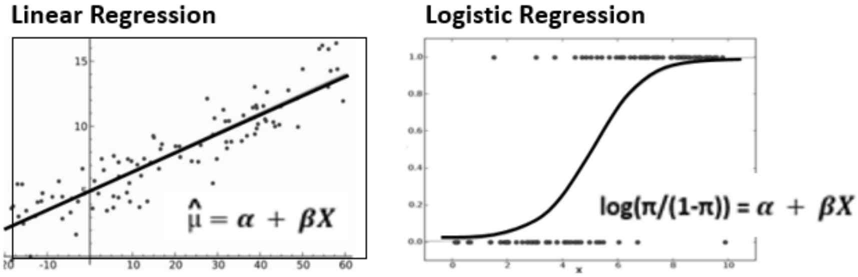


Figure 14-1: Comparative perspectives of linear and logistic regression.

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: û. For logistic regression it is called the log odds: log(π/(1-π)). The systematic components to the right of the equality operator are identical.

The linking function relates the systematic component to the right of the equality to the solution element of the left û for linear and π for logistic. Figure 14-2 presents the contrasting solution equations.

Linear Regression	Logistic Regression
$\hat{\mu} = \alpha + \beta X$	$\pi = \dfrac{e^{(\alpha + \beta X)}}{1 + e^{(\alpha + \beta X)}}$

Figure 14-2: Solution equations to linear and logistic regression.

The solution equation for linear regression is straightforward. We are solving a simple identity. In contrast, for the logit we are solving for probability, π, the vertical axis to the logistic regression curve in Figure 14-1.

What is shown in the figure is the binary logistic. It is actually a special case of the multinomial logistic. Because the reader will most likely use the binary over the multinomial logistic the chapter will not demonstrate the multinomial. Instead, the reader is referred to Chapter 7, Section 7.9 of the book, Discovering Statistics with R, by Field for an entertaining demonstration. There are also good demonstrations with R available on the internet. The explanation of the binary variation will prepare the reader to work with the multinomial variation.

The concept for binary and multinomial logistic is the same. Imagine that we seek to classify three failure modes rather than two. It is required that the modes be mutually exclusive and completely exhaustive. Figure 14-3 shows the solution formula as a set of equations, one for each mode.

$$\pi_1 = \frac{e^{(\alpha_1 + \beta_1 X)}}{1 + e^{(\alpha_1 + \beta_1 X)} + e^{(\alpha_2 + \beta_2 X)}}$$

$$\pi_2 = \frac{e^{(\alpha_2 + \beta_2 X)}}{1 + e^{(\alpha_1 + \beta_1 X)} + e^{(\alpha_2 + \beta_2 X)}}$$

$$\pi_3 = \frac{1}{1 + e^{(\alpha_1 + \beta_1 X)} + e^{(\alpha_2 + \beta_2 X)}}$$

Figure 14-3: Solution equations to a classifier of three classifications.

Notice that each of the three solution equations has the same form as we saw for the solution equation for binary logistic regression in Figure 14-2. The differences are the intercept and coefficient: α and β. They reference to an associated classification.

A careful inspection will spot how multinomial works. Notice the numerator to each solution equation. A systematic component has been written for two of the three classifications. The denominator is same to all three. Consequently, for a case, the formulas are computing a probability for each classification. Because they are multinomial, in sum they total to 100 percent.

Chapter 7, Section 7.1.1 for linear regression, explained the systematic component as almost limitless in makeup. They can be combinations of main and interactive effects, numeric and factorial variables, and polynomial terms. The full range of variables holds for logistic regression. The makeup of the systematic component is so unlimited that the big issue is what to put in, what to leave out. The overarching process of selection is much the same as explained in Chapter 7.

The interpretation is also largely the same. Except that the effects of the coefficient are additive for linear regression and multiplicative for logistic regression, all else follows the interpretation as explained in Chapter 7 for linear regression.

Earlier chapters mentioned or demonstrated the multilevel regression. Until now it has surfaced as mentioned for linear regression and utilized for repeated measures ANOVA. Multilevel modeling is also possible for binary and multinomial logistic regression. If the reader feels that the classifications are somehow sensitive to hierarchy, she is directed to the book, Multilevel Modeling Using R, by Finch, Bolin and Kelly.

There is also the option of what is call "ordinate" logistic regression. It can be used if the classifications are ordinately related, e.g., low, medium and high. If that is the case the reader is advised to seek examples on the internet. Ordinal logistic regressions tends to be more accurate than binary and multinomial.

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.

Index

Data and Analytics/Operational Effectiveness

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

Analytics4Strategy

ISBN 978-1-7343947-0-2

90000>

9 781734 394702