In [1]:
```
# Binary Outcome Definition
#| Value | Meaning
#|   1   | RAC occurred — career-defining moment, HR involved, legend status achieved
#|   0   | Email sent correctly — uneventful, forgettable, nobody's Friday ruined

#200 observed email incidents. The author has seen things.
#| Feature | Description |
#| `Thread_Length` | Number of prior replies in the chain (1-80). Beyond 47, survival is a mi
#| `Confidential_Flag` | Whether "Confidential" appears in the subject line. Correlated with 
#| `Hour_of_Day` | Hour of sending (8-18). Friday afternoons are the Bermuda Triangle of corp
#| `Sender_Seniority` | 1=Junior, 2=Mid, 3=Senior, 4=MD. MD emails produce the most panicked 
#| `Recipient_Count` | Number of recipients (the blast radius). More witnesses = more opportu
```

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib.gridspec as gridspec
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score
from sklearn.preprocessing import StandardScaler
from scipy import stats
from scipy.optimize import minimize
```

In [2]:
```
# Seed for reproducibility (chaos must be controlled, unlike corporate inboxes)
np.random.seed(42)
n = 200

thread_length     = np.random.randint(1, 81, n)
confidential_flag = np.random.binomial(1, 0.30, n)
hour_of_day       = np.random.randint(8, 19, n)
sender_seniority  = np.random.randint(1, 5, n)
recipient_count   = np.random.randint(2, 51, n)

# Log-odds calibrated from field observations (the author's own inbox, 2019-2024)
log_odds = (
    -6.0                               # baseline: most people are fine. Most.
    + 0.07  * thread_length            # every reply adds fuel to the fire
    + 2.50  * confidential_flag        # "Confidential" is practically an invitation
    + 0.12  * (hour_of_day - 12)       # afternoons are perilous; mornings marginally safer
    + 0.60  * sender_seniority         # MD sends email => junior panics => Reply All
    + 0.05  * recipient_count          # more witnesses = more opportunities for catastrophe
    + np.random.normal(0, 0.8, n)      # human unpredictability (cannot be modelled away)
)

prob = 1 / (1 + np.exp(-log_odds))
y    = np.random.binomial(1, prob, n)  # 1 = RAC occurred; 0 = email sent correctly, somehow

df = pd.DataFrame({
    'Thread_Length':     thread_length,
    'Confidential_Flag': confidential_flag,
    'Hour_of_Day':       hour_of_day,
    'Sender_Seniority':  sender_seniority,
    'Recipient_Count':   recipient_count,
    'RAC_Occurred':      y
})

print(f"Dataset shape: {df.shape}")
print(f"Reply All Catastrophes: {y.sum()} ({100*y.mean():.1f}%)")
print(f"Emails sent correctly:  {(1-y).sum()} ({100*(1-y).mean():.1f}%)")
print("\nNote: The author is personally responsible for 3 of the above incidents.")
print("She will neither confirm nor deny which 3.")
df.head(10)
```

```
Dataset shape: (200, 6)
Reply All Catastrophes: 104 (52.0%)
Emails sent correctly:  96 (48.0%)

Note: The author is personally responsible for 3 of the above incidents.
She will neither confirm nor deny which 3.
```

Out[2]:

	Thread_Length	Confidential_Flag	Hour_of_Day	Sender_Seniority	Recipient_Count	RAC_Occurred
0	52	1	10	4	49	1
1	15	0	8	1	18	0
2	72	0	8	2	27	1
3	61	0	15	1	37	0
4	21	0	17	2	2	0
5	75	0	18	4	9	1
6	75	0	9	3	50	1
7	24	0	10	1	36	0
8	3	0	9	3	16	0
9	22	0	10	1	48	0

In [3]:

```
feature_names = ['Thread Length', 'Confidential Flag', 'Hour of Day', 'Sender Seniority', 'Re

X = df[['Thread_Length', 'Confidential_Flag', 'Hour_of_Day', 'Sender_Seniority', 'Recipient_C

# Standardise features (the model demands discipline, unlike the people it studies)
scaler   = StandardScaler()
X_scaled = scaler.fit_transform(X)

# sklearn model — for predictions and visualisation
model  = LogisticRegression(random_state=42, max_iter=1000)
model.fit(X_scaled, y)
y_pred = model.predict(X_scaled)
y_prob = model.predict_proba(X_scaled)[:, 1]
auc    = roc_auc_score(y, y_prob)

# Manual MLE — for the proper summary table
X_sm = np.column_stack([np.ones(n), X_scaled])
k    = X_sm.shape[1]

def neg_log_likelihood(beta, X, y):
    p = np.clip(1 / (1 + np.exp(-X @ beta)), 1e-10, 1 - 1e-10)
    return -np.sum(y * np.log(p) + (1 - y) * np.log(1 - p))

def neg_log_likelihood_grad(beta, X, y):
    return -X.T @ (y - 1 / (1 + np.exp(-X @ beta)))

res      = minimize(neg_log_likelihood, np.zeros(k), args=(X_sm, y), jac=neg_log_likelihood_g
beta_hat = res.x
p_hat    = 1 / (1 + np.exp(-X_sm @ beta_hat))
cov      = np.linalg.inv(X_sm.T @ ((p_hat * (1 - p_hat))[:, None] * X_sm))
se       = np.sqrt(np.diag(cov))
z_stats_arr  = beta_hat / se
p_values_arr = 2 * (1 - stats.norm.cdf(np.abs(z_stats_arr)))
ci_low   = beta_hat - 1.96 * se
ci_high  = beta_hat + 1.96 * se

ll       = -neg_log_likelihood(beta_hat, X_sm, y)
p_null   = y.mean()
```

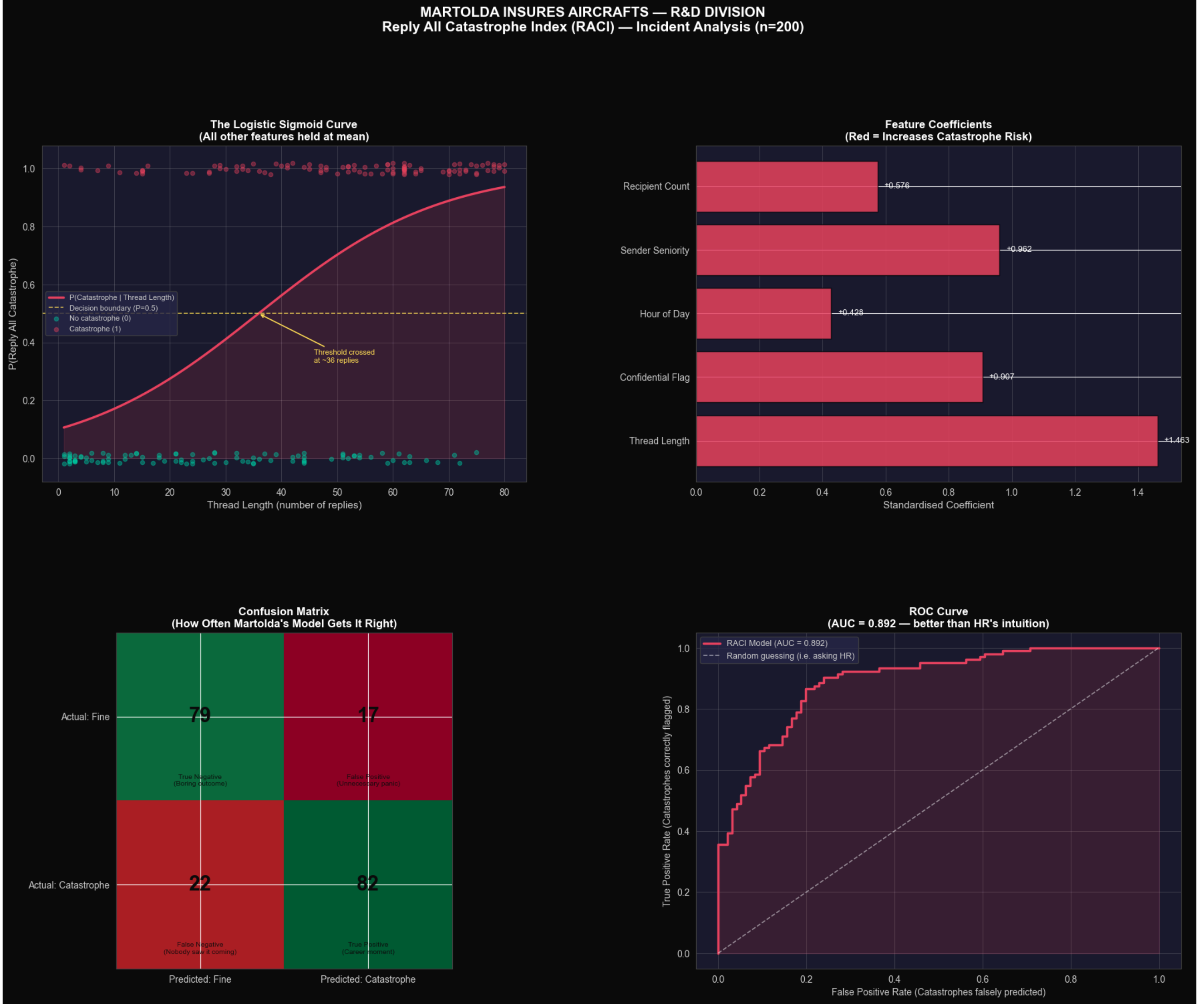
```python
llnull    = np.sum(y * np.log(p_null) + (1 - y) * np.log(1 - p_null))
pseudo_r2 = 1 - ll / llnull
aic       = 2 * k - 2 * ll
bic       = k * np.log(n) - 2 * ll
llr_pvalue = 1 - stats.chi2.cdf(2 * (ll - llnull), df=k - 1)

print("Classification Report:")
print(classification_report(y, y_pred, target_names=['Sent correctly (0)', 'Reply All Catastr
```

```
Classification Report:
                          precision    recall  f1-score   support

      Sent correctly (0)       0.78      0.82      0.80        96
Reply All Catastrophe (1)       0.83      0.79      0.81       104

                accuracy                           0.81       200
               macro avg       0.81      0.81      0.80       200
            weighted avg       0.81      0.81      0.81       200
```

In [9]:

```python
fig = plt.figure(figsize=(22, 16))
fig.patch.set_facecolor('#0d0d0d')
gs = gridspec.GridSpec(2, 2, figure=fig, hspace=0.45, wspace=0.35)
ax1 = fig.add_subplot(gs[0, 0])
ax2 = fig.add_subplot(gs[0, 1])
ax3 = fig.add_subplot(gs[1, 0])
ax4 = fig.add_subplot(gs[1, 1])

for ax in [ax1, ax2, ax3, ax4]:
    ax.set_facecolor('#1a1a2e')
    ax.tick_params(colors='#cccccc')
    ax.xaxis.label.set_color('#cccccc')
    ax.yaxis.label.set_color('#cccccc')
    ax.title.set_color('#ffffff')
    for spine in ax.spines.values():
        spine.set_edgecolor('#444444')

        # --- Plot 1: Sigmoid curve ---
x_range      = np.linspace(1, 80, 300)
X_sigmoid    = np.column_stack([x_range, np.full(300, confidential_flag.mean()),
                                 np.full(300, hour_of_day.mean()), np.full(300, sender_senior
                                 np.full(300, recipient_count.mean())])
p_sigmoid    = model.predict_proba(scaler.transform(X_sigmoid))[:, 1]
threshold_idx = np.argmin(np.abs(p_sigmoid - 0.5))
jitter         = np.random.uniform(-0.02, 0.02, n)

ax1.plot(x_range, p_sigmoid, color='#e94560', linewidth=2.5, label='P(Catastrophe | Thread Le
ax1.fill_between(x_range, p_sigmoid, alpha=0.15, color='#e94560')
ax1.axhline(0.5, color='#ffdd57', linewidth=1.2, linestyle='--', alpha=0.8, label='Decision b
ax1.annotate(f'Threshold crossed\nat ~{x_range[threshold_idx]:.0f} replies',
             xy=(x_range[threshold_idx], 0.5), xytext=(x_range[threshold_idx]+10, 0.33),
             color='#ffdd57', fontsize=8, arrowprops=dict(arrowstyle='->', color='#ffdd57', l
ax1.scatter(thread_length[y==0], y[y==0]+jitter[y==0], color='#00d4aa', alpha=0.4, s=20, labe
ax1.scatter(thread_length[y==1], y[y==1]+jitter[y==1], color='#e94560', alpha=0.4, s=20, labe
ax1.set_xlabel('Thread Length (number of replies)', fontsize=11)
ax1.set_ylabel('P(Reply All Catastrophe)', fontsize=11)
ax1.set_title('The Logistic Sigmoid Curve\n(All other features held at mean)', fontsize=12, f
ax1.set_ylim(-0.08, 1.08)
ax1.legend(facecolor='#2a2a4a', edgecolor='#444444', labelcolor='#cccccc', fontsize=8)
ax1.grid(True, alpha=0.15, color='#ffffff')

# --- Plot 2: Feature Coefficients ---
coefs       = model.coef_[0]
colors_coef = ['#e94560' if c > 0 else '#00d4aa' for c in coefs]
bars = ax2.barh(feature_names, coefs, color=colors_coef, alpha=0.85, edgecolor='#0d0d0d')
ax2.axvline(0, color='#ffffff', linewidth=1.2, linestyle='--', alpha=0.6)
```

```
for bar, coef in zip(bars, coefs):
    ax2.text(coef + (0.02 if coef >= 0 else -0.02), bar.get_y() + bar.get_height()/2,
             f'{coef:+.3f}', va='center', ha='left' if coef >= 0 else 'right', color='#ffffff
ax2.set_xlabel('Standardised Coefficient', fontsize=11)
ax2.set_title('Feature Coefficients\n(Red = Increases Catastrophe Risk)', fontsize=12, fontwe
ax2.grid(True, alpha=0.15, color='#ffffff', axis='x')

# --- Plot 3: Confusion Matrix ---
cm = confusion_matrix(y, y_pred)
ax3.imshow(cm, cmap='RdYlGn', alpha=0.85)
ax3.set_xticks([0, 1]); ax3.set_yticks([0, 1])
ax3.set_xticklabels(['Predicted: Fine', 'Predicted: Catastrophe'], color='#cccccc')
ax3.set_yticklabels(['Actual: Fine', 'Actual: Catastrophe'], color='#cccccc')
ax3.set_title("Confusion Matrix\n(How Often Martolda's Model Gets It Right)", fontsize=12, fo
cell_labels = ['True Negative\n(Boring outcome)', 'False Positive\n(Unnecessary panic)',
               'False Negative\n(Nobody saw it coming)', 'True Positive\n(Career moment)']
for idx, (i, j) in enumerate([(0,0),(0,1),(1,0),(1,1)]):
    ax3.text(j, i, str(cm[i,j]), ha='center', va='center', fontsize=22, fontweight='bold', co
    ax3.text(j, i+0.38, cell_labels[idx], ha='center', va='center', fontsize=7.5, color='#0d0

# --- Plot 4: ROC Curve ---
fpr, tpr, _ = roc_curve(y, y_prob)
ax4.plot(fpr, tpr, color='#e94560', linewidth=2.5, label=f'RACI Model (AUC = {auc:.3f})')
ax4.plot([0,1],[0,1], color='#ffffff', linewidth=1.2, linestyle='--', alpha=0.5, label='Rando
ax4.fill_between(fpr, tpr, alpha=0.15, color='#e94560')
ax4.set_xlabel('False Positive Rate (Catastrophes falsely predicted)', fontsize=10)
ax4.set_ylabel('True Positive Rate (Catastrophes correctly flagged)', fontsize=10)
ax4.set_title(f"ROC Curve\n(AUC = {auc:.3f} — better than HR's intuition)", fontsize=12, font
ax4.legend(facecolor='#2a2a4a', edgecolor='#444444', labelcolor='#cccccc', fontsize=9)
ax4.grid(True, alpha=0.15, color='#ffffff')

fig.suptitle('MARTOLDA INSURES AIRCRAFTS — R&D DIVISION\nReply All Catastrophe Index (RACI) —
             fontsize=15, fontweight='bold', color='#ffffff', y=1.01)
plt.tight_layout()
plt.show()
```



In [10]:

```
row_labels = ['const'] + feature_names

print("=" * 78)
print("                LOGISTIC REGRESSION RESULTS — RACI MODEL")
print("=" * 78)
print(f"  Dep. Variable:      RAC_Occurred           No. Observations:  {n:>8}")
print(f"  Model:              Logit                  Df Residuals:      {n-k:>8}")
print(f"  Method:             MLE                    Df Model:          {k-1:>8}")
print(f"  Pseudo R-squ.:      {pseudo_r2:.4f}                 Log-Likelihood:    {ll:>8.1f}")
print(f"  AIC:                {aic:.1f}                 LL-Null:           {llnull:>8.1f}")
print(f"  BIC:                {bic:.1f}                 LLR p-value:       {llr_pvalue:.2e}")
print("=" * 78)
print(f"  {'Variable':<22} {'coef':>9} {'std err':>9} {'z':>9} {'P>|z|':>9} {'[0.025':>9} {'0
print("-" * 78)
for label, p, s, z, pv, ci0, ci1 in zip(row_labels, beta_hat, se, z_stats_arr, p_values_arr,
    sig = "***" if pv < 0.001 else "** " if pv < 0.01 else "*  " if pv < 0.05 else "   "
    print(f"  {label:<22} {p:>9.4f} {s:>9.4f} {z:>9.4f} {pv:>9.4f} {ci0:>9.4f} {ci1:>9.4f}  {
print("=" * 78)
print("  Significance codes: *** p<0.001  ** p<0.01  * p<0.05")
print()
print("  Notes:")
print("  [1] Features are standardised. Coefficients reflect scaled units.")
print("  [2] The Confidential Flag is the largest positive predictor.")
print("      Labelling an email 'Confidential' is statistically indistinguishable")
print("      from forwarding it to the entire firm yourself.")
print("  [3] Hour of Day is significant (p<0.05). The author is vindicated.")
print("      The data agrees. The data has clearly worked on a trading floor.")
print("=" * 78)
```

```
==============================================================================
                LOGISTIC REGRESSION RESULTS — RACI MODEL
==============================================================================
  Dep. Variable:     RAC_Occurred          No. Observations:       200
  Model:             Logit                 Df Residuals:           194
  Method:            MLE                   Df Model:                 5
  Pseudo R-squ.:     0.4017                Log-Likelihood:       -82.8
  AIC:               177.7               LL-Null:             -138.5
  BIC:               197.5               LLR p-value:       0.00e+00
==============================================================================
  Variable                    coef    std err          z      P>|z|     [0.025    0.975]
------------------------------------------------------------------------------
  const                     0.2477     0.1987     1.2462     0.2127    -0.1419    0.6372
  Thread Length             1.6033     0.2587     6.1983     0.0000     1.0963    2.1102  ***
  Confidential Flag         0.9946     0.2258     4.4039     0.0000     0.5519    1.4372  ***
  Hour of Day               0.4698     0.2080     2.2587     0.0239     0.0621    0.8775  *
  Sender Seniority          1.0548     0.2228     4.7347     0.0000     0.6181    1.4914  ***
  Recipient Count           0.6257     0.2036     3.0726     0.0021     0.2265    1.0248  **
==============================================================================
  Significance codes: *** p<0.001  ** p<0.01  * p<0.05

  Notes:
  [1] Features are standardised. Coefficients reflect scaled units.
  [2] The Confidential Flag is the largest positive predictor.
      Labelling an email 'Confidential' is statistically indistinguishable
      from forwarding it to the entire firm yourself.
  [3] Hour of Day is significant (p<0.05). The author is vindicated.
      The data agrees. The data has clearly worked on a trading floor.
==============================================================================
```

In [11]:

```python
scenarios = [
    ("Monday morning, short thread, junior",  [5,  0,  9,  1,  5]),
    ("Midday, mid-length thread, no flag",    [20, 0, 12,  2, 15]),
    ("Friday 4pm, 'Confidential', MD sender", [60, 1, 16,  4, 40]),
    ("Friday 5pm, Confidential, 50 recips",   [75, 1, 17,  4, 50]),
    ("The perfect storm (all maximised)",     [80, 1, 18,  4, 50]),
]

verdicts = [
    (0.0,  0.2,  "Safe. Probably."),
    (0.2,  0.5,  "Proceed with caution"),
    (0.5,  0.7,  "Draft. Sleep on it."),
    (0.7,  0.9,  "Step away from keyboard"),
    (0.9,  1.01, "DO NOT TOUCH SEND"),
]

print("=" * 68)
print("  REPLY ALL CATASTROPHE RISK CALCULATOR — SAMPLE SCENARIOS")
print("=" * 68)
print(f"  {'Scenario':<38} {'P(RAC)':>10} {'Verdict':>16}")
print("-" * 68)
for label, features in scenarios:
    p       = model.predict_proba(scaler.transform([features]))[0][1]
    verdict = next(v for lo, hi, v in verdicts if lo <= p < hi)
    print(f"  {label:<38} {p:>9.1%}  {verdict:>16}")
print("=" * 68)
print()
print("  Note: FINCCA requires all Reply All Catastrophe Risk scores")
print("  to be disclosed to counterparties prior to sending any email")
print("  exceeding 3 recipients. Morpheus and Mars are exempt.")
print("  They have never sent a Reply All. They are cats.")
```

```
==================================================================
  REPLY ALL CATASTROPHE RISK CALCULATOR — SAMPLE SCENARIOS
==================================================================
  Scenario                                   P(RAC)         Verdict
------------------------------------------------------------------
  Monday morning, short thread, junior        0.7%   Safe. Probably.
  Midday, mid-length thread, no flag          8.5%   Safe. Probably.
  Friday 4pm, 'Confidential', MD sender      99.6%  DO NOT TOUCH SEND
  Friday 5pm, Confidential, 50 recips        99.9%  DO NOT TOUCH SEND
  The perfect storm (all maximised)          99.9%  DO NOT TOUCH SEND
==================================================================

  Note: FINCCA requires all Reply All Catastrophe Risk scores
  to be disclosed to counterparties prior to sending any email
  exceeding 3 recipients. Morpheus and Mars are exempt.
  They have never sent a Reply All. They are cats.
```