Simulation Data Analysis Demo

James Jang

2025-12-22

1. Load Libraries

The library `tidyverse` is used for data manipulation and plot generation. `here` is used to create an OS-independent path to access the `simulation_data.csv` file, and `car` is used to perform Levene's Test.

```
if(!require(here)) install.packages("here")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(car)) install.packages("car")
library(tidyverse)
library(here)
library(car)
```

2. Load Data

I am using a pre-generated CSV file for demonstration purposes. If you would like to replicate the analysis with your own data, use the `analysis.r` file.

```
df <- read.csv("simulation_data_demo.csv")
df$Success <- as.logical(df$Success) # Converting numerical type (1/0) to logical (true/false)
```

3. Data Quality Check

I begin by viewing an overview of the data with `glimpse()`. I also verify that the data is clean and ready for further analysis.

- **RunId**: Unique identifier for each row (run).
- **Success**: Indicates whether the system survived or failed (System Jam).
- **Duration**: The count of *'ticks'* (seconds) the simulation lasted.
- **TotalTrucks**: Total number of trucks that arrived at the hub.
- **TotalParcels**: Total number of parcels that were injected to the main conveyor belt.
- **MinInterarrival**: Shortest time difference between two truck visits.
- **MaxBeltLoad**: Largest number of parcels on the main belt at any given time.
- **MaxStationLoad**: Largest queue length of any station at any given time.
- **AvgProcessingTime**: Average time to process a single parcel (worker speed).
- **StationLoadStdDev**: Standard deviation of the total parcels ingested by each station.

```
# Structure and Column types
glimpse(df)
```

```
Rows: 1,000
Columns: 10
$ RunId             <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
$ Success           <lgl> TRUE, TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, TRUE, FA~
$ Duration          <int> 28800, 28800, 21815, 28800, 28800, 18712, 28800, 288~
$ TotalTrucks       <int> 68, 57, 44, 71, 78, 36, 70, 60, 33, 42, 65, 55, 54, ~
$ TotalParcels      <int> 6971, 5576, 4538, 6777, 7680, 3796, 7200, 6118, 3393~
$ MinInterarrival   <int> 65, 21, 1, 6, 10, 37, 1, 2, 15, 1, 3, 4, 7, 7, 4, 26~
$ MaxBeltLoad       <int> 170, 142, 300, 264, 260, 300, 294, 239, 300, 300, 22~
$ MaxStationLoad    <int> 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, ~
$ AvgProcessingTime <dbl> 4.98, 4.99, 5.02, 5.00, 4.99, 4.99, 4.98, 5.01, 5.00~
$ StationLoadStdDev <dbl> 141.98, 69.03, 132.24, 97.13, 141.76, 108.59, 152.84~
```

```
# Count total missing values
sum(is.na(df))
```

```
[1] 0
```

```
# Find and print any invalid rows (negative truck count or duration)
invalid_runs <- df %>%
  filter(TotalTrucks <= 0 | Duration <= 0)
print(invalid_runs)
```

```
 [1] RunId             Success           Duration          TotalTrucks
 [5] TotalParcels      MinInterarrival   MaxBeltLoad       MaxStationLoad
 [9] AvgProcessingTime StationLoadStdDev
<0 rows> (or 0-length row.names)
```

4. Exploratory Data Analysis

Next, I applied descriptive statistics and visualization techniques to gain insights into the main simulation metrics.

Hypothesizing that metrics with high variability (standard deviation) across runs are the likely determinants of success or failure, I calculated the standard deviations for all numerical columns.

The boxplot below compares the successful simulation group against the failure group.

```
# Entire summary
df %>% select(-RunId) %>% summary()
```

```
  Success           Duration      TotalTrucks     TotalParcels
 Mode :logical   Min.   :  268   Min.   : 4.00   Min.   : 421
 FALSE:311       1st Qu.:22246   1st Qu.:48.75   1st Qu.:4846
 TRUE :689       Median :28800   Median :59.00   Median :5938
                 Mean   :23993   Mean   :53.46   Mean   :5336
                 3rd Qu.:28800   3rd Qu.:66.00   3rd Qu.:6563
                 Max.   :28800   Max.   :85.00   Max.   :8496
 MinInterarrival   MaxBeltLoad    MaxStationLoad AvgProcessingTime
 Min.   : 1.000   Min.   :141.0   Min.   :50     Min.   :4.630
 1st Qu.: 3.000   1st Qu.:222.0   1st Qu.:50     1st Qu.:4.990
 Median : 6.000   Median :259.0   Median :50     Median :5.000
```

```
 Mean   : 9.198   Mean   :254.3   Mean   :50     Mean   :5.002
 3rd Qu.:12.000   3rd Qu.:300.0   3rd Qu.:50     3rd Qu.:5.020
 Max.   :69.000   Max.   :300.0   Max.   :50     Max.   :5.240
 StationLoadStdDev
 Min.   : 17.83
 1st Qu.: 94.08
 Median :115.49
 Mean   :110.90
 3rd Qu.:133.87
 Max.   :207.40
```

```r
# Survival & Failure Rate (%)
survival_data <- df %>%
  summarise(
    Total_Runs = n(),
    Total_Successes = sum(Success),
    Survival_Rate_Percent = mean(Success) * 100,
    Failure_Rate_Percent = (1-mean(Success)) * 100
  )
print(survival_data)
```

```
  Total_Runs Total_Successes Survival_Rate_Percent Failure_Rate_Percent
1       1000             689                  68.9                 31.1
```

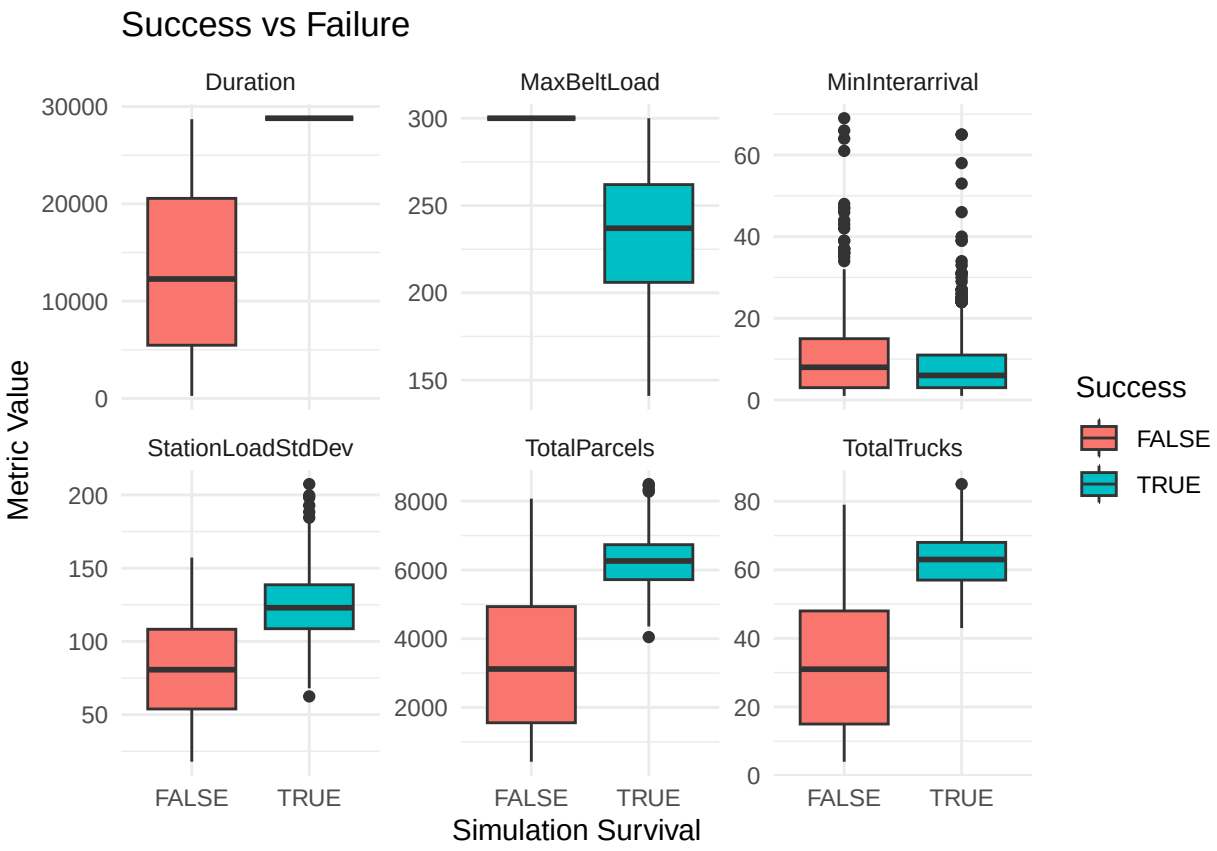
```r
# Standard Deviations (Imbalance)
metric_spread <- df %>%
  select(where(is.numeric)) %>%
  select(-RunId, -Duration, -MaxStationLoad) %>%
  summarise(across(everything(), sd)) %>%
  pivot_longer(everything(), names_to = "Metric", values_to = "Std_Dev") %>%
  mutate(Std_Dev = signif(Std_Dev, 4)) %>%
  arrange(desc(Std_Dev))
print(metric_spread)
```

```
# A tibble: 6 x 2
  Metric              Std_Dev
  <chr>                 <dbl>
1 TotalParcels       1863
2 MaxBeltLoad          43.7
3 StationLoadStdDev    33.0
4 TotalTrucks          18.7
5 MinInterarrival       9.55
6 AvgProcessingTime     0.0302
```

```r
# Box Plots: Failure vs Success
long_data <- df %>%
  select(Success, Duration, TotalTrucks, TotalParcels, MinInterarrival, MaxBeltLoad, StationLoadStdDev)
  pivot_longer(cols = -Success, names_to = "Metric", values_to = "Value")

ggplot(long_data, aes(x = Success, y = Value, fill = Success)) +
  geom_boxplot() +
  facet_wrap(~Metric, scales='free_y') +
```

```
  theme_minimal() +
  labs(title = "Success vs Failure",
       y = "Metric Value",
       x = "Simulation Survival")
```



Interpretation

- **Saturation is constant:** The summary shows that `MaxStationLoad` reached 50 in all runs, indicating that at least one station hits its absolute capacity limit (saturation) 100% of the time, regardless of whether the system eventually crashes or not.
- **Volume metrics are time-dependent:** Failed simulations generally had fewer `TotalTrucks` and `TotalParcels` than the Success group (see boxplot). However, because these totals depend directly on the duration of the simulation, they are biased by time and offer limited explanatory power regarding system failures.
- **Worker efficiency is uniform:** The low standard deviation in `AvgProcessingTime` indicates that worker speed was consistent across all runs. This effectively rules out worker inefficiency as a possible cause of system failure.
- **Belt Load is a symptom, not a cause:** `MaxBeltLoad` exhibited relatively high standard deviations compared to other metrics. However, it will be excluded from causal analysis because it is a lagging indicator. A simulation reaching a `MaxBeltLoad` of 300 is simply the definition of a jam, it lacks the additional insight we are looking for.

Consequently, I will normalize the three time-dependent metrics (`TotalTrucks`, `TotalParcels`, and `Duration`) with respect to time to derive three time-independent rates: `AvgTruckSize`, `ParcelsPerMinute`,

and `TruckPace`. These metrics allow me to capture the intensity of the traffic, distinct from total accumulated volumes.

Armed with these new indicators, alongside `StationLoadStdDev` and `MinInterarrival`, I will delve deeper into the forensics of system failure.

5. Feature Engineering

Here, I quickly create the 3 new time-independent metrics.

```
# Create new columns
df_eng <- df %>%
  mutate(
    ParcelsPerMinute = (TotalParcels / Duration) * 60,
    TruckPace = Duration / TotalTrucks,
    AvgTruckSize = TotalParcels / TotalTrucks
  )

# Preview of new columns
df_eng %>%
  select(Success, ParcelsPerMinute, TruckPace, AvgTruckSize) %>%
  head()
```

```
  Success ParcelsPerMinute TruckPace AvgTruckSize
1    TRUE         14.52292  423.5294    102.51471
2    TRUE         11.61667  505.2632     97.82456
3   FALSE         12.48132  495.7955    103.13636
4    TRUE         14.11875  405.6338     95.45070
5    TRUE         16.00000  369.2308     98.46154
6   FALSE         12.17187  519.7778    105.44444
```

6. Forensics

Next, I use the correaltion matrix to investigate which metrics have the strongest correlation with `Duration` among failed runs to see what accelerates system failure.

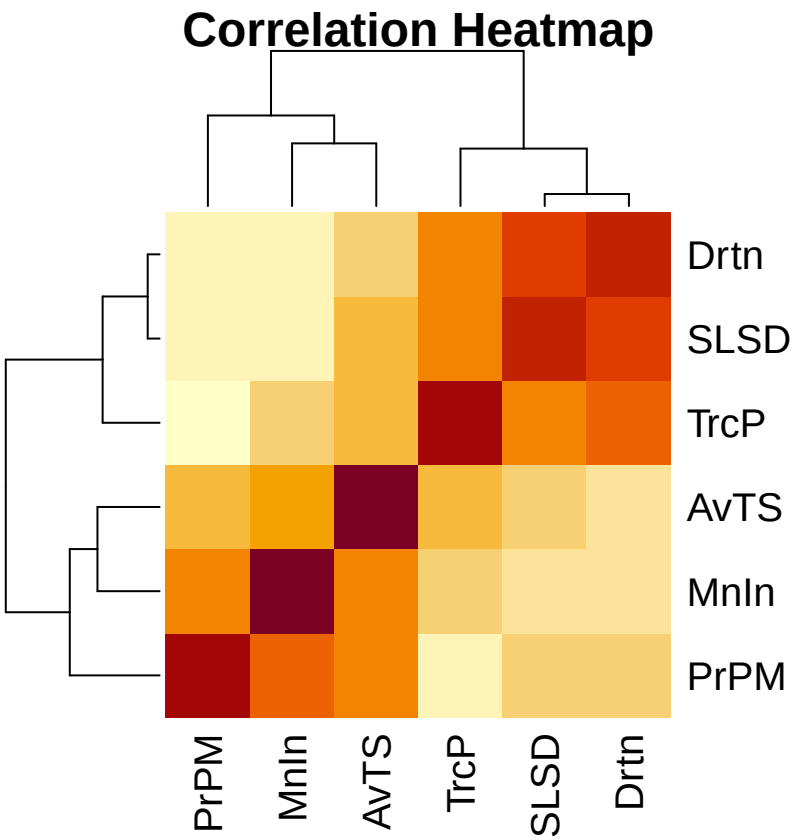
```
# Filter failures
failures_only <- df_eng %>% filter(Success == F)
failed_metrics <- failures_only %>%
  select(
    Duration,
    StationLoadStdDev,
    MinInterarrival,
    ParcelsPerMinute,
    TruckPace,
    AvgTruckSize
  )

# Correlation Matrix
cor_matrix <- cor(failed_metrics)
failure_factors <- sort(cor_matrix[, "Duration"])
```

```r
failure_cor_df <- enframe(failure_factors, name = "Metric", value = "Correlation with Duration") %>% fi
print(failure_cor_df)
```

```
# A tibble: 5 x 2
  Metric            `Correlation with Duration`
  <chr>                                   <dbl>
1 ParcelsPerMinute                      -0.445
2 MinInterarrival                       -0.388
3 AvgTruckSize                          -0.0179
4 TruckPace                              0.513
5 StationLoadStdDev                      0.880
```

```r
# Heatmap
colnames(cor_matrix) <- abbreviate(colnames(cor_matrix), minlength = 4)
rownames(cor_matrix) <- abbreviate(rownames(cor_matrix), minlength = 4)
heatmap(cor_matrix, main = "Correlation Heatmap")
```

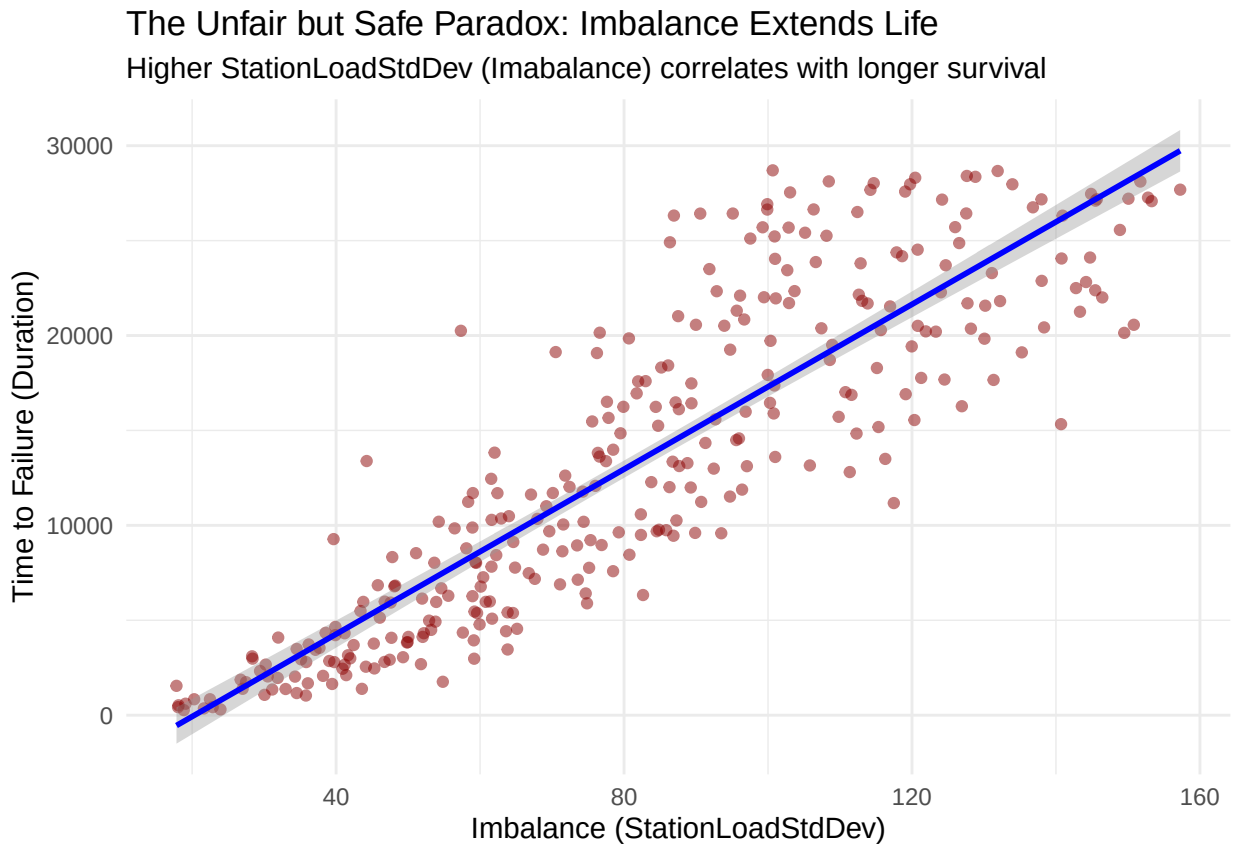


Interpretation Analyzing the correlation matrix reveals three distinct points:

- **The Irrelevant**: `AvgTruckSize` shows near-zero correlation (-0.01), suggesting that batch size itself is not a risk factor.
- **The Obvious**: `ParcelsPerMinute` (-0.45) and `TruckPace` (+0.51) confirm that intensity is the primary stressor; higher speeds yield shorter lifespans.

- **The Paradox**: The strongest correlation, however, belonged to `StationLoadStdDev` (+0.88), which presents a counter-intuitive mechanism: the most 'balanced' runs (low StdDev) are actually the ones that crash the fastest. This phenomenon presents an "Unfair but Safe" paradox. I visualize this strong relationship using a scatter plot below.

```
# Scatter Plot: StationLoadStdDev vs Duration
ggplot(failures_only, aes(x = StationLoadStdDev, y = Duration)) +
  geom_point(alpha = 0.5, color = "darkred") +
  geom_smooth(method = "lm", color = "blue") +
  labs(
    title = "The Unfair but Safe Paradox: Imbalance Extends Life",
    subtitle = "Higher StationLoadStdDev (Imabalance) correlates with longer survival",
    x = "Imbalance (StationLoadStdDev)",
    y = "Time to Failure (Duration)"
  ) +
  theme_minimal()
```



7. Hypothesis Testing

While I observed a strong correlation between "Uniform Saturation" (`StationLoadStdDev` ≈ 0) and system failure, `StationLoadStdDev` is an outcome of the simulation rather than a controllable input. Therefore, it serves as a symptom rather than the root cause. To identify the actual driver of these jams, I must investigate the input parameters.

I propose two competing theories regarding the root cause of failure:

- **Theory A - The "Machine Gun" Hypothesis** (`MinInterarrival`): System failure is driven by *Instability* and *Aggression*. We suspect that erratic, "bursty" (high variance in gaps and truck sizes) and "clustered" (short time intervals between) truck arrivals overwhelm the conveyor belt momentarily, causing jams even if the average volume is manageable.
- **Theory B - The "Speed Limit" Hypothesis** (`ParcelsPerMinute`): System failure is driven by *Volume*. We suspect that the system simply has a hard physical throughput limit. If the input intensity exceeds this limit, the system will inevitably jam, regardless of how consistent the arrival gaps are.

Theory A According to this theory, failed runs should exhibit significantly higher variance in arrival gaps as well as truck sizes. Also, failed runs should generally have shorter arrival gaps than successful runs.

I will use Levene's Test to compare the variance of `MinInterarrival`. The Mann-Whitney U Test will be used check if failed runs had stochastically smaller gaps between truck arrivals than successful runs. Since `AvgTruckSize` had near-zero correlation with failure (see the correlation matrix), it will be excluded from further hypothesis testing.

```
# Calculate Coefficients of Variation before Levene's Tests for result directionality
cv_analysis <- df_eng %>%
  group_by(Success) %>%
  summarise(
    InterarrivalCV = signif(sd(MinInterarrival) / mean(MinInterarrival) * 100, 3)
  )
cat("Instability of Arrival Gaps (Coefficient of Variation):\n")
```

```
Instability of Arrival Gaps (Coefficient of Variation):
```

```
print(cv_analysis)
```

```
# A tibble: 2 x 2
  Success InterarrivalCV
  <lgl>            <dbl>
1 FALSE              104
2 TRUE                99.2
```

```
# Levene's Test

# Alternative Hypothesis: Failures have higher variance in arrival gaps
levene_arrival <- leveneTest(MinInterarrival ~ Success, data = df_eng)
cat("--- Levene's Test: Arrival Gap Variance ---\n")
```

```
--- Levene's Test: Arrival Gap Variance ---
```

```
print(levene_arrival)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value    Pr(>F)
group   1   22.42 2.508e-06 ***
      998
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Robust test for variance homogeneity (Insensitive to normality/outliers)
fligner.test(MinInterarrival ~ Success, data = df_eng)
```

```
    Fligner-Killeen test of homogeneity of variances

data:  MinInterarrival by Success
Fligner-Killeen:med chi-squared = 31.131, df = 1, p-value = 2.412e-08
```

```r
# Mann-Whitney U/Wilcoxon Test
# Alternative Hypothesis: Failures generally have smaller gaps than Successes
test_interarrival <- wilcox.test(x = df_eng$MinInterarrival[df_eng$Success == T],
                                 y = df_eng$MinInterarrival[df_eng$Success == F],
                                 data = df_eng,
                                 alternative = "greater")
cat("--- MWU Test: Minimum Interarrival Time ---\n")
```

```
--- MWU Test: Minimum Interarrival Time ---
```

```r
print(test_interarrival)
```

```
    Wilcoxon rank sum test with continuity correction

data:  df_eng$MinInterarrival[df_eng$Success == T] and df_eng$MinInterarrival[df_eng$Success == F]
W = 91466, p-value = 0.9999
alternative hypothesis: true location shift is greater than 0
```

Theory B According to this theory, failed runs should have significantly higher intensity.

I will use the Mann-Whitney U Test to check if failed runs have stochastically higher `ParcelsPerMinute` than successful runs.

- Note: We are only considering `ParcelsPerMinute` and not `TruckPace` because it is collinear with `ParcelsPerMinute` (Intensity). I selected `ParcelsPerMinute` as the primary metric for this theory as it provides a more direct measure of system throughput.

```r
# Alternative Hypothesis: Failures have higher intensity than Successes
test_intensity <- wilcox.test(x = df_eng$ParcelsPerMinute[df_eng$Success == T],
                              y = df_eng$ParcelsPerMinute[df_eng$Success == F],
                              data = df_eng,
                              alternative = "less")
cat("--- MWU Test: Parcel Intensity ---\n")
```

```
--- MWU Test: Parcel Intensity ---
```

```r
print(test_intensity)
```

```
    Wilcoxon rank sum test with continuity correction

data:  df_eng$ParcelsPerMinute[df_eng$Success == T] and df_eng$ParcelsPerMinute[df_eng$Success == F]
W = 48593, p-value < 2.2e-16
alternative hypothesis: true location shift is less than 0
```

Interpretation

Theory A

- The coefficient of variance for both `MinInterarrival` and `AvgTruckSize` was higher in the Failure group.
- **Levene's Test** ($p < 0.05$): **Significant**. The failed runs do have significantly higher variance in arrival gaps than successful runs. This confirms that failed environments were indeed more "chaotic" or unstable.
- **Mann-Whitney U** ($p \approx 1.0$): **Not Significant**. While the failed runs were more variable, the gaps were not stochastically smaller (tighter) than successful runs. In fact, the failed runs actually had LARGER gaps on average, meaning systems tend to survive under bursts of truck arrivals.

Theory B

- **Mann-Whitney U** ($p < 2.2e^{-16}$): **Highly Significant**. There is a massive stochastic difference between the groups. The failed runs consistently operated at a higher `ParcelsPerMinute` (intensity) than the successful runs.

These results indicate that Theory A is unlikely to be the root cause of system failure. Conversely, the evidence strongly points to Theory B as the primary determinant.

8. Logistic Regression

Having tested our theories individually, we now integrate them into a single multivariate model. I use Logistic Regression to determine the probability of survival based on the three input dimensions: Intensity (`ParcelsPerMinute`), Volatility (`MinInterarrival`), and Payload (`AvgTruckSize`).

```
# Logistic Regression
risk_model <- glm(Success ~ ParcelsPerMinute + MinInterarrival + AvgTruckSize,
                  data = df_eng,
                  family = "binomial")

# The Raw Coefficients (Log-Odds)
cat("--- Model Summary (Significance Check) ---\n")
```

```
--- Model Summary (Significance Check) ---
```

```
summary(risk_model)
```

```

Call:
glm(formula = Success ~ ParcelsPerMinute + MinInterarrival +
    AvgTruckSize, family = "binomial", data = df_eng)

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      12.558377   2.102185   5.974 2.32e-09 ***
ParcelsPerMinute -0.515193   0.044074 -11.689  < 2e-16 ***
MinInterarrival  -0.038760   0.009231  -4.199 2.68e-05 ***
```

```
AvgTruckSize     -0.040958   0.020400  -2.008   0.0447 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1239.80  on 999  degrees of freedom
Residual deviance:  941.24  on 996  degrees of freedom
AIC: 949.24

Number of Fisher Scoring iterations: 6
```

```r
# The Odds Ratios
# If number < 1: Reduces odds of success (Bad)
# If number > 1: Increases odds of success (Good)
odds_ratios <- exp(coef(risk_model))

cat("--- Odds Ratios (Risk Multipliers) ---\n")
```

```
--- Odds Ratios (Risk Multipliers) ---
```

```r
print(odds_ratios)
```

```
     (Intercept) ParcelsPerMinute  MinInterarrival     AvgTruckSize
    2.844684e+05     5.973851e-01     9.619811e-01     9.598691e-01
```

Interpretation The p-values for all factors are below 0.05, confirming the statistical significance of the results. An analysis of the Odds Ratios identifies `ParcelsPerMinute` as the dominant driver of system outcomes. Specifically, for every 1 unit increase in `ParcelsPerMinute`, the odds of survival drop by 40% (Odds Ratio ≈ 0.6), effectively driving the system toward failure. In comparison, `MinInterarrival` and `AvgTruckSize` showed negligible influence (Odds Ratio ≈ 0.96).

Safety Envelope By plotting the model's predicted probabilities against our primary driver (`ParcelsPerMinute`), I visualize the system's *Safety Envelope* below.

The green reference line is where the system has a 95% probability of surviving. The red area indicates the "Danger Zone" where probability of failure is equal to or greater than 50%.

```r
# The Safe Operating Envelope

# Create the "Hypothetical" Dataset
simulated_data <- data.frame(
  ParcelsPerMinute = seq(5, 25, by = 0.1),
  MinInterarrival = median(df_eng$MinInterarrival),
  AvgTruckSize = median(df_eng$AvgTruckSize)
)

# Predict the Probability of Success
simulated_data$Prob_Success <- predict(risk_model,
                                       newdata = simulated_data,
                                       type = "response")
```

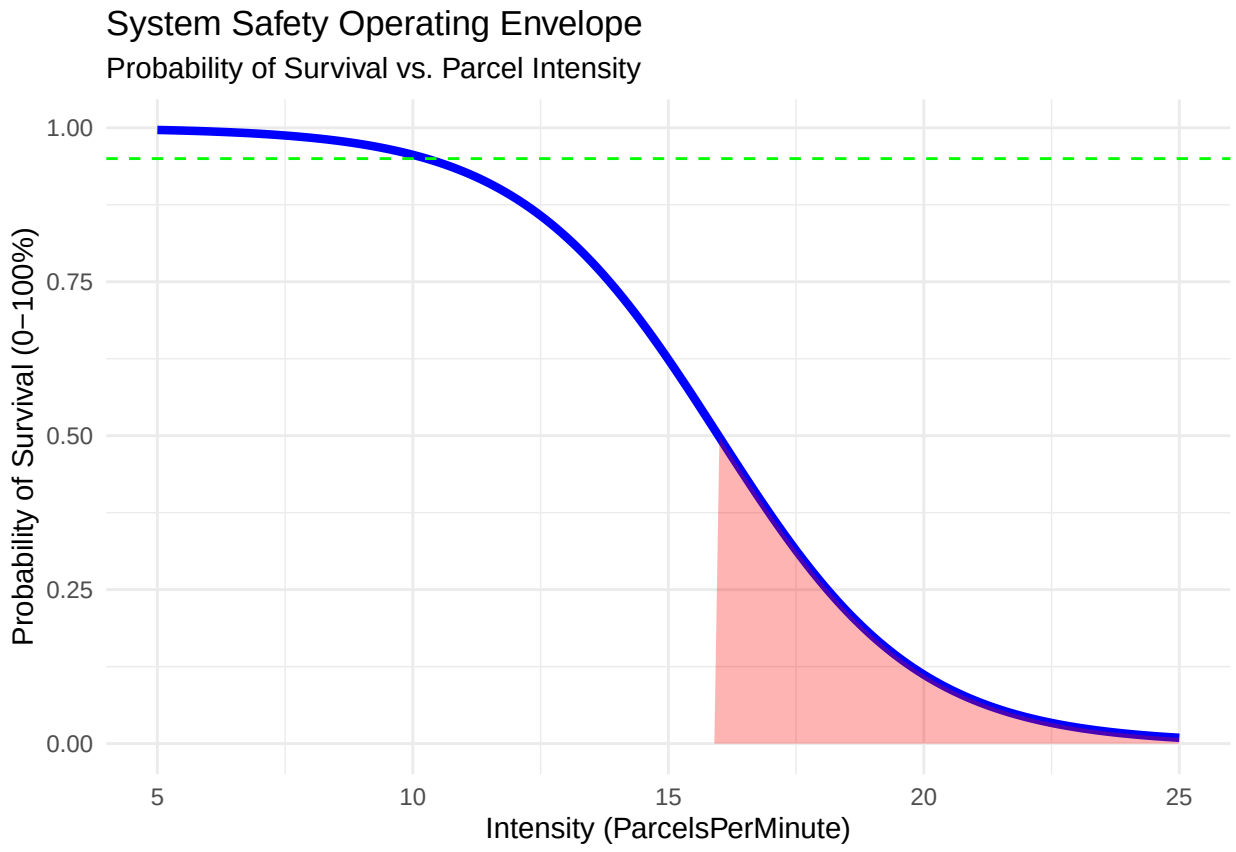
```r
# Plot the Curve
ggplot(simulated_data, aes(x = ParcelsPerMinute, y = Prob_Success)) +
  geom_line(color = "blue", size = 1.5) +

  # Add the "Danger Zone" shading
  geom_area(aes(y = ifelse(Prob_Success < 0.50, Prob_Success, 0)), fill = "red", alpha = 0.3) +

  # Add a reference line at 95% Safety (The "Rated Capacity")
  geom_hline(yintercept = 0.95, linetype = "dashed", color = "green") +

  # Formatting
  labs(
    title = "System Safety Operating Envelope",
    subtitle = "Probability of Survival vs. Parcel Intensity",
    x = "Intensity (ParcelsPerMinute)",
    y = "Probability of Survival (0-100%)"
  ) +

  theme_minimal()
```



9. Conclusion

This analysis concludes that the sortation hub is not failing due to random bad luck or erratic truck patterns. It is failing because it hits a hard physical throughput limit.

In response, I propose two key interventions to address this:

1. **Enforce the "Safety Envelope"**: Operations must adhere to the utilization limits defined by the Logistic Model. Crucially, "Truck Arrival" should no longer equate to "Immediate Injection." We can implement a strict Max Injection Rate at the unloading dock to throttle input intensity (`ParcelsPerMinute`).
2. **Targeted Capacity Expansion**: Insights from the imbalance analysis (`StationLoadStdDev`) reveal that simulations with uneven load distribution actually survived longer. This suggests that specific destinations are naturally heavier than others. By identifying the top 3 highest-volume regions and installing additional stations specifically for them, we can prevent the critical "Uniform Saturation" state that leads to system-wide failure.