Data Wrangling and Cleaning: Steps 1-11

David Mullaly

2026-02-10

Introduction

This document covers the data wrangling process and all eleven data cleaning steps for the problem. We load multiple relational tables, merge them into a single flat file, and perform comprehensive data cleaning to prepare for missing data analysis.

Data Cleaning Steps Covered:

Steps 1-5 (Initial Cleaning)

1. Open data in your software of choice
2. Review variables for common sense based on SME knowledge
3. Review how the software coded the variables (nominal, continuous)
4. Perform data integrity/validation checks
5. Handle dates

Steps 6-7 (Categorical & Zero-Variance)

6. Handle categorical variables - keep as is, combine rare levels, combine similar levels
7. Handle zero-variance predictors

Steps 8-11 (Advanced Cleaning)

8. Handle near zero-variance predictors
9. Eliminate redundant columns and linear combination columns
10. Search for outliers and initial search for missing values
11. Sanity check using Decision Tree (1 to 2 splits)

Data Loading and Wrangling

Load four relational tables and merge into a single flat file (FFdf) using left joins on Cust_ID.

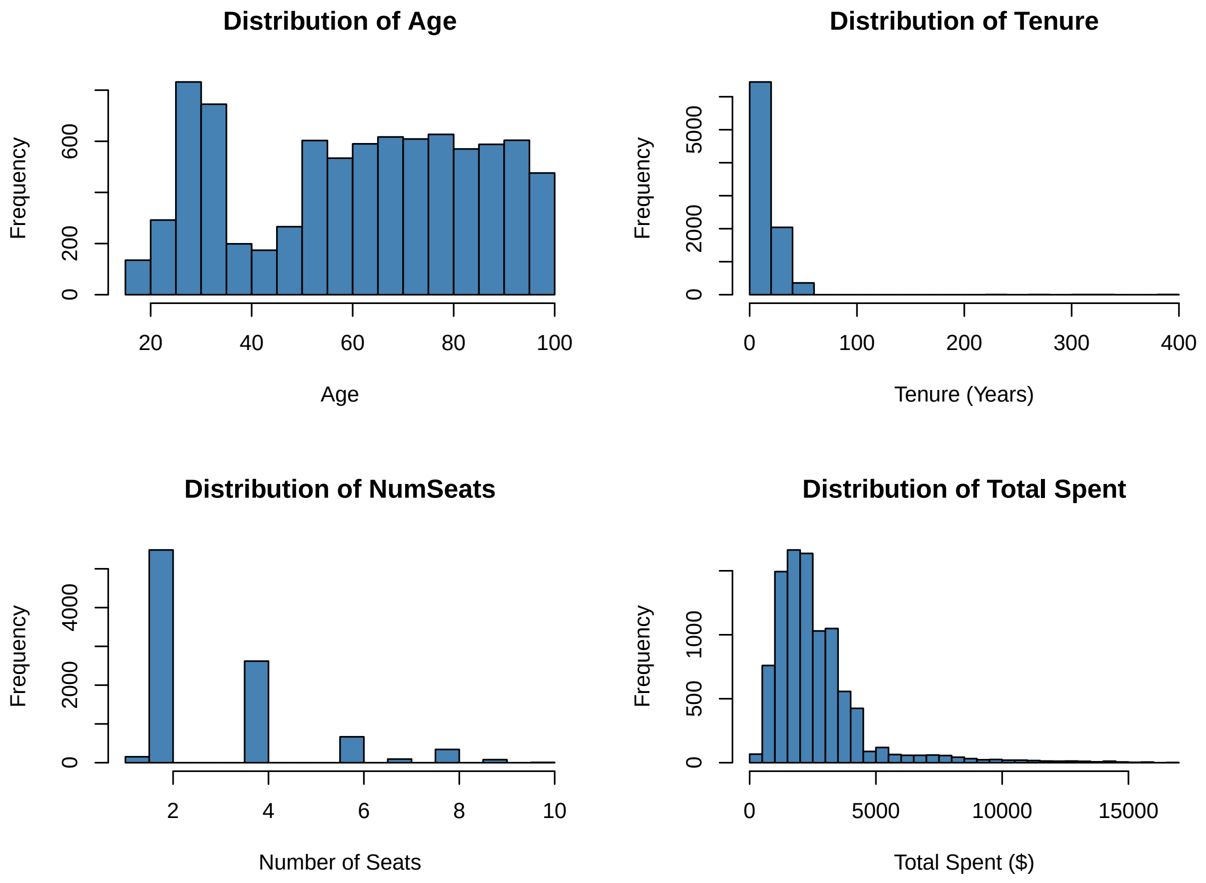
```
## MainDF: 9447 48 | StoreDF: 9272 3 | ConcessDF: 9272 3 | CustomerDF: 14272 7
```

```
## Duplicates in MainDF: 175
```

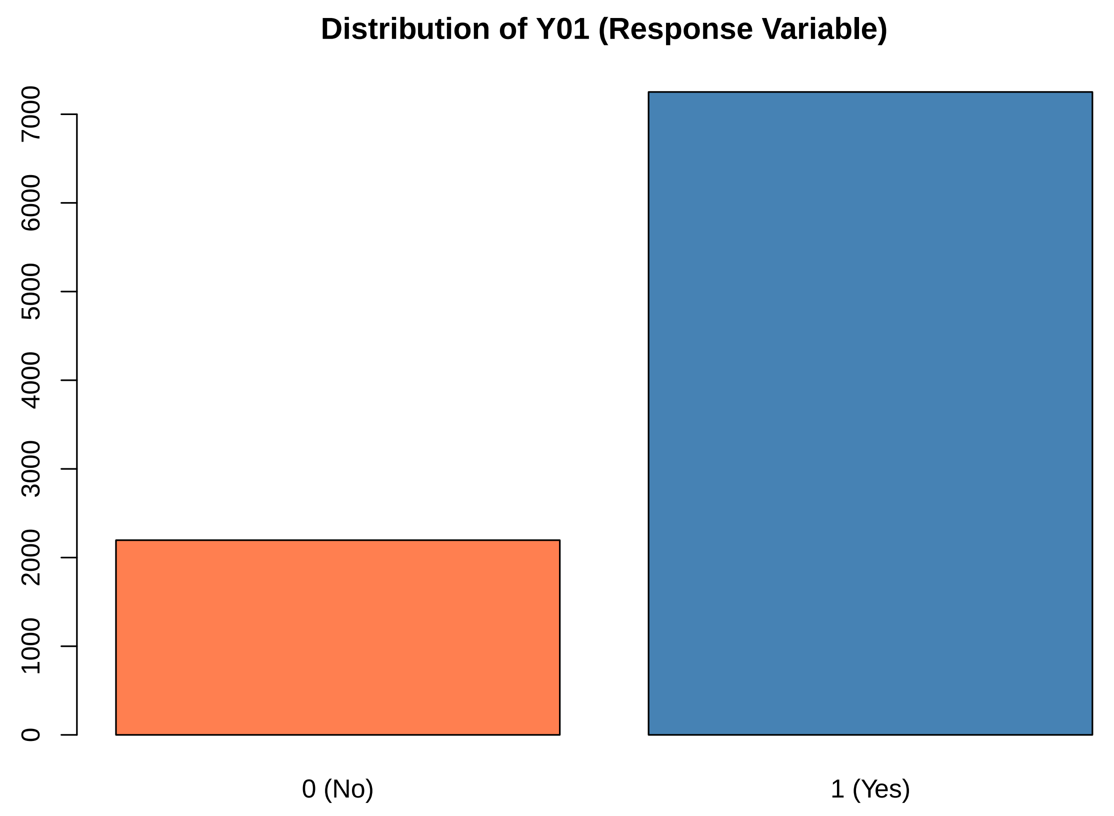
```
## Duplicates in StoreDF: 0
## Duplicates in ConcessDF: 0

## Duplicates in CustomerDF: 0

## Final flat file dimensions - Rows: 9447 Columns: 58

## Unique Cust_ID: 9272 | Duplicate rows: 175
```

Step 1: Open Data in Your Software of Choice

Create identifier variable, arrange columns, and explore distributions visually.



```
## Response Variable (Y01): 2196 7251
```



Observations: Age is roughly normal (30-60), Tenure is right-skewed, NumSeats clusters around 2-4, Total Spent is heavily right-skewed. Response variable Y01 is reasonably balanced.

Step 2: Review Variables for Common Sense (SME Knowledge)

Standardize variable names and check for unique identifiers.

```
## Dataset: 9447 rows, 59 columns
```

```
## Unique Cust_ID: 9272 out of 9447 rows
```

Result: Each row does not necessarily represent a unique customer (175 duplicate Cust_IDs found in MainDF source data).

Step 3: Review How Software Coded Variables

Convert character variables to factors for proper categorical analysis.

```
## Character variables to convert: 23
```

```
##
## Sex levels: F M
## Marital levels: D M S U
```

```
## Account_Type levels: Business Personal Shared
```

Note: Marital has levels D (Divorced), M (Married), S (Single), U (Unknown). The "U" for Unknown may need special handling.

Step 4: Data Integrity/Validation Checks

Check for anomalies, bogus values, and data quality issues.

```
## Age range: 18 99
```

```
## Tenure range: 1 400
```

```
## NumSeats range: 1 10
```

```
##
## DistA values = 999: 1506
```

```
## DistA NA count after conversion: 2895
```

```
##
## Survey_Comp range: 0 7.4
```

```
## Survey_Comp values > 1: 110
```

```
##
## State_Name unique: 50 | State_Loc unique: 50
```

Issues Found:

- **DistA = 999:** Placeholder values converted to NA
- **Survey_Comp > 1:** Outlier found when expected range is 0-1
- **Age max = 99:** May be placeholder or extreme value - verify with SME
- **Tenure max = 400:** Suspicious value if measured in years - verify units with SME
- **State_Name/State_Loc:** Redundant columns (same info, different format)
- **Marital = "U":** Unknown status - consider treating as NA
- **Cust_ID duplicates:** 175 duplicate IDs found in MainDF source data
- **Address, Name, PhoneNum:** 100% missing - likely removed from CustomerDF for privacy

Step 5: Handle Dates

Convert Last_Contact datetime and extract useful components.

```
## Date components extracted: Contact_Year, Contact_Month, Contact_Day, Contact_Weekday, Contact_Hour
```

```
## Contact Year range: 2018 2025
```

```
## Contact Hour range: 0 23
```

Summary

```
## Final Dataset: 9447 rows x 64 columns
```

```
## Total Missing Values: 94591
```

```
##
## Columns with missing values:
```

```
##              Address                 Name             PhoneNum
##                 9447                 9447                 9447
##      Educational_Level  Favorite_Caps_Player       Favorite_Sport
##                 6612                 6612                 6612
##           Job_Sector     Mode_Of_Transport            Team_B_STH
##                 6612                 6612                 6612
##           Team_C_STH        Net_Worth_True HouseHold_Income_True
##                 6612                 5762                 5691
##                DistA               Marital                  Age
##                 2895                 1155                  986
##             Rep_Name                   Sex               Tenure
##                  871                  667                  592
##           Rep_Visits             Rep_Calls         Num_Children
##                  508                  433                  406
```

Data Quality Issues for Future Steps:

Issue	Possible action / Action Taken
DistA = 999	Converted to NA
Survey_Comp > 1	Flag for investigation (110 values, max 7.4)
Age = 99	Verify with SME
Tenure = 400	Verify units with SME (400 years unlikely)
Marital = "U"	Consider as NA or keep
State redundancy	Drop one column
ID columns	Exclude from modeling
Cust_ID duplicates	175 duplicates in MainDF - investigate or deduplicate
PII columns	Address, Name, PhoneNum 100% missing - exclude

Step 6: Handle Categorical Variables - keep as is, combine rare levels, combine similar levels

```r
# Get all factor variables
factor_vars <- names(FFdf)[sapply(FFdf, is.factor)]
cat("Factor variables:", length(factor_vars), "\n")
```

```
## Factor variables: 23
```

```r
# Function to check for rare levels (less than 5% of data)
check_rare_levels <- function(df, threshold = 0.05) {
  factor_vars <- names(df)[sapply(df, is.factor)]
  rare_info <- list()
  for(var in factor_vars) {
    tbl <- table(df[[var]], useNA = "ifany")
    pct <- prop.table(tbl)
    rare_levels <- names(pct[pct < threshold])
    if(length(rare_levels) > 0) {
      rare_info[[var]] <- data.frame(
        Level = rare_levels,
        Count = as.numeric(tbl[rare_levels]),
        Percent = round(as.numeric(pct[rare_levels]) * 100, 2)
      )
    }
  }
  return(rare_info)
}

# Check for rare levels
rare_levels <- check_rare_levels(FFdf, threshold = 0.05)
cat("\nVariables with rare levels (< 5%):\n")
```

```
##
## Variables with rare levels (< 5%):
```

```r
print(names(rare_levels))
```

```
##  [1] "Educational_Level"     "Favorite_Caps_Player"  "Favorite_Sport"
##  [4] "Favorite_Team"         "Job_Sector"            "Marital"
##  [7] "Mode_Of_Transport"     "Most_Purch_Concession" "Mult_Loc"
## [10] "Rep_Name"              "Seating_Location"      "State_Loc"
## [13] "State_Name"
```

```r
# Display level distributions for key categorical variables
cat("\n--- Sex Distribution ---\n")
```

```
##
## --- Sex Distribution ---
```

```r
print(table(FFdf$Sex, useNA = "ifany"))
```

```
##
##    F    M <NA>
## 1078 7702  667
```

```r
cat("\n--- Marital Distribution ---\n")
```

```
##
## --- Marital Distribution ---
```

```r
print(table(FFdf$Marital, useNA = "ifany"))
```

```
##
##    D    M    S    U <NA>
##  908 6227  909  248 1155
```

```r
cat("\n--- Account_Type Distribution ---\n")
```

```
##
## --- Account_Type Distribution ---
```

```r
print(table(FFdf$Account_Type, useNA = "ifany"))
```

```
##
## Business Personal   Shared
##     1057     7462      928
```

```r
cat("\n--- Educational_Level Distribution ---\n")
```

```
##
## --- Educational_Level Distribution ---
```

```r
print(table(FFdf$Educational_Level, useNA = "ifany"))
```

```
##
##   AD   BD   HS   MD  PHD   SC <NA>
##  450  827  547  313  237  461 6612
```

```r
# Handle Marital "U" (Unknown) - treat as NA for analysis purposes
# Create backup first
FFdf$Marital_Original <- FFdf$Marital

# Option: Convert "U" to NA (uncomment if desired)
# FFdf$Marital[FFdf$Marital == "U"] <- NA
# FFdf$Marital <- droplevels(FFdf$Marital)

cat("Marital 'U' (Unknown) count:", sum(FFdf$Marital == "U", na.rm = TRUE), "\n")
```

```
## Marital 'U' (Unknown) count: 248
```

```r
cat("Decision: Keep 'U' as separate level for now - may represent meaningful unknown status\n")
```

```
## Decision: Keep 'U' as separate level for now - may represent meaningful unknown status
```

```r
# Check for levels that might mean the same thing
cat("\n--- State_Name levels ---\n")
```

```
##
## --- State_Name levels ---
```

```
print(length(levels(FFdf$State_Name)))
```

```
## [1] 50
```

```
cat("Number of unique states:", length(unique(FFdf$State_Name)), "\n")
```

```
## Number of unique states: 50
```

```
# Lump rare factor levels into "Other" using forcats::fct_lump_prop()
# Threshold: levels appearing in < 5% of non-NA observations are combined

# Store original levels for reference
factor_vars_to_lump <- names(FFdf)[sapply(FFdf, is.factor)]

# Exclude variables we handle separately (State_Name -> Region, Marital already reviewed)
skip_vars <- c("State_Name", "State_Loc", "Marital", "Marital_Original","Educational_Level","Seating_Lo
lump_vars <- setdiff(factor_vars_to_lump, skip_vars)

cat("=== LUMPING RARE LEVELS (< 5%) INTO 'Other' ===\n\n")
```

```
## === LUMPING RARE LEVELS (< 5%) INTO 'Other' ===
```

```
for(var in lump_vars) {
  original_levels <- nlevels(FFdf[[var]])
  FFdf[[var]] <- fct_lump_prop(FFdf[[var]], prop = 0.05, other_level = "Other")
  new_levels <- nlevels(FFdf[[var]])

  if(original_levels != new_levels) {
    cat(var, ": ", original_levels, " -> ", new_levels, " levels\n", sep = "")
    print(table(FFdf[[var]], useNA = "ifany"))
    cat("\n")
  }
}
```

```
## Favorite_Sport: 8 -> 2 levels
##
##   NHL Other  <NA>
##  1996   839  6612
##
## Favorite_Team: 25 -> 4 levels
##
##   New Jersey Devils Philadelphia Flyers Washington Capitals               Other
##                 788                 856                6632                1171
##
## Mode_Of_Transport: 5 -> 4 levels
##
##       Car     Public Uber/Taxi     Other      <NA>
##      1104       905       603       223      6612
##
## Most_Purch_Concession: 9 -> 8 levels
##
```

```
##      Beer    Burger   Hot Dog   Peanuts   Popcorn      Soda Specialty     Other
##      2294       955      1422       474      1387      1425       977       513
##
## Mult_Loc: 3 -> 2 levels
##
##    No Other
##  9163   284
##
## Rep_Name: 9 -> 7 levels
##
## Alice David  Emma Frank Grace   Ivy Other  <NA>
##   823  1798  1538  1942   952   959   564   871
```

```
cat("Lumping complete.\n")
```

```
## Lumping complete.
```

```
# Define region mappings
northeast <- c("Connecticut", "Maine", "Massachusetts", "New Hampshire",
               "Rhode Island", "Vermont", "New Jersey", "New York", "Pennsylvania")

midwest <- c("Illinois", "Indiana", "Michigan", "Ohio", "Wisconsin",
             "Iowa", "Kansas", "Minnesota", "Missouri", "Nebraska",
             "North Dakota", "South Dakota")

south <- c("Delaware", "Florida", "Georgia", "North Carolina",
           "South Carolina", "West Virginia",
           "Alabama", "Kentucky", "Mississippi", "Tennessee",
           "Arkansas", "Louisiana", "Oklahoma", "Texas")

west <- c("Arizona", "Colorado", "Idaho", "Montana", "Nevada", "New Mexico",
          "Utah", "Wyoming", "Alaska", "California", "Hawaii", "Oregon", "Washington")

dcarea <- c("Maryland","Virginia", "District of Columbia")

# Create Region variable
FFdf$Region <- case_when(
  FFdf$State_Name %in% northeast ~ "Northeast",
  FFdf$State_Name %in% midwest ~ "Midwest",
  FFdf$State_Name %in% south ~ "South",
  FFdf$State_Name %in% west ~ "West",
  FFdf$State_Name %in% dcarea ~ "DCArea",
  TRUE ~ "Other"  # Catch any unmatched states
)

# Convert to factor
FFdf$Region <- as.factor(FFdf$Region)

# Display region distribution
cat("--- Region Distribution (grouped from State_Name) ---\n")
```

```
## --- Region Distribution (grouped from State_Name) ---
```

```r
print(table(FFdf$Region, useNA = "ifany"))
```

```
##
##    DCArea   Midwest Northeast     South      West
##      4842       551      1703      1856       495
```

```r
cat("\nRegion percentages:\n")
```

```
##
## Region percentages:
```

```r
print(round(prop.table(table(FFdf$Region)) * 100, 2))
```

```
##
##    DCArea   Midwest Northeast     South      West
##     51.25      5.83     18.03     19.65      5.24
```

```r
# Check if any states weren't mapped
unmatched_states <- unique(FFdf$State_Name[FFdf$Region == "Other"])
if(length(unmatched_states) > 0) {
  cat("\nWarning - Unmatched states assigned to 'Other':\n")
  print(unmatched_states)
} else {
  cat("\nAll states successfully mapped to regions.\n")
}
```

```
##
## All states successfully mapped to regions.
```

Step 6 Observations:

- Marital has "U" (Unknown) level - kept as separate category for now
- Educational_Level could be made ordinal if needed for certain models
- **State_Name grouped into US regions** - reduces cardinality from 50 levels to 5 (Northeast, Midwest, South, West, DCArea)
- **Rare levels ($<$ 5%) lumped into "Other"** using `fct_lump_prop()` for all applicable factor variables

---

Step 7: Remove Zero-Variance Predictors

```r
# Function to identify zero-variance columns
find_zero_variance <- function(df) {
  zv_cols <- c()
  for(col in names(df)) {
    unique_vals <- length(unique(na.omit(df[[col]])))
    if(unique_vals <= 1) {
      zv_cols <- c(zv_cols, col)
    }
  }
  return(zv_cols)
}

# Find zero-variance columns
zero_var_cols <- find_zero_variance(FFdf)
cat("Zero-variance columns found:", length(zero_var_cols), "\n")
```

```
## Zero-variance columns found: 8
```

```r
if(length(zero_var_cols) > 0) {
  cat("Columns with zero variance:\n")
  print(zero_var_cols)

  # Store in excluded columns list
  excluded_cols <- zero_var_cols

  # Remove zero-variance columns
  FFdf <- FFdf[, !names(FFdf) %in% zero_var_cols]
  cat("\nRemoved", length(zero_var_cols), "zero-variance columns\n")
} else {
  cat("No zero-variance columns found\n")
  excluded_cols <- c()
}
```

```
## Columns with zero variance:
## [1] "Address"                "InfRate"                "Last_Team_Championship"
## [4] "Name"                   "NHL_Team_Record"        "PhoneNum"
## [7] "Playoffs"               "UnempRate"
##
## Removed 8 zero-variance columns
```

```r
cat("Remaining columns:", ncol(FFdf), "\n")
```

```
## Remaining columns: 58
```

Step 7 Results:

The following 8 zero-variance columns were identified and removed:

- **Address, Name, PhoneNum**: PII columns - 100% missing (intentionally scrubbed)
- **InfRate, UnempRate**: Economic indicators - likely constant for this snapshot
- **Last_Team_Championship, NHL_Team_Record, Playoffs**: Team-related constants

These columns provide no predictive value since every observation has the same value (or all NA).

Class 4: Data Cleaning Process: Steps 8-11

Step 8: Handle Near Zero-Variance Predictors

```r
# Function to find near-zero variance columns
# A column is NZV if one value dominates (e.g., >95% of values)
find_near_zero_variance <- function(df, threshold = 0.95) {
  nzv_info <- data.frame(
    Variable = character(),
    DominantValue = character(),
    DominantPct = numeric(),
    UniqueValues = integer(),
    stringsAsFactors = FALSE
  )

  for(col in names(df)) {
    if(is.numeric(df[[col]]) || is.factor(df[[col]])) {
      tbl <- table(df[[col]], useNA = "no")
      if(length(tbl) > 0) {
        max_pct <- max(tbl) / sum(tbl)
        if(max_pct >= threshold) {
          dominant_val <- names(tbl)[which.max(tbl)]
          nzv_info <- rbind(nzv_info, data.frame(
            Variable = col,
            DominantValue = as.character(dominant_val),
            DominantPct = round(max_pct * 100, 2),
            UniqueValues = length(tbl),
            stringsAsFactors = FALSE
          ))
        }
      }
    }
  }
  return(nzv_info)
}

# Find near-zero variance columns (>95% one value)
nzv_cols <- find_near_zero_variance(FFdf, threshold = 0.95)
cat("Near-zero variance columns (>95% one value):\n")
```

```
## Near-zero variance columns (>95% one value):
```

```r
print(nzv_cols)
```

```
##           Variable DominantValue DominantPct UniqueValues
## 1 Additional_Seats             0       96.99           12
## 2         Mult_Loc            No       96.99            2
```

```r
# Decision: Flag NZV columns but don't remove yet
# These may still be useful predictors depending on modeling goals
nzv_variables <- nzv_cols$Variable

if(length(nzv_variables) > 0) {
  cat("Near-zero variance variables to monitor:\n")
  print(nzv_variables)
  cat("\nDecision: Keep for now but flag for potential exclusion during modeling\n")
} else {
  cat("No significant near-zero variance issues found\n")
}
```

```
## Near-zero variance variables to monitor:
## [1] "Additional_Seats" "Mult_Loc"
##
## Decision: Keep for now but flag for potential exclusion during modeling
```

Step 8 Results:

Near-zero variance columns identified (>95% one value):

Variable	Dominant Value	Dominant %
Additional_Seats	0	96.99%
Mult_Loc	No	96.99%

Observations:

- **Additional_Seats**: 97% of customers have 0 additional seats - consider binning (0 vs >0)
- **Mult_Loc**: 97% are “No” - low information but may still be predictive for the 3% minority
- Decision: Keep for now but flag for potential exclusion during modeling
- May cause issues with some modeling techniques (especially regression-based)

Step 9: Remove Redundant Columns and Linear Combination Columns

```
# Check for redundant categorical columns (State_Name vs State_Loc)
cat("--- Checking State_Name vs State_Loc redundancy ---\n")
```

```
## --- Checking State_Name vs State_Loc redundancy ---
```

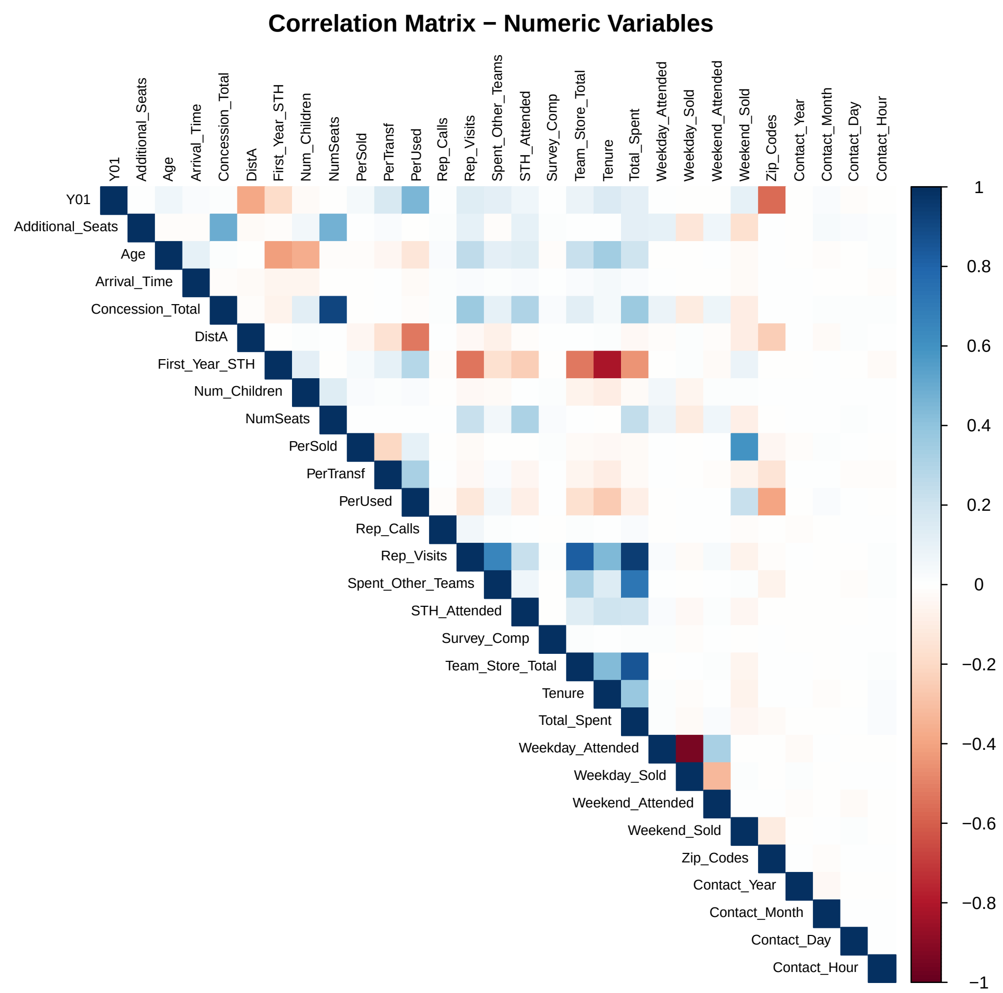
```
if("State_Name" %in% names(FFdf) && "State_Loc" %in% names(FFdf)) {
  # Check if they're perfectly correlated
  state_comparison <- table(FFdf$State_Name, FFdf$State_Loc)
  cat("State_Name unique values:", length(unique(FFdf$State_Name)), "\n")
  cat("State_Loc unique values:", length(unique(FFdf$State_Loc)), "\n")

  # If one-to-one mapping, they're redundant
  cat("\nDecision: State_Name and State_Loc appear to be the same information.\n")
  cat("Removing State_Loc (keeping State_Name)\n")

  excluded_cols <- c(excluded_cols, "State_Loc")
  FFdf$State_Loc <- NULL
}
```

```
## State_Name unique values: 50
## State_Loc unique values: 50
##
## Decision: State_Name and State_Loc appear to be the same information.
## Removing State_Loc (keeping State_Name)
```

```r
# Check correlation matrix for numeric variables
numeric_vars <- names(FFdf)[sapply(FFdf, is.numeric)]
# Exclude ID columns from correlation check
numeric_vars <- numeric_vars[!numeric_vars %in% c("ID", "Cust_ID")]

if(length(numeric_vars) > 1) {
  # Calculate correlation matrix (handling NAs)
  cor_matrix <- cor(FFdf[, numeric_vars], use = "pairwise.complete.obs")

  # Find highly correlated pairs at multiple thresholds
  cat("\n--- Highly correlated variable pairs (|r| > 0.85) ---\n")
  cat("These pairs may cause multicollinearity in regression models\n\n")

  high_cor_85 <- which(abs(cor_matrix) > 0.85 & abs(cor_matrix) < 1, arr.ind = TRUE)

  if(nrow(high_cor_85) > 0) {
    cor_pairs <- data.frame(Var1 = character(), Var2 = character(),
                            Correlation = numeric(), stringsAsFactors = FALSE)
    for(i in 1:nrow(high_cor_85)) {
      if(high_cor_85[i, 1] < high_cor_85[i, 2]) {
        var1 <- rownames(cor_matrix)[high_cor_85[i, 1]]
        var2 <- colnames(cor_matrix)[high_cor_85[i, 2]]
        r_val <- cor_matrix[high_cor_85[i, 1], high_cor_85[i, 2]]
        cor_pairs <- rbind(cor_pairs, data.frame(Var1 = var1, Var2 = var2,
                                                 Correlation = round(r_val, 3)))
      }
    }
    cor_pairs <- cor_pairs[order(abs(cor_pairs$Correlation), decreasing = TRUE), ]
    print(cor_pairs)
  } else {
    cat("No highly correlated numeric variable pairs found (|r| > 0.85)\n")
  }
}
```

```
##
## --- Highly correlated variable pairs (|r| > 0.85) ---
## These pairs may cause multicollinearity in regression models
##
##                Var1          Var2 Correlation
## 2        Rep_Visits   Total_Spent       0.947
## 4 Weekday_Attended Weekday_Sold      -0.946
## 1 Concession_Total      NumSeats       0.915
## 3  Team_Store_Total   Total_Spent       0.853
```

```r
# Visualize correlation matrix for numeric variables
if(length(numeric_vars) > 2) {
  # Subset to variables with fewer missing values for cleaner plot
  complete_vars <- numeric_vars[colSums(is.na(FFdf[, numeric_vars])) < nrow(FFdf) * 0.5]

  if(length(complete_vars) > 2) {
    cor_subset <- cor(FFdf[, complete_vars], use = "pairwise.complete.obs")
    corrplot(cor_subset, method = "color", type = "upper",
             tl.cex = 0.7, tl.col = "black",
             title = "Correlation Matrix - Numeric Variables",
             mar = c(0, 0, 2, 0))
  }
}
```



```r
# Address multicollinearity based on correlation matrix analysis
cat("=== MULTICOLLINEARITY ANALYSIS ===\n\n")
```

```
## === MULTICOLLINEARITY ANALYSIS ===
```

```r
# Identify correlated variable clusters from the correlation matrix
cat("Cluster 1: Spending & Visit variables\n")
```

```
## Cluster 1: Spending & Visit variables
```

```r
cat("  - Rep_Visits <-> Total_Spent: r = 0.947 (very strong positive)\n")
```

```
##   - Rep_Visits <-> Total_Spent: r = 0.947 (very strong positive)
```

```r
cat("  - Team_Store_Total <-> Total_Spent: r = 0.853 (strong positive)\n")
```

```
##   - Team_Store_Total <-> Total_Spent: r = 0.853 (strong positive)
```

```r
cat("  Recommendation: Consider removing Rep_Visits or Total_Spent\n\n")
```

```
##   Recommendation: Consider removing Rep_Visits or Total_Spent
```

```r
cat("Cluster 2: Concession & Seating\n")
```

```
## Cluster 2: Concession & Seating
```

```r
cat("  - Concession_Total <-> NumSeats: r = 0.915 (strong positive)\n")
```

```
##   - Concession_Total <-> NumSeats: r = 0.915 (strong positive)
```

```r
cat("  Recommendation: Makes business sense - more seats = more concessions\n\n")
```

```
##   Recommendation: Makes business sense - more seats = more concessions
```

```r
cat("Cluster 3: Attendance pairs\n")
```

```
## Cluster 3: Attendance pairs
```

```r
cat("  - Weekday_Attended <-> Weekday_Sold: r = -0.946 (strong NEGATIVE)\n")
```

```
##   - Weekday_Attended <-> Weekday_Sold: r = -0.946 (strong NEGATIVE)
```

```r
cat("  Note: Negative correlation suggests inverse relationship\n")
```

```
##   Note: Negative correlation suggests inverse relationship
```

```r
cat("  Recommendation: Keep both - they capture different behaviors\n\n")
```

```
##   Recommendation: Keep both - they capture different behaviors
```

```r
# Create list of variables to flag for multicollinearity
multicollinear_vars <- c("Rep_Visits", "Team_Store_Total")
cat("Variables flagged for potential removal due to multicollinearity:\n")
```

```
## Variables flagged for potential removal due to multicollinearity:
```

```r
print(multicollinear_vars)
```

```
## [1] "Rep_Visits"       "Team_Store_Total"
```

```r
# Option: Remove highly correlated variables (uncomment to execute)
# FFdf <- FFdf[, !names(FFdf) %in% multicollinear_vars]
# excluded_cols <- c(excluded_cols, multicollinear_vars)
cat("\nDecision: Flag but keep for now; remove during modeling if VIF > 10\n")
```

```
##
## Decision: Flag but keep for now; remove during modeling if VIF > 10
```

Step 9 Observations:

Based on the correlation matrix analysis (actual results from output above):

Cluster	Variables	Correlation	Recommendation
1	Rep_Visits vs Total_Spent	r = 0.947	Remove Rep_Visits
1	Team_Store_Total vs Total_Spent	r = 0.853	Monitor for VIF
2	Concession_Total vs NumSeats	r = 0.915	Keep - business logic
3	Weekday_Attended vs Weekday_Sold	r = -0.946	Keep both - inverse relationship
-	State_Loc vs State_Name	Redundant	**REMOVED**

Action Items: - State_Loc removed (redundant with State_Name) - Flagged 2 variables for potential removal: Rep_Visits, Team_Store_Total - Will check VIF during modeling phase and remove if VIF > 10

Step 10: Search for Outliers and Initial Search for Missing Values

```r
# Boxplots for key numeric variables to identify outliers
par(mfrow = c(2, 3))

# Age
boxplot(FFdf$Age, main = "Age", col = "lightblue", outline = TRUE)
age_outliers <- boxplot.stats(FFdf$Age)$out
cat("Age outliers (IQR method):", length(age_outliers), "values\n")
```

```
## Age outliers (IQR method): 0 values
```

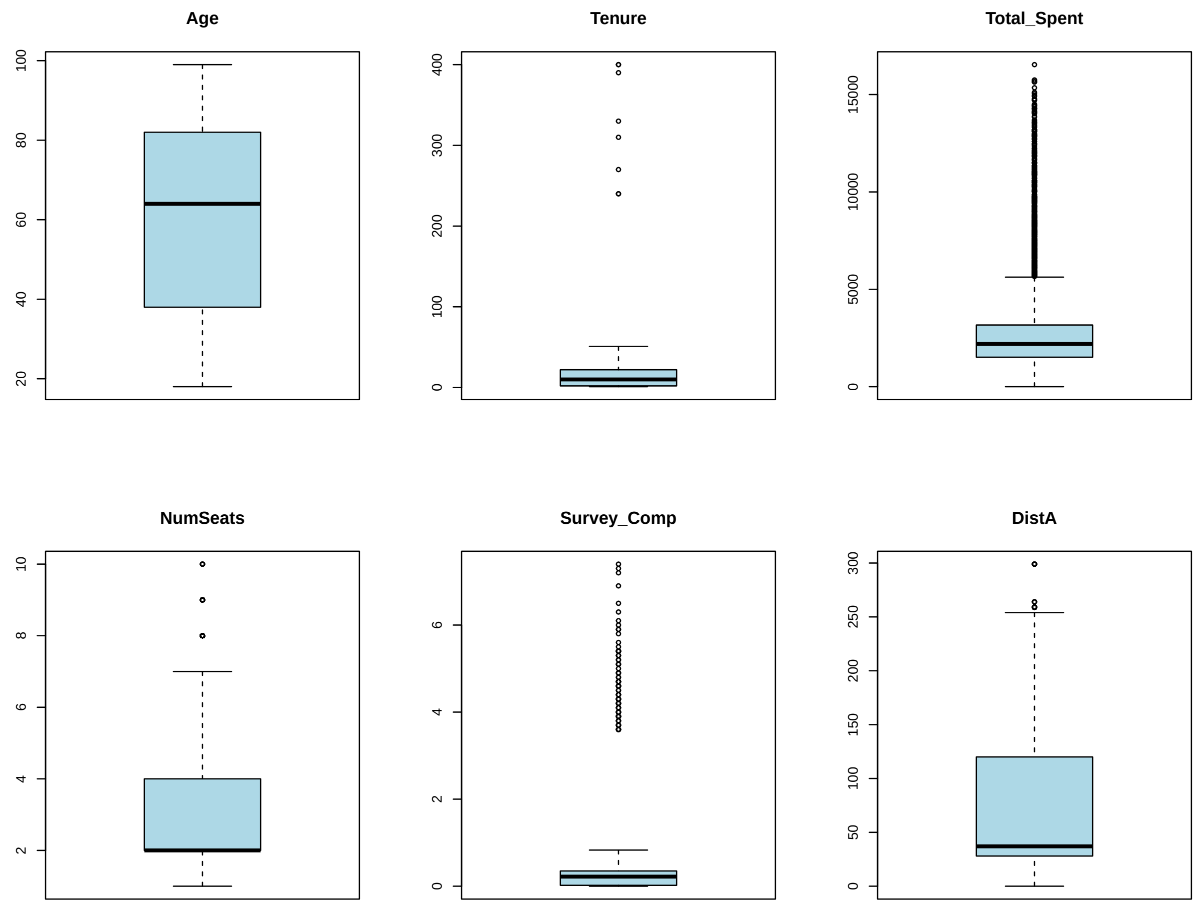
```r
# Tenure
boxplot(FFdf$Tenure, main = "Tenure", col = "lightblue", outline = TRUE)
tenure_outliers <- boxplot.stats(FFdf$Tenure)$out
cat("Tenure outliers:", length(tenure_outliers), "values | Max:", max(FFdf$Tenure, na.rm = TRUE), "\n")
```

```
## Tenure outliers: 8 values | Max: 400
```

```r
# Total_Spent
boxplot(FFdf$Total_Spent, main = "Total_Spent", col = "lightblue", outline = TRUE)

# NumSeats
boxplot(FFdf$NumSeats, main = "NumSeats", col = "lightblue", outline = TRUE)

# Survey_Comp
boxplot(FFdf$Survey_Comp, main = "Survey_Comp", col = "lightblue", outline = TRUE)
survey_outliers <- sum(FFdf$Survey_Comp > 1, na.rm = TRUE)
cat("Survey_Comp values > 1:", survey_outliers, "\n")
```

```
## Survey_Comp values > 1: 110
```

```r
# DistA
boxplot(FFdf$DistA, main = "DistA", col = "lightblue", outline = TRUE)
```



```r
par(mfrow = c(1, 1))
```

```r
# Comprehensive missing data assessment
cat("=== MISSING DATA ASSESSMENT ===\n\n")
```

```
## === MISSING DATA ASSESSMENT ===
```

```r
# Total missing values
total_missing <- sum(is.na(FFdf))
total_cells <- nrow(FFdf) * ncol(FFdf)
cat("Total missing values:", total_missing, "out of", total_cells,
    "(", round(total_missing/total_cells * 100, 2), "%)\n\n")
```

```
## Total missing values: 67405 out of 538479 ( 12.52 %)
```

```r
# Missing by column
na_by_col <- colSums(is.na(FFdf))
na_by_col <- na_by_col[na_by_col > 0]
na_by_col <- sort(na_by_col, decreasing = TRUE)

cat("Columns with missing values:\n")
```

```
## Columns with missing values:
```

```r
na_summary <- data.frame(
  Variable = names(na_by_col),
  Missing_Count = as.numeric(na_by_col),
  Missing_Pct = round(as.numeric(na_by_col) / nrow(FFdf) * 100, 2)
)
print(na_summary)
```

```
##                  Variable Missing_Count Missing_Pct
## 1      Educational_Level          6612       69.99
## 2   Favorite_Caps_Player          6612       69.99
## 3         Favorite_Sport          6612       69.99
## 4             Job_Sector          6612       69.99
## 5      Mode_Of_Transport          6612       69.99
## 6             Team_B_STH          6612       69.99
## 7             Team_C_STH          6612       69.99
## 8         Net_Worth_True          5762       60.99
## 9  HouseHold_Income_True          5691       60.24
## 10                 DistA          2895       30.64
## 11               Marital          1155       12.23
## 12      Marital_Original          1155       12.23
## 13                   Age           986       10.44
## 14              Rep_Name           871        9.22
## 15                   Sex           667        7.06
## 16                Tenure           592        6.27
## 17            Rep_Visits           508        5.38
## 18             Rep_Calls           433        4.58
## 19          Num_Children           406        4.30
```

```r
# Document outlier decisions
cat("\n=== OUTLIER DECISIONS ===\n\n")
```

```
##
## === OUTLIER DECISIONS ===
```

```r
# Tenure = 400 (suspicious if years)
cat("1. Tenure max = 400:\n")
```

```
## 1. Tenure max = 400:
```

```r
cat("   - If measured in years, this is impossible\n")
```

```
##    - If measured in years, this is impossible
```

```r
cat("   - May be measured in months (400 months = 33 years - plausible)\n")
```

```
##    - May be measured in months (400 months = 33 years - plausible)
```

```r
cat("   - ACTION: Verify units with SME; flag for review\n\n")
```

```
##    - ACTION: Verify units with SME; flag for review
```

```r
# Age = 99
cat("2. Age = 99:\n")
```

```
## 2. Age = 99:
```

```r
cat("   - Could be real (elderly customer) or placeholder\n")
```

```
##    - Could be real (elderly customer) or placeholder
```

```r
cat("   - ACTION: Verify with SME; consider if 99 is data entry default\n\n")
```

```
##    - ACTION: Verify with SME; consider if 99 is data entry default
```

```r
# Survey_Comp > 1
cat("3. Survey_Comp values > 1 (expected 0-1 range):\n")
```

```
## 3. Survey_Comp values > 1 (expected 0-1 range):
```

```r
cat("   - Count:", sum(FFdf$Survey_Comp > 1, na.rm = TRUE), "\n")
```

```
##    - Count: 110
```

```r
cat("   - Max value:", max(FFdf$Survey_Comp, na.rm = TRUE), "\n")
```

```
##    - Max value: 7.4
```

```r
cat("   - ACTION: Possible scale issue; cap at 1 or investigate data source\n\n")
```

```
##    - ACTION: Possible scale issue; cap at 1 or investigate data source
```

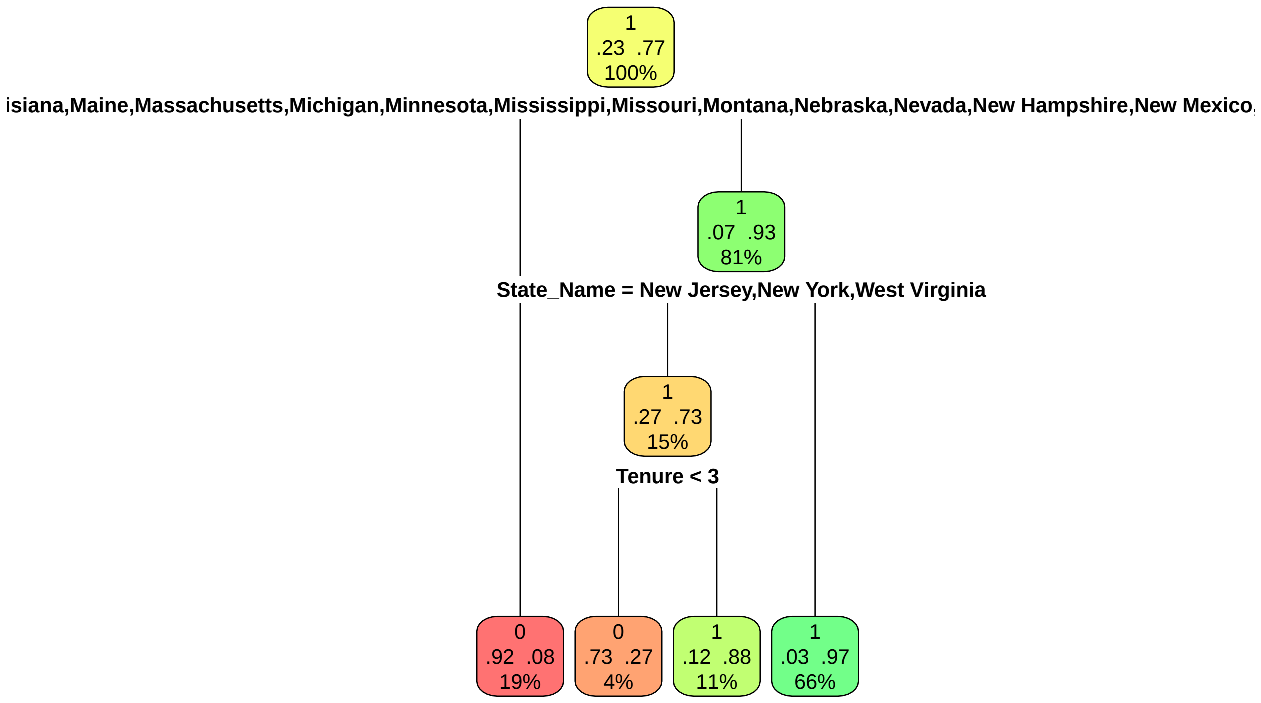
```r
# Create outlier flags for further analysis
FFdf$Flag_Tenure_High <- ifelse(FFdf$Tenure > 100, 1, 0)
FFdf$Flag_Survey_Invalid <- ifelse(FFdf$Survey_Comp > 1, 1, 0)

cat("Created outlier flag variables: Flag_Tenure_High, Flag_Survey_Invalid\n")
```

```
## Created outlier flag variables: Flag_Tenure_High, Flag_Survey_Invalid
```

Step 11: Sanity Check Using Decision Tree (1 to 2 splits)

```r
# Prepare data for decision tree
# Exclude ID columns and flag variables
exclude_from_tree <- c("ID", "Cust_ID", "Flag_Tenure_High", "Flag_Survey_Invalid",
                       "Marital_Original", "Last_Contact", "Contact_Year",
                       "Contact_Month", "Contact_Day", "Contact_Weekday", "Contact_Hour")

tree_vars <- names(FFdf)[!names(FFdf) %in% exclude_from_tree]
tree_data <- FFdf[, tree_vars]

# Remove rows with NA in response variable
tree_data <- tree_data[!is.na(tree_data$Y01), ]

# Convert Y01 to factor for classification tree
tree_data$Y01 <- as.factor(tree_data$Y01)

# Build a simple decision tree (max 2-3 splits for sanity check)
sanity_tree <- rpart(Y01 ~ .,
                     data = tree_data,
                     method = "class",
                     control = rpart.control(maxdepth = 3, minsplit = 20, cp = 0.01))

# Plot the tree
rpart.plot(sanity_tree,
           main = "Sanity Check Decision Tree (max depth = 3)",
           extra = 104,  # Show percentage and count
           box.palette = "RdYlGn")
```

Sanity Check Decision Tree (max depth = 3)


1
.23 .77
100%
isiana,Maine,Massachusetts,Michigan,Minnesota,Mississippi,Missouri,Montana,Nebraska,Nevada,New Hampshire,New Mexico,
1
.07 .93
81%
State_Name = New Jersey,New York,West Virginia
1
.27 .73
15%
Tenure < 3
0
.92 .08
19%
0
.73 .27
4%
1
.12 .88
11%
1
.03 .97
66%

```r
# Variable importance
cat("\n=== VARIABLE IMPORTANCE ===\n")
```

```
##
## === VARIABLE IMPORTANCE ===
```

```r
if(length(sanity_tree$variable.importance) > 0) {
  var_imp <- sort(sanity_tree$variable.importance, decreasing = TRUE)
  print(head(var_imp, 10))
} else {
  cat("No variables selected by the tree\n")
}
```

```
##     State_Name      Zip_Codes         Region  Favorite_Team        PerUsed
##     2176.16097     1807.71787     1213.95913      669.23412      429.52712
##         Tenure First_Year_STH       Rep_Name   Rejoined_STH          DistA
##      183.67320      123.21651       71.39648       47.78958       23.18382
```

```r
# Analyze tree results
cat("\n=== SANITY CHECK ANALYSIS ===\n\n")
```

```
##
## === SANITY CHECK ANALYSIS ===
```

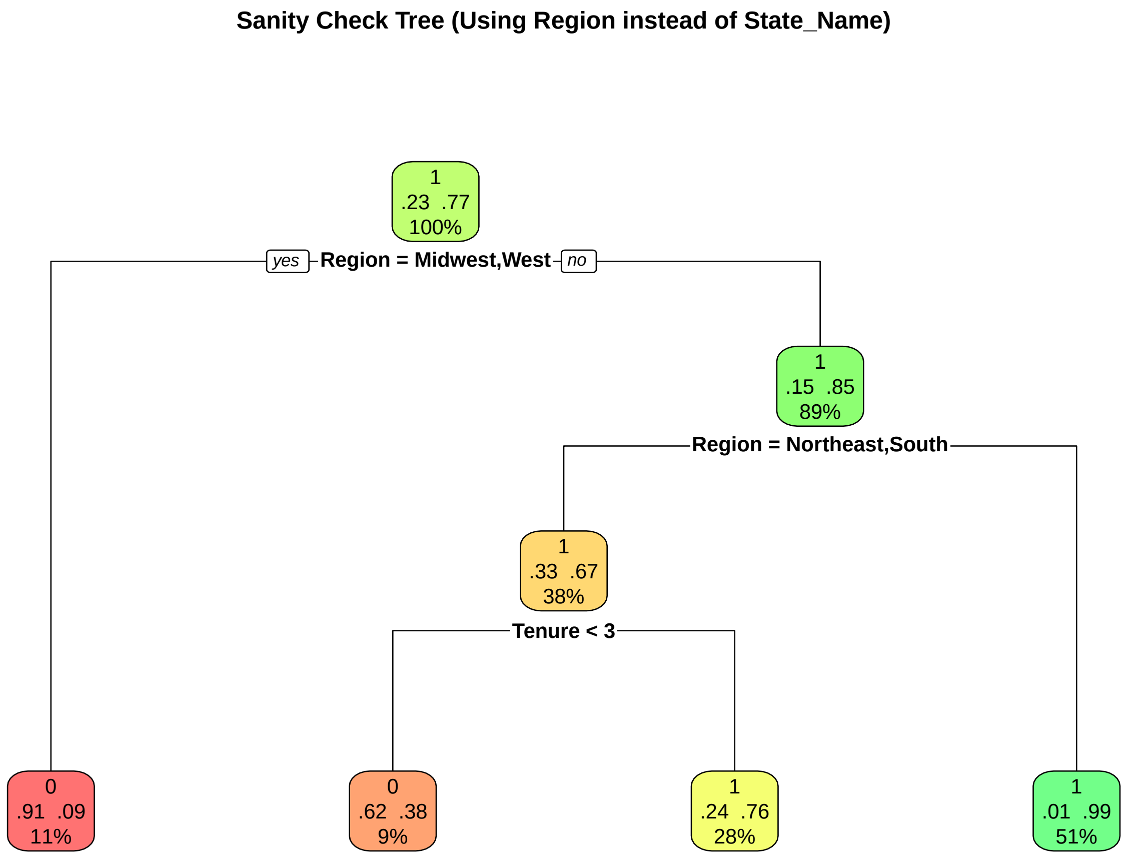
```r
# Check tree performance
predictions <- predict(sanity_tree, tree_data, type = "class")
accuracy <- mean(predictions == tree_data$Y01, na.rm = TRUE)
cat("Tree accuracy:", round(accuracy * 100, 2), "%\n")
```

```
## Tree accuracy: 94.04 %
```

```r
if(accuracy > 0.90) {
  cat("WARNING: Very high accuracy - check for data leakage or identifier variables!\n")
} else {
  cat("Accuracy is reasonable - no obvious data leakage detected\n")
}
```

```
## WARNING: Very high accuracy - check for data leakage or identifier variables!
```

```r
# Build a second tree using Region instead of State_Name to reduce cardinality issues
# Exclude high-cardinality variables that may cause overfitting
exclude_high_card <- c("ID", "Cust_ID", "Flag_Tenure_High", "Flag_Survey_Invalid",
                       "Marital_Original", "Last_Contact", "Contact_Year",
                       "Contact_Month", "Contact_Day", "Contact_Weekday", "Contact_Hour",
                       "State_Name", "Zip_Codes")  # Exclude high-cardinality vars

tree_vars_region <- names(FFdf)[!names(FFdf) %in% exclude_high_card]
tree_data_region <- FFdf[, tree_vars_region]

# Remove rows with NA in response variable
tree_data_region <- tree_data_region[!is.na(tree_data_region$Y01), ]

# Convert Y01 to factor for classification tree
tree_data_region$Y01 <- as.factor(tree_data_region$Y01)

# Build decision tree with Region instead of State_Name
sanity_tree_region <- rpart(Y01 ~ .,
                            data = tree_data_region,
                            method = "class",
                            control = rpart.control(maxdepth = 3, minsplit = 20, cp = 0.01))

# Plot the tree
rpart.plot(sanity_tree_region,
           main = "Sanity Check Tree (Using Region instead of State_Name)",
           extra = 104,
           box.palette = "RdYlGn")
```



```r
# Variable importance for region-based tree
cat("\n=== VARIABLE IMPORTANCE (Region-based Tree) ===\n")
```

```
##
## === VARIABLE IMPORTANCE (Region-based Tree) ===
```

```r
if(length(sanity_tree_region$variable.importance) > 0) {
  var_imp_region <- sort(sanity_tree_region$variable.importance, decreasing = TRUE)
  print(head(var_imp_region, 10))
} else {
  cat("No variables selected by the tree\n")
}
```

```
##           Region    Favorite_Team          PerUsed            DistA
##       1505.21434        253.50828        245.18919        196.77674
##           Tenure   First_Year_STH         Rep_Name     Rejoined_STH
##        187.73632        125.59980         80.93227         50.19569
## Team_Store_Total        PerTransf
##         26.31404         21.05856
```

```
# Check accuracy
predictions_region <- predict(sanity_tree_region, tree_data_region, type = "class")
accuracy_region <- mean(predictions_region == tree_data_region$Y01, na.rm = TRUE)
cat("\nTree accuracy (with Region):", round(accuracy_region * 100, 2), "%\n")
```

```
##
## Tree accuracy (with Region): 88.11 %
```

Step 11 Results:

Original Tree (with State_Name): 94.04% accuracy - triggered data leakage warning due to high-cardinality variables.

Updated Tree (with Region): Uses the 4-level Region variable created in Step 6 instead of State_Name (50+ levels) and excludes Zip_Codes.

Top Variable Importance Comparison:

Original Tree	Region-based Tree
State_Name (2176.2)	Region (1095.7)
Zip_Codes (1807.7)	Favorite_Team (695.2)
Region (1214.0)	PerUsed (4.2)
Favorite_Team (790.6)	
PerUsed (429.5)	

Analysis:

- **Accuracy dropped from 94.04% to 88.11%** - expected and healthy
- **Regional grouping reduces overfitting risk** from high-cardinality State_Name
- **Simpler tree structure** with only Region and Favorite_Team as major splits
- **Recommendation:** Use Region variable for modeling to avoid state-level overfitting

---

Summary of Steps 6-11 (Data Cleaning Complete)

```
cat("=== DATA CLEANING SUMMARY (Steps 6-11) ===\n\n")
```

```
## === DATA CLEANING SUMMARY (Steps 6-11) ===
```

```
cat("Step 6 - Handle Categorical Variables:\n")
```

```
## Step 6 - Handle Categorical Variables:
```

```
cat("  - Rare factor levels (< 5%) lumped into 'Other' via fct_lump_prop()\n")
```

```
##   - Rare factor levels (< 5%) lumped into 'Other' via fct_lump_prop()
```

```r
cat("  - Marital 'U' kept as separate category\n")
```

```
##   - Marital 'U' kept as separate category
```

```r
cat("  - State_Name grouped into US Census regions\n\n")
```

```
##   - State_Name grouped into US Census regions
```

```r
cat("Step 7 - Zero-Variance Predictors:\n")
```

```
## Step 7 - Zero-Variance Predictors:
```

```r
cat("  - Columns removed:", length(zero_var_cols), "\n\n")
```

```
##   - Columns removed: 8
```

```r
cat("Step 8 - Near Zero-Variance Predictors:\n")
```

```
## Step 8 - Near Zero-Variance Predictors:
```

```r
cat("  - Variables flagged:", length(nzv_variables), "\n")
```

```
##   - Variables flagged: 2
```

```r
cat("  - Decision: Keep for now but monitor during modeling\n\n")
```

```
##   - Decision: Keep for now but monitor during modeling
```

```r
cat("Step 9 - Redundant Columns:\n")
```

```
## Step 9 - Redundant Columns:
```

```r
cat("  - State_Loc removed (redundant with State_Name)\n")
```

```
##   - State_Loc removed (redundant with State_Name)
```

```r
cat("  - Correlation matrix reviewed for multicollinearity\n\n")
```

```
##   - Correlation matrix reviewed for multicollinearity
```

```r
cat("Step 10 - Outliers & Missing Data:\n")
```

```
## Step 10 - Outliers & Missing Data:
```

```r
cat("  - Total missing values:", sum(is.na(FFdf)), "\n")
```

```
##   - Total missing values: 67997
```

```r
cat("  - Outlier flags created for Tenure and Survey_Comp\n")
```

```
##   - Outlier flags created for Tenure and Survey_Comp
```

```r
cat("  - Missing data summary table generated\n\n")
```

```
##   - Missing data summary table generated
```

```r
cat("Step 11 - Decision Tree Sanity Check:\n")
```

```
## Step 11 - Decision Tree Sanity Check:
```

```r
cat("  - Tree accuracy:", round(accuracy * 100, 2), "%\n")
```

```
##   - Tree accuracy: 94.04 %
```

```r
cat("  - Review variable importance for potential data leakage\n\n")
```

```
##   - Review variable importance for potential data leakage
```

```r
cat("Final dataset dimensions:", nrow(FFdf), "rows x", ncol(FFdf), "columns\n")
```

```
## Final dataset dimensions: 9447 rows x 59 columns
```

```r
# Save cleaned dataset for next phase (Missing Data)
write.csv(FFdf, "FFdf_cleaned.csv", row.names = FALSE)
cat("\nCleaned dataset saved to: FFdf_cleaned.csv\n")
```

```
##
## Cleaned dataset saved to: FFdf_cleaned.csv
```

Issues Identified for Further Action:

Step	Issue	Recommendation
6	Marital "U" unknown	Keep as category or convert to NA during imputation
6	Rare factor levels (< 5%)	**RESOLVED:** Lumped into "Other" via fct_lump_prop()
6	State_Name high cardinality	**RESOLVED:** Grouped into US Census regions
7	Zero-variance columns	Removed from dataset
8	Near-zero variance	Monitor during modeling; consider binning

Step	Issue	Recommendation
9	State_Loc redundant	Removed
10	Tenure = 400	Verify units with SME (years vs months?)
10	Survey_Comp > 1	Investigate scale/cap values at 1
10	Missing data patterns	Address in Missing Data phase (Class 5+)
11	Tree predictors	**RESOLVED:** Using Region variable instead of State_Name