Data Wrangling and Cleaning: Steps 1-5

David Mullaly

2026-01-30

Introduction

This document covers the data wrangling process and the first five steps of data cleaning for the Missing Data course. We load multiple relational tables, merge them into a single flat file, and perform initial cleaning steps.

Data Cleaning Steps Covered:

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

Data Loading and Wrangling

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

```
# Load the raw data files
MainDF <- read.csv("Raw.csv")
StoreDF <- read.csv("StoreTable.csv")
ConcessDF <- read.csv("ConcessTable.csv")
CustomerDF <- read.csv("CustomerTable.csv")

# Check dimensions of each file
cat("MainDF:", dim(MainDF), "| StoreDF:", dim(StoreDF),
    "| ConcessDF:", dim(ConcessDF), "| CustomerDF:", dim(CustomerDF), "\n")
```

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

```
# Check for duplicate Cust_IDs in each source table BEFORE merging
cat("Duplicates in MainDF:", sum(duplicated(MainDF$Cust_ID)), "\n")
```

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

```
cat("Duplicates in StoreDF:", sum(duplicated(StoreDF$Cust_ID)), "\n")
```

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

```r
cat("Duplicates in ConcessDF:", sum(duplicated(ConcessDF$Cust_ID)), "\n")
```

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

```r
cat("Duplicates in CustomerDF:", sum(duplicated(CustomerDF$Cust_ID)), "\n")
```

```
## Duplicates in CustomerDF: 0
```

```r
# Create flat file by merging all tables on Cust_ID
FFdf <- MainDF
FFdf <- merge(FFdf, StoreDF, by = "Cust_ID", all.x = TRUE)
FFdf <- merge(FFdf, ConcessDF, by = "Cust_ID", all.x = TRUE)
FFdf <- merge(FFdf, CustomerDF, by = "Cust_ID", all.x = TRUE)

cat("Final flat file dimensions - Rows:", nrow(FFdf), "Columns:", ncol(FFdf), "\n")
```

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

```r
cat("Unique Cust_ID:", length(unique(FFdf$Cust_ID)), "| Duplicate rows:", nrow(FFdf) - length(unique(FF
```

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

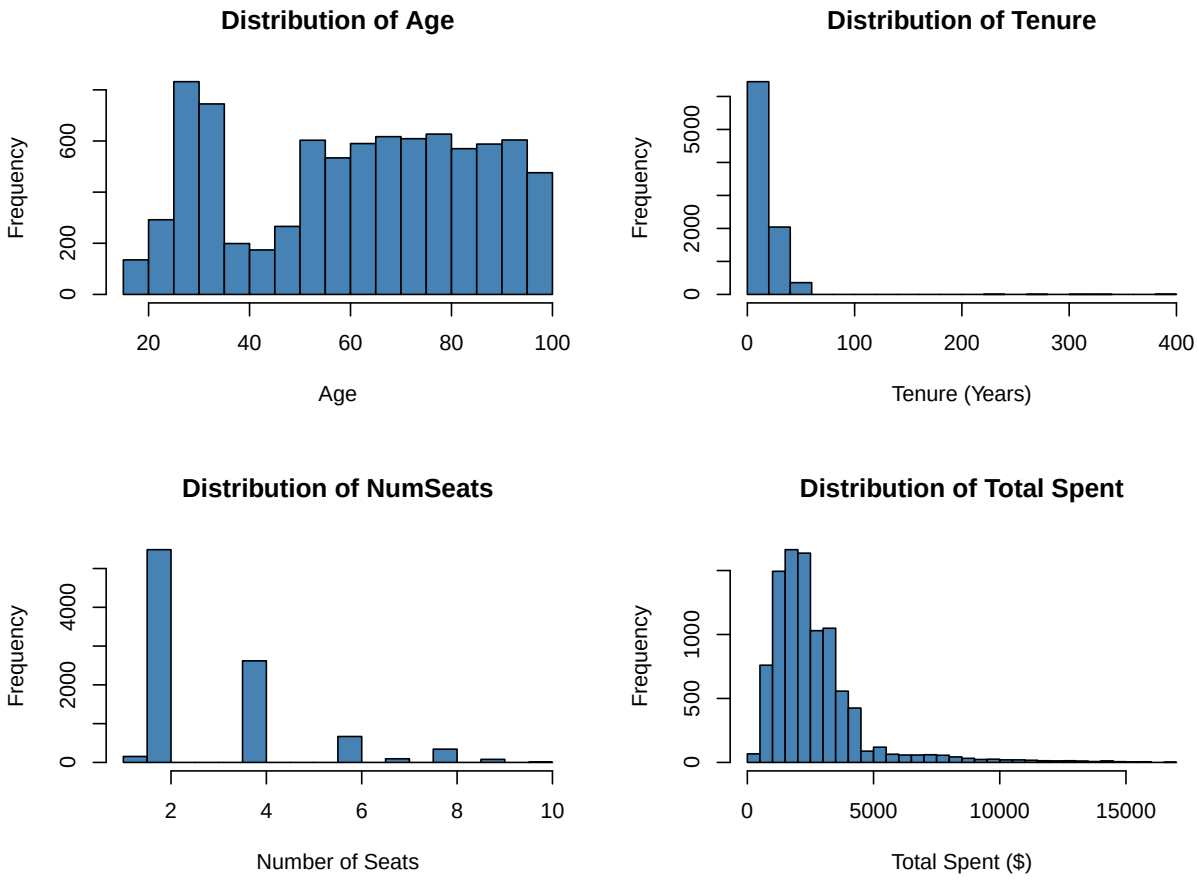
Step 1: Open Data in Your Software of Choice

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

```r
# Create an identifier variable
FFdf$ID <- 1:nrow(FFdf)

# Reorder columns: Y-variable first, then alphabetically
y_var <- "Y01"
other_vars <- sort(setdiff(names(FFdf), y_var))
FFdf <- FFdf[, c(y_var, other_vars)]

# Histograms for key numeric variables
par(mfrow = c(2, 2))
hist(FFdf$Age, main = "Distribution of Age", xlab = "Age", col = "steelblue", breaks = 20)
hist(FFdf$tenure, main = "Distribution of Tenure", xlab = "Tenure (Years)", col = "steelblue", breaks =
hist(FFdf$NumSeats, main = "Distribution of NumSeats", xlab = "Number of Seats", col = "steelblue")
hist(FFdf$Total.Spent, main = "Distribution of Total Spent", xlab = "Total Spent ($)", col = "steelblue
```

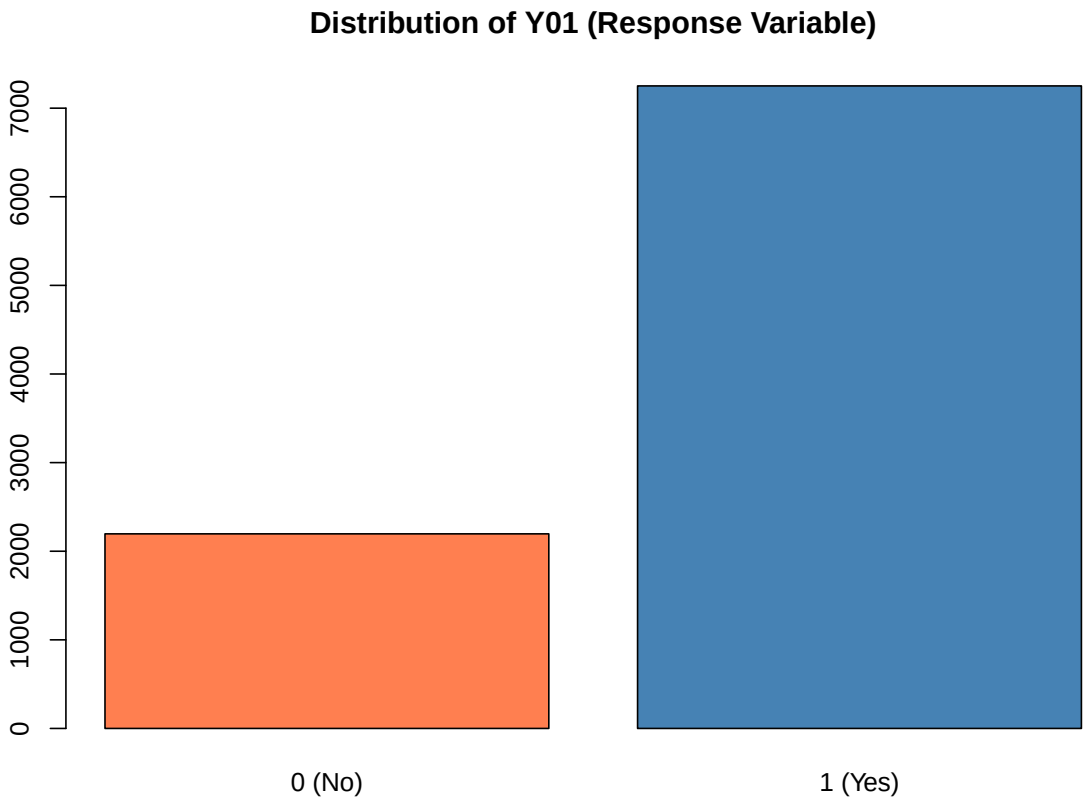



```r
par(mfrow = c(1, 1))

# Response variable distribution
cat("Response Variable (Y01):", table(FFdf$Y01), "\n")
```

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

```r
barplot(table(FFdf$Y01), main = "Distribution of Y01 (Response Variable)",
        col = c("coral", "steelblue"), names.arg = c("0 (No)", "1 (Yes)"))
```



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.

```
# Clean up variable names - replace dots/spaces with underscores, capitalize
names(FFdf) <- gsub("\\.", "_", names(FFdf))
names(FFdf) <- gsub(" ", "_", names(FFdf))
names(FFdf) <- gsub("(^|_)([a-z])", "\\1\\U\\2", names(FFdf), perl = TRUE)

cat("Dataset:", nrow(FFdf), "rows,", ncol(FFdf), "columns\n")
```

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

```
cat("Unique Cust_ID:", length(unique(FFdf$Cust_ID)), "out of", nrow(FFdf), "rows\n")
```

```
## 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.

```
# Identify character variables
chr_vars <- names(FFdf)[sapply(FFdf, is.character)]
cat("Character variables to convert:", length(chr_vars), "\n")
```

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

```
# Convert character variables to factors
for(var in chr_vars) {
  FFdf[[var]] <- as.factor(FFdf[[var]])
}

# Show key factor levels
cat("\nSex levels:", levels(FFdf$Sex), "\n")
```

```
##
## Sex levels: F M
```

```
cat("Marital levels:", levels(FFdf$Marital), "\n")
```

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

```
cat("Account_Type levels:", levels(FFdf$Account_Type), "\n")
```

```
## 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.

```
# Check numeric variable ranges
cat("Age range:", range(FFdf$Age, na.rm = TRUE), "\n")
```

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

```
cat("Tenure range:", range(FFdf$Tenure, na.rm = TRUE), "\n")
```

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

```
cat("NumSeats range:", range(FFdf$NumSeats, na.rm = TRUE), "\n")
```

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

```r
# Check for 999 placeholder values in DistA
cat("\nDistA values = 999:", sum(FFdf$DistA == 999, na.rm = TRUE), "\n")
```

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

```r
# Convert 999 to NA (placeholder for missing)
FFdf$DistA[FFdf$DistA == 999] <- NA
cat("DistA NA count after conversion:", sum(is.na(FFdf$DistA)), "\n")
```

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

```r
# Check Survey_Comp for outliers (expected range 0-1)
cat("\nSurvey_Comp range:", range(FFdf$Survey_Comp, na.rm = TRUE), "\n")
```

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

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

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

```r
# Check for redundant columns
cat("\nState_Name unique:", length(unique(FFdf$State_Name)),
    "| State_Loc unique:", length(unique(FFdf$State_Loc)), "\n")
```

```
##
## 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.

```r
# Convert Last_Contact to datetime format
FFdf$Last_Contact <- ymd_hms(as.character(FFdf$Last_Contact))

# Extract date components
```

```r
FFdf$Contact_Year <- year(FFdf$Last_Contact)
FFdf$Contact_Month <- month(FFdf$Last_Contact)
FFdf$Contact_Day <- day(FFdf$Last_Contact)
FFdf$Contact_Weekday <- wday(FFdf$Last_Contact, label = TRUE)
FFdf$Contact_Hour <- hour(FFdf$Last_Contact)

cat("Date components extracted: Contact_Year, Contact_Month, Contact_Day, Contact_Weekday, Contact_Hour")
```

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

```r
cat("Contact Year range:", range(FFdf$Contact_Year, na.rm = TRUE), "\n")
```

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

```r
cat("Contact Hour range:", range(FFdf$Contact_Hour, na.rm = TRUE), "\n")
```

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

Summary

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

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

```r
cat("Total Missing Values:", sum(is.na(FFdf)), "\n")
```

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

```r
# Show columns with missing values
na_counts <- colSums(is.na(FFdf))
na_cols <- na_counts[na_counts > 0]
cat("\nColumns with missing values:\n")
```

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

```r
print(sort(na_cols, decreasing = TRUE))
```

```
##               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