```r
# Load required libraries for data manipulation, visualization, and reporting
library(knitr)        # For generating R Markdown reports
library(tidyverse)   # For data manipulation and visualization (includes dplyr, ggplot2, etc.)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(tinytex)     # For rendering R Markdown to PDF if needed

library(dplyr)       # For data manipulation (part of tidyverse, explicitly loaded for clarity)
# Install and load required packages
if (!require("dplyr")) install.packages("dplyr")
if (!require("reshape2")) install.packages("reshape2")
```

```
## Loading required package: reshape2
##
## Attaching package: ’reshape2’
##
## The following object is masked from ’package:tidyr’:
##
##     smiths
```

```r
library(dplyr)
library(reshape2)
```

```r
# Check current working directory to ensure the file path is correct
getwd()
```

```
## [1] "C:/Users/amand/OneDrive/Desktop/Assignment2/RprojectAssignment2"
```

```r
# Import the combined dataset from a CSV file
datacombined2 <- read.csv("library_data.csv", header = TRUE, sep = ",")




# Why Import This Way?
# - The dataset 'library_data.csv' contains library performance metrics across multiple years.
# - Specifying header = TRUE and sep = "," ensures the CSV is read correctly with column names.
# - This dataset includes columns like TotalOperatingRevenues, XofActiveLibraryCardholders, etc., which
```

```r
# Clean column names to remove spaces and special characters for easier manipulation
colnames(datacombined2) <- gsub("[^[:alnum:]]", "", colnames(datacombined2))
```

```r
# Why Clean Column Names?
# - Column names with spaces or special characters (e.g., "X of Active Library Cardholders") can cause
# - Using gsub("[^[:alnum:]]", "", ...) ensures names are alphanumeric (e.g., "XofActiveLibraryCardhold
# - This step prevents errors during data manipulation and improves code readability.

# Rename duplicate column names to avoid conflicts
dup_cols <- duplicated(colnames(datacombined2))
if (any(dup_cols)) {
  colnames(datacombined2)[dup_cols] <- paste0(colnames(datacombined2)[dup_cols], "_dup")
}

# Why Handle Duplicates?
# - Duplicate column names can cause unexpected behavior in R (e.g., during subsetting or summarization
# - Appending "_dup" to duplicates ensures all columns are uniquely identifiable.
```

```r
# Ensure key columns are in the correct format for analysis
datacombined2$Year <- as.character(datacombined2$Year)
datacombined2$Library <- as.character(datacombined2$Library)
datacombined2$TotalOperatingRevenues <- as.numeric(as.character(datacombined2$TotalOperatingRevenues))
datacombined2$XofActiveLibraryCardholders <- as.numeric(as.character(datacombined2$XofActiveLibraryCard
datacombined2$PopulationResident <- as.numeric(as.character(datacombined2$PopulationResident))
datacombined2$LocalOperatingGrant <- as.numeric(as.character(datacombined2$LocalOperatingGrant))
datacombined2$Donations <- as.numeric(as.character(datacombined2$Donations))
datacombined2$SelfgeneratedRevenue <- as.numeric(as.character(datacombined2$SelfgeneratedRevenue))
datacombined2$Staffingexpenditure <- as.numeric(as.character(datacombined2$Staffingexpenditure))
datacombined2$TotalAnnualDirectCirculation <- as.numeric(as.character(datacombined2$TotalAnnualDirectCi
```

```
## Warning: NAs introduced by coercion
```

```r
datacombined2$Xofprogramsheldannually <- as.numeric(as.character(datacombined2$Xofprogramsheldannually))
datacombined2$Annualprogramattendance <- as.numeric(as.character(datacombined2$Annualprogramattendance))
datacombined2$XofPublicaccessworkstations <- as.numeric(as.character(datacombined2$XofPublicaccessworks
datacombined2$MainLibrarytotalhoursopenperweek <- as.numeric(as.character(datacombined2$MainLibrarytota
datacombined2$ProjectGrants <- as.numeric(as.character(datacombined2$ProjectGrants))

# Why Convert Data Types?
# - Year and Library are treated as categorical variables (character type) for grouping and labeling.
# - Numeric columns (e.g., TotalOperatingRevenues) must be numeric for calculations like division or co
# - Using as.numeric(as.character(...)) handles cases where numbers might be stored as factors or text,
```

```r
# Create a new column: Operating Revenue per Active Cardholder
datacombined2 <- datacombined2 %>%
  mutate(RevPerCardholder = TotalOperatingRevenues / XofActiveLibraryCardholders)

# Why Create This Column?
# - RevPerCardholder measures how much revenue each active library cardholder generates on average.
# - This metric helps assess the financial efficiency of libraries in serving their active users.
# - It's a key performance indicator (KPI) for understanding how well resources are utilized per user.
```

```r
# Remove rows where RevPerCardholder is NA, infinite, or exceeds 2,500, and ensure no NA in key variabl
datacombined2 <- datacombined2 %>%
  filter(!is.na(RevPerCardholder) & is.finite(RevPerCardholder) & RevPerCardholder <= 2500 &
           !is.na(TotalAnnualDirectCirculation))

# Why Filter These Rows?
# - !is.na(RevPerCardholder): Removes rows where RevPerCardholder is missing (e.g., due to missing Tota
# - is.finite(RevPerCardholder): Removes infinite values (e.g., if XofActiveLibraryCardholders is 0, ca
# - RevPerCardholder <= 2500: Removes outliers (values above 2,500 are unrealistic for revenue per card
# - !is.na(TotalAnnualDirectCirculation): Ensures no missing values in TotalAnnualDirectCirculation, a

# Why Set the Threshold at 2,500?
# - A threshold of 2,500 was chosen as a reasonable upper limit based on domain knowledge: it's highly
# - This threshold helps exclude data entry errors or anomalies (e.g., incorrect revenue or cardholder
```

```
# Calculate the correlation between Revenue per Cardholder and Local Operating Grant
insight1 <- datacombined2 %>%
  summarise(Correlation = cor(RevPerCardholder, LocalOperatingGrant, use = "complete.obs"))

# Display the correlation
print("Insight 1: Correlation between Revenue per Cardholder and Local Operating Grant")
```

```
## [1] "Insight 1: Correlation between Revenue per Cardholder and Local Operating Grant"
```

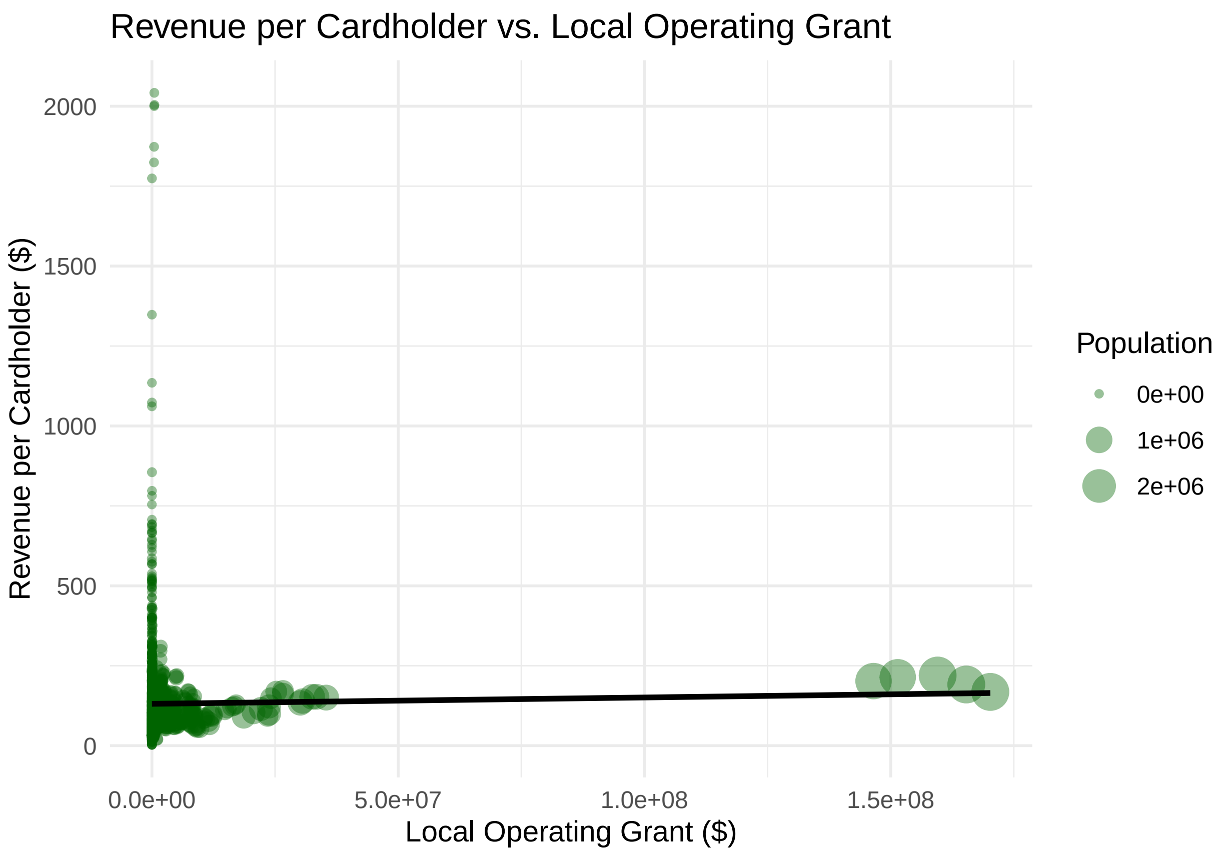
```
print(insight1)
```

```
##   Correlation
## 1   0.0118961
```

```
# Why Calculate This Correlation?
# - We want to understand how strongly LocalOperatingGrant (a key funding source) influences RevPerCard
# - A positive correlation suggests that more local funding leads to higher revenue efficiency per card
# - The 'complete.obs' argument ensures only rows with non-missing values for both variables are used,
```

```
# Bubble plot visualization to explore the relationship
ggplot(datacombined2, aes(x = LocalOperatingGrant, y = RevPerCardholder)) +
  geom_point(aes(size = PopulationResident), alpha = 0.4, color = "darkgreen") +
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  labs(title = "Revenue per Cardholder vs. Local Operating Grant",
       x = "Local Operating Grant ($)",
       y = "Revenue per Cardholder ($)",
       size = "Population") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
# Why Use a Bubble Plot?
# - The x-axis (LocalOperatingGrant) and y-axis (RevPerCardholder) show the primary relationship.
# - Bubble size (PopulationResident) adds a third dimension, showing how population size might influenc
# - The linear trend line (geom_smooth) helps visualize the overall direction of the relationship.
# - Alpha = 0.4 ensures overlapping points are visible, and theme_minimal() keeps the plot clean for pr

# Insight 1 Interpretation:
# - A positive correlation (if observed) indicates that libraries with greater local funding achieve hi
# - Larger bubbles (higher population) may cluster differently, suggesting that population size influen
# - For example, larger populations might have economies of scale, allowing more efficient use of funds
```

```
# Summarize average Revenue per Cardholder by Year and Population Group
insight2 <- datacombined2 %>%
  mutate(PopulationGroup = cut(PopulationResident, breaks = quantile(PopulationResident, probs = 0:3/3,
                               labels = c("Small", "Medium", "Large"), include.lowest = TRUE)) %>%
  group_by(Year, PopulationGroup) %>%
  summarise(AvgRevPerCardholder = mean(RevPerCardholder, na.rm = TRUE), .groups = "drop")

# Display the summarized data
print("Insight 2: Revenue per Cardholder Distribution by Year and Population")
```

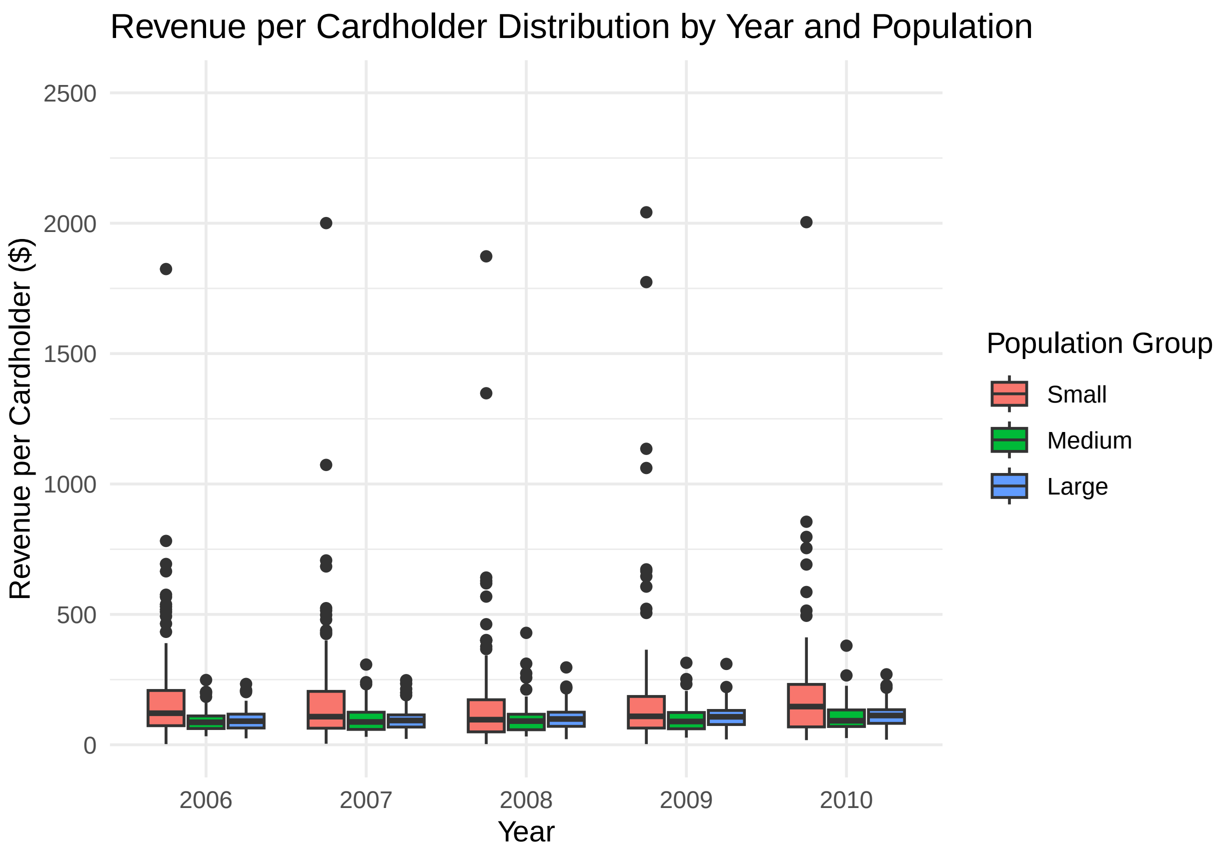
```
## [1] "Insight 2: Revenue per Cardholder Distribution by Year and Population"
```

```r
print(insight2)
```

```
## # A tibble: 15 x 3
##    Year  PopulationGroup AvgRevPerCardholder
##    <chr> <fct>                         <dbl>
##  1 2006  Small                         186.
##  2 2006  Medium                         94.3
##  3 2006  Large                          96.6
##  4 2007  Small                         183.
##  5 2007  Medium                         99.1
##  6 2007  Large                          98.4
##  7 2008  Small                         171.
##  8 2008  Medium                         98.6
##  9 2008  Large                         105.
## 10 2009  Small                         201.
## 11 2009  Medium                        100.
## 12 2009  Large                         110.
## 13 2010  Small                         203.
## 14 2010  Medium                        106.
## 15 2010  Large                         114.
```

```r
# Why Summarize This Way?
# - We categorize PopulationResident into Small, Medium, and Large groups using quantiles (tertiles: 0:
# - This allows us to compare RevPerCardholder across different population sizes and years.
# - Grouping by Year and PopulationGroup helps identify trends over time and across community sizes.
```

```r
# Box plot visualization to show distribution
ggplot(datacombined2, aes(x = Year, y = RevPerCardholder,
                          fill = cut(PopulationResident, breaks = quantile(PopulationResident, probs = 0
                                     labels = c("Small", "Medium", "Large"), include.lowest = TRUE))) +
  geom_boxplot() +
  labs(title = "Revenue per Cardholder Distribution by Year and Population",
       x = "Year", y = "Revenue per Cardholder ($)", fill = "Population Group") +
  ylim(0, 2500) +
  theme_minimal()
```



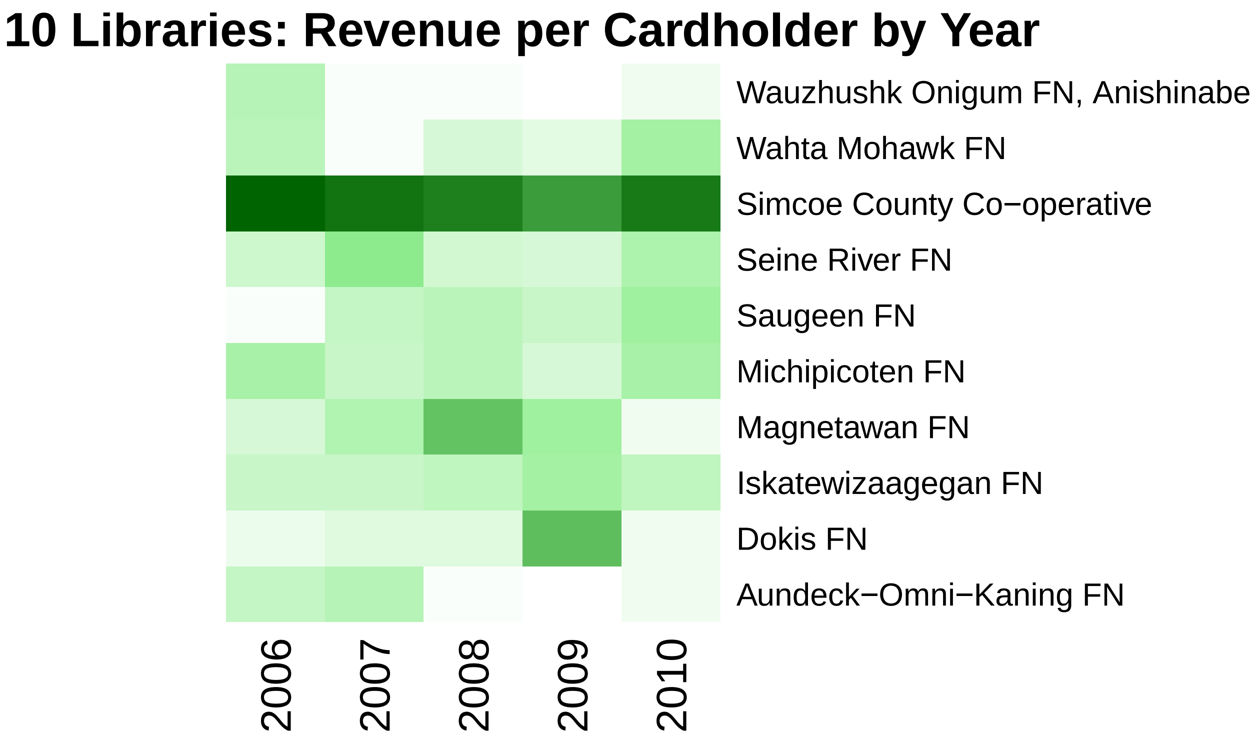
```
# Why Use a Box Plot?
# - Box plots show the distribution (median, quartiles, outliers) of RevPerCardholder for each Year and
# - The fill aesthetic differentiates Small, Medium, and Large population groups, making it easy to com
# - ylim(0, 2500) ensures the y-axis aligns with our filtering threshold, keeping the plot focused on r
# - This visualization highlights trends over time and differences across population sizes.

# Insight 2 Interpretation:
# - If RevPerCardholder increases over time for certain population groups, it might indicate improving
# - If Small population libraries consistently have lower RevPerCardholder, they may face greater chall
# - Outliers in the box plot might indicate specific libraries that are exceptionally efficient or inef
# Save the updated dataset with the new column and filtered rows
```

```
# ----------------------------------------------
# INSIGHT 3: HEATMAP (Top 10 Libraries only)
# Shows Avg Rev per Cardholder across Libraries & Years
# ----------------------------------------------




# Filter top 10 libraries by overall average RevPerCardholder
top_libraries <- datacombined2 %>%
  group_by(Library) %>%
  summarise(OverallAvgRev = mean(RevPerCardholder, na.rm = TRUE)) %>%
  top_n(10, OverallAvgRev) %>%
  pull(Library)
```

```r
# Filter main data
filtered_data <- datacombined2 %>%
  filter(Library %in% top_libraries)

# Group and reshape
heatmap_data <- filtered_data %>%
  group_by(Library, Year) %>%
  summarise(AvgRev = mean(RevPerCardholder, na.rm = TRUE)) %>%
  ungroup()
```

```
## ‘summarise()‘ has grouped output by ’Library’. You can override using the
## ‘.groups‘ argument.
```

```r
heatmap_matrix <- dcast(heatmap_data, Library ~ Year, value.var = "AvgRev")

# Prepare matrix
rownames(heatmap_matrix) <- heatmap_matrix$Library
heatmap_matrix <- heatmap_matrix[, -1]
heatmap_matrix[is.na(heatmap_matrix)] <- 0

# Plot heatmap
heatmap(as.matrix(heatmap_matrix),
        Rowv = NA, Colv = NA,
        col = colorRampPalette(c("white", "lightgreen", "darkgreen"))(50),
        scale = "column",
        margins = c(8, 10),
        main = "Top 10 Libraries: Revenue per Cardholder by Year")
```


10 Libraries: Revenue per Cardholder by Year

```
# WHAT THIS HEATMAP MEANS:
# - Rows: Libraries (Top 10 with highest average RevPerCardholder)
# - Columns: Years
# - Colors: Darker green = higher average revenue per cardholder
# - White = very low or 0 revenue
# ---------------------------------------------
```

```
write.csv(datacombined2, "library_data_with_metrics.csv", row.names = FALSE)

# Why Save the Dataset?
# - Saving the updated dataset ensures that the new column (RevPerCardholder) and filtered data are pre
# - This file can be used for future analyses or shared with stakeholders for transparency.
# - row.names = FALSE prevents adding an unnecessary index column to the CSV.
```