Data Analysis Report and Reflection

Rishi Lal

2025-11-10

Table of contents

```
{r, echo=FALSE, message=FALSE, warning=FALSE} # Load all required packages
library(ggplot2)#Used for plotting the graphs and creating the data visuals.
library(tinytex) # To load tinytex and get latex in order to render the pdf.
library(dcData) # To load the Babynames data into the code. library(tidyverse)#
Used for data manipulation  library(rvest)# Used for web scraping the rank
names library(googlesheets4)# Used for reading data from Google Sheets library(knitr)#
Used for creating the table using the kable() function library(kableExtra)#
Used for styling the table library(janitor)# Used for adding totals to the
table
```

1 Armed Forces Data Wrangling Redux

This section revisits the data wrangling from Activities 8 and 10. The code loads the original dataset, performs the necessary cleaning, and filters for a subset of soldiers (Army) to ensure the tables fit on the PDF page. Unfortunately, the table is crashing the pdf file.

```
#| label: tbl-armed-forces-table
#| tbl-cap: "Frequency of Active Duty US Army Soldiers by Rank and Sex"
#| echo: true
#| warning: false
#| message: false


# Scrape the WebRank Data

rank_url <- "https://neilhatfield.github.io/Stat184_PayGradeRanks.html"
web_ranks <- read_html(x = rank_url) %>%
  html_elements(css = "table") %>%
  html_table()

raw_ranks <- web_ranks[[1]]

# Wrangle the Rank Data

raw_ranks[1, 1] <- "Type"
rank_headers <- raw_ranks[1, ]
names(raw_ranks) <- rank_headers[1, ]
raw_ranks <- raw_ranks[-c(1, 26), ]
```

```
clean_ranks <- raw_ranks %>%
  dplyr::select(!Type) %>%
  pivot_longer(
    cols = !`Pay Grade`,
    names_to = "Branch",
    values_to = "Rank"
  ) %>%
  mutate(
    Rank = na_if(x = Rank, y = "--")
  )

# Load Armed Forces Aggregate Data

gs4_deauth()
sheet_url <- "https://docs.google.com/spreadsheets/d/19xQnI1cBh6Jkw7eP8YQuuicMlVDF7Gr-nXCb5q

forces_headers <- read_sheet(
  ss = sheet_url,
  col_names = FALSE,
  n_max = 3
)

raw_forces <- read_sheet(
  ss = sheet_url,
  col_names = FALSE,
  skip = 3,
  n_max = 28,
  na = c("N/A*")
)

# Wrangle The Armed Forces Aggregate Data

branch_names <- rep(
  x = c("Army", "Navy", "Marine Corps", "Air Force", "Space Force", "Total"),
  each = 3
)
temp_headers <- paste(
  c("", branch_names),
  forces_headers[3,],
  sep = "."
)
names(raw_forces) <- temp_headers
```

```
clean_forces <- raw_forces %>%
  rename(Pay.Grade = `.Pay Grade`) %>%
  dplyr::select(
    !contains("Total")
  ) %>%
  filter(
    !grepl(pattern = "Total", x = Pay.Grade)
  ) %>%
  pivot_longer(
    cols = !Pay.Grade,
    names_to = "Branch.Sex",
    values_to = "Frequency_Chr"
  ) %>%
  separate_wider_delim(
    cols = Branch.Sex,
    delim = ".",
    names = c("Branch", "Sex")
  ) %>%
  mutate(
    Frequency = as.numeric(gsub(pattern = ",", replacement = "", x = Frequency_Chr)) # Remove
  )

# Merge Forces Data with Rank Names

key_forces_ranks <- left_join(
  x = clean_forces,
  y = clean_ranks,
  by = join_by(Pay.Grade == `Pay Grade`, Branch == Branch)
)

#Transform to the Individual Soldier Data Frame

individual_soldiers <- key_forces_ranks %>%
  filter(
    !is.na(Frequency)
  ) %>%
  uncount(
    weights = Frequency
  )

# Create the Final Frequency Table
```

```r
army_rank_sex_table_data <- individual_soldiers %>%
  filter(
    Branch == "Army"
  ) %>%
  filter(
    !is.na(Rank)
  ) %>%
  group_by(
    Pay.Grade,
    Rank,
    Sex
  ) %>%
  tally(name = "Frequency") %>%
  ungroup() %>%
  pivot_wider(
    names_from = Sex,
    values_from = Frequency,
    values_fill = 0
  ) %>%
  dplyr::select(
    -Pay.Grade
  ) %>%
  # Add row and column totals
  janitor::adorn_totals(
    where = c("row", "col"),
    name = "Total"
  )

# Display the Table

kable(
  x = army_rank_sex_table_data,
  col.names = c("Rank", "Female", "Male", "Total"),
  align = 'lrrr'
) %>%
  # Apply styling for better legibility
  kable_styling(
    bootstrap_options = c("striped", "dark"),
    full_width = FALSE,
    position = "left"
  )
```

The table above displays personnel counts for the US Army, broken down by rank and sex. It focuses specifically on a subset of the enlisted ranks, listing them vertically in the first column, beginning with “Private” and “Private First Class” and ascending to “Sergeant First Class.” The other columns quantify the number of “Female” and “Male” soldiers for each specific rank, with a final “Total” column summing these two groups. A clear pattern visible can be seen in this data is that male soldiers significantly outnumber female soldiers across every rank category shown. For instance, the “Corporal OR Specialist” rank includes 15,143 females and 79,234 males, demonstrating this difference in distribution. This data visualization provides a clear, disaggregated view of the gender composition within these specific Army enlisted levels.

2 Popularity of Baby Names

This section incorporates the visualization from Activity 13. The plot below shows the popularity of several baby names over time.

I chose the names Michael, David, Matthew, and Olivia because they are some of the most common names here in the US and are also the names of some of my friends and also people that I know from my different classes, this semester.

“‘{r, echo=FALSE, message=FALSE, warning=FALSE, fig.cap=“Popularity of selected baby names over time.”} #| fig-alt: “A line chart showing the popularity of four U.S. baby names from 1880 to around 2014. The x-axis represents the ‘Year’ and the y-axis represents the ‘Total Number of Babies Given Name’. Four names are tracked: ‘Michael’, ‘David’, ‘Matthew’, and ‘Olivia’. ‘Michael’ and ‘David’ show sharp rises and peaks in the mid-20th century before declining. ‘Matthew’ peaks later, around the 1980s-1990s, before declining. ‘Olivia’ shows low popularity for most of the chart’s history, followed by a rise in the late 20th and early 21st century, crossing the other names in popularity by the end.”

3 Load the data from the package

data(BabyNames)

4 Define the names I want to track

my_names <- c(“Michael”, “David”, “Matthew”, “Olivia”)

5 Process the data

names_over_time <- BabyNames %>% filter(name %in% my_names) %>% # Filter for the names I want to track. group_by(name, year) %>% # Group by name and year summarize(total_count = sum(count)) %>% # Sum counts for total popularity (M+F). ungroup()

6 Define a color-blind friendly palette.

7 We name the vector to ensure the correct color is assigned to each name.

cb_friendly_colors <- c(“Michael” = “#009E73”, # Bluish Green “David” = “#E69F00”, # Orange “Matthew” = “#D55E00”, # Vermillion “Olivia” = “#56B4E9” # Sky Blue)

8 Create the plot

ggplot(data = names_over_time, aes(x = year, y = total_count, color = name)) +

# Map name to color only geom_line(linewidth = 1.2) + # Make lines thicker

# Use the manually defined color-blind friendly palette scale_color_manual(values = cb_friendly_colors) +

# Adds more context with a clearer title, subtitle, and caption labs(title = “Popularity Trends of Four U.S. Baby Names”, subtitle = “Total annual count of ‘Michael’, ‘David’, ‘Matthew’, and ‘Olivia’ (Late 19th to Early 21st Century)”, x = “Year”, # Set the X value to Year y = “Total Number of Babies Given Name”, # Set the Y value to the total number of babies. color = “Name” # Legend title for color)+ # Use a clean, minimal theme theme_minimal() +

# Centers the title and subtitle theme(plot.title = element_text(hjust = 0.5, face = “bold”, size = 16), plot.subtitle = element_text(hjust = 0.5, size = 12), # Note: ‘plot.caption’ is removed as Quarto handles the figure caption legend.position = “right” # Ensure legend is clearly visible)

```
This line chart, titled "Popularity Trends of Four U.S. Baby Names," displays the annual cou

The visualization highlights distinct generational trends for these four names. "Michael" (t
```

~~~~
In stark contrast, "Olivia" (the light blue line) remain quite low in popularity for over a 

# Plotting the Volume of the Box by the size of the cutout

This section revisits the Box Problem. We are starting with a standard piece of 36 x 48 inch

$V(a) = (36 - 2a)(48 - 2a)a$

The plot below visualizes this volume function using `ggplot2`. The x-axis represents the si

```{r, echo=FALSE, message=FALSE, warning=FALSE, fig.cap="Volume of a box as a function of c
#Similar to my code from activity 4, changed for different values.
# Define the function for a 36 x 48 inch paper
# The function can handle a vector of input values 'a'.
get_box_volume_36_48 <- function(a) {
  # Define paper dimensions
  paper_len <- 48
  paper_width <- 36

  # Calculate box dimensions based on the cutout 'a'
  box_len <- paper_len - 2 * a
  box_width <- paper_width - 2 * a
  box_height <- a

  # Calculate the volume
  box_vol <- box_len * box_width * box_height

  # Check for any invalid results
  # 'a' must be positive.
  # '2 * a' must be less than the shortest side (36 in this case), so 'a' must be less than 
  box_vol <- ifelse(a <= 0 | a >= (paper_width / 2), NA, box_vol)

  return(box_vol)
}

# Create the ggplot using the stat_function
box_plot_36x48 <- ggplot(
  data = data.frame(x = c(0, 18)), # Set the x-axis limits for the function
  mapping = aes(x = x)
) +
  stat_function(
    fun = get_box_volume_36_48,
    linewidth = 1.2,
~~~~

```
    color = "blue"
  ) +
  labs(
    title = "Box Volume by Cutout Size",
    subtitle = "Based on 36 x 48 inch paper",
    x = "Side Length of Cutout 'a' (inches)",
    y = "Volume V(a) (cubic inches)",
    caption = "Figure 1: Plot of the box volume function V(a) = (48-2a)(36-2a)a.",
    alt = "A line plot showing the volume of a box based on a cutout size 'a'. The x-axis ra
  ) + # Alt text.
  theme_minimal() # Use a minimal theme for a clean look

# Print the plot
print(box_plot_36x48)
```

The data visualization, “Box Volume by Cutout Size,” visually illustrates the relationship between the side length of a square cutout, ‘a’ (shown on the x-axis from 0 to 18 inches), and the resulting volume of an open-top box (shown on the y-axis) made from a 36x48 inch piece of paper. The plot’s curve begins at a zero volume for a zero-inch cutout, rises as the cutout size increases, and then falls back to zero volume as the cutout approaches 18 inches, which logically represents the physical limits of the paper. This visual representation clearly shows that the volume is not linear; it reaches a distinct maximum. Based off of the peak of the curve, we can see that the optimal cutout size ‘a’ is approximately between 6 and 7 inches, which yields the maximum possible box volume of approximately 5,000+ cubic inches.

9 My Reflection

Honestly, this course has been time-consuming, and now that I am reflecting on it, it’s not particularly a bad thing. I’ve spent countless hours on RStudio, trying to get my code to work, trying to polish, and make my code look neater, trying to debug and find effective solutions and trying to learn more about the libraries that we have used so far in this course. I never thought about planning my code so far in college, I’ve always winged it more or less, and have never been able to sit down and actually plan the code, including the functions, the variables and the requirements that I needed. Oftentimes, it lead to problems down the line, and now, I find that really trying to think through what I need to do and also putting it down on paper, so, that I don’t forget it has proven to be enormously beneficial. I have been fortunate to work with R in the past, but I never got to do a deep dive into it and cover the basics. For example, with respect to ggplot2, I never used it for anything more than a scatterplot, and now, I’ve had to add a line plot on top in order to allow for better visualization for the casual viewer. At the same time, in my previous course, we had access to the raw dataset as an excel file, but in this course(albeit, a bit painful at times), I’ve had to web scrape actual websites and also google sheets in order to get the data to be used, so, this is something that I have found to be practical in nature, and cannot wait to use it later on for my own personal projects, I have found it easier to do in R, then in Python for example.

10 Code Appendix

This Code Appendix has all the necessary code used above.

10.1 Setup and Libraries

This chunk contains all the R packages loaded for this analysis.

`{r, eval=FALSE, echo=TRUE} # Load all required R packages library(ggplot2)#Used for plotting the graphs and creating the data visuals. library(tinytex) library(dcData) library(tidyverse)# Used for data manipulation (dplyr, tidyr, etc.) library(rvest)# Used for web scraping the rank names library(googlesheets4)# Used for reading data from Google Sheets library(knitr)# Used for creating the kable() table library(kableExtra)# Used for styling the table for dark mode library(janitor)# Used for adding totals to the table`

10.2 Armed Forces Data Wrangling

This code was used to load, clean, and wrangle the armed forces data to produce the frequency tables.

```
#| tbl-cap: "Frequency of Active Duty US Army Soldiers by Rank and Sex (June 2025)"
#| echo: true
#| warning: false
#| message: false


# Scrape the WebRank Data

rank_url <- "https://neilhatfield.github.io/Stat184_PayGradeRanks.html"
web_ranks <- read_html(x = rank_url) %>%
  html_elements(css = "table") %>%
  html_table()

raw_ranks <- web_ranks[[1]]
```

```r
# Wrangle the Rank Data

raw_ranks[1, 1] <- "Type"
rank_headers <- raw_ranks[1, ]
names(raw_ranks) <- rank_headers[1,]
raw_ranks <- raw_ranks[-c(1, 26), ]

clean_ranks <- raw_ranks %>%
  dplyr::select(!Type) %>%
  pivot_longer(
    cols = !`Pay Grade`,
    names_to = "Branch",
    values_to = "Rank"
  ) %>%
  mutate(
    Rank = na_if(x = Rank, y = "--")
  )

# Load Armed Forces Aggregate Data

gs4_deauth()
sheet_url <- "https://docs.google.com/spreadsheets/d/19xQnI1cBh6Jkw7eP8YQuuicMlVDF7Gr-nXCb5q

forces_headers <- read_sheet(
  ss = sheet_url,
  col_names = FALSE,
  n_max = 3
)

raw_forces <- read_sheet(
  ss = sheet_url,
  col_names = FALSE,
  skip = 3,
  n_max = 28,
  na = c("N/A*")
)
#Wrangle The Armed Forces Aggregate Data
branch_names <- rep(
  x = c("Army", "Navy", "Marine Corps", "Air Force", "Space Force", "Total"),
  each = 3
)
temp_headers <- paste(
  c("", branch_names),
  forces_headers[3,],
  sep = "."
)
names(raw_forces) <- temp_headers

clean_forces <- raw_forces %>%
  rename(Pay.Grade = `.Pay Grade`) %>%
  dplyr::select(
    !contains("Total")
  ) %>%
  filter(
    !grepl(pattern = "Total", x = Pay.Grade)
  ) %>%
  pivot_longer(
    cols = !Pay.Grade,
    names_to = "Branch.Sex",
    values_to = "Frequency_Chr"
  ) %>%
  separate_wider_delim(
    cols = Branch.Sex,
    delim = ".",
    names = c("Branch", "Sex")
  ) %>%
  mutate(
    Frequency = as.numeric(gsub(pattern = ",", replacement = "", x = Frequency_Chr)) # Remove
  )

# Merge Forces Data with Rank Names

key_forces_ranks <- left_join(
  x = clean_forces,
  y = clean_ranks,
  by = join_by(Pay.Grade == `Pay Grade`, Branch == Branch)
)

#Transform to the Individual Soldier Data Frame

individual_soldiers <- key_forces_ranks %>%
  filter(
    !is.na(Frequency)
  ) %>%
  uncount(
    weights = Frequency
  )

# Create the Final Frequency Table
army_rank_sex_table_data <- individual_soldiers %>%
  filter(
    Branch == "Army"
  ) %>%
  filter(
    !is.na(Rank)
  ) %>%
  group_by(
    Pay.Grade,
    Rank,
    Sex
  ) %>%
  tally(name = "Frequency") %>%
  ungroup() %>%
  pivot_wider(
    names_from = Sex,
    values_from = Frequency,
    values_fill = 0
  ) %>%
  dplyr::select(
    -Pay.Grade
  ) %>%
  # Add row and column totals
  janitor::adorn_totals(
    where = c("row", "col"),
    name = "Total"
  )

# Display the Table

kable(
  x = army_rank_sex_table_data,
  format = "latex",
  col.names = c("Rank", "Female", "Male", "Total"),
  align = 'lrrr'
) %>%
  # Apply styling for dark backgrounds and better legibility
  kable_styling(
    bootstrap_options = c("striped", "hold_position"), #
    full_width = FALSE,
    position = "left"
  )
  )
```

10.3 Popularity of Baby Names

This code was used to filter the `BabyNames` data and generate the visualization.

"'{r, echo=FALSE, message=FALSE, warning=FALSE, fig.cap="Popularity of selected baby names over time."} #| fig-alt: "A line chart showing the popularity of four U.S. baby names from 1880 to around 2014. The x-axis represents the 'Year' and the y-axis represents the 'Total Number of Babies Given Name'. Four names are tracked: 'Michael', 'David', 'Matthew', and 'Olivia'. 'Michael' and 'David' show sharp rises and peaks in the mid-20th century before declining. 'Matthew' peaks later, around the 1980s-1990s, before declining. 'Olivia' shows low popularity for most of the chart's history, followed by a rise in the late 20th and early 21st century, crossing the other names in popularity by the end."

11 Load the data from the package

data(BabyNames)

12 Define the names I want to track

my_names <- c("Michael", "David", "Matthew", "Olivia")

13 Process the data

names_over_time <- BabyNames %>% filter(name %in% my_names) %>% # Filter for the names I want to track. group_by(name, year) %>% # Group by name and year summarize(total_count = sum(count)) %>% # Sum counts for total popularity (M+F). ungroup()

14 Define a color-blind friendly palette

15 We name the vector to ensure the correct color is assigned to each name.

cb_friendly_colors <- c(“Michael” = “#009E73”, # Bluish Green “David” = “#E69F00”, # Orange “Matthew” = “#D55E00”, # Vermillion “Olivia” = “#56B4E9” # Sky Blue)

16 Create the plot

ggplot(data = names_over_time, aes(x = year, y = total_count, color = name)) +

# Map name to color only geom_line(linewidth = 1.2) + # Make lines thicker

# Use the manually defined color-blind friendly palette scale_color_manual(values = cb_friendly_colors) +

# Adds more context with a clearer title, subtitle, and caption labs(title = “Popularity Trends of Four U.S. Baby Names”, subtitle = “Total annual count of ‘Michael’, ‘David’, ‘Matthew’, and ‘Olivia’ (Late 19th to Early 21st Century)”, x = “Year”, # Set the X value to Year y = “Total Number of Babies Given Name”, # Set the Y value to the total number of babies. color = “Name” # Legend title for color)+ # Use a clean, minimal theme theme_minimal() +

# Centers the title and subtitle theme(plot.title = element_text(hjust = 0.5, face = “bold”, size = 16), plot.subtitle = element_text(hjust = 0.5, size = 12), # Note: ‘plot.caption’ is removed as Quarto handles the figure caption legend.position = “right” # Ensure legend is clearly visible)

~~~
## Plotting the Volume of the Box function.

This code defines the function for the Volume of the Box calculation

```{r, echo=FALSE, message=FALSE, warning=FALSE, fig.cap="Volume of a box as a function of cu
#Similar to my code from activity 4, changed for different values.
# Define the function for a 36 x 48 inch paper
# The function can handle a vector of input values 'a'.
get_box_volume_36_48 <- function(a) {
  # Define paper dimensions
  paper_len <- 48
  paper_width <- 36
~~~

```
  # Calculate box dimensions based on the cutout 'a'
  box_len <- paper_len - 2 * a
  box_width <- paper_width - 2 * a
  box_height <- a

  # Calculate the volume
  box_vol <- box_len * box_width * box_height

  # Check for any invalid results
  # 'a' must be positive.
  # '2 * a' must be less than the shortest side (36 in this case), so 'a' must be less than 
  box_vol <- ifelse(a <= 0 | a >= (paper_width / 2), NA, box_vol)

  return(box_vol)
}

# Create the ggplot using stat_function
box_plot_36x48 <- ggplot(
  data = data.frame(x = c(0, 18)), # Set the x-axis limits for the function
  mapping = aes(x = x)
) +
  stat_function(
    fun = get_box_volume_36_48,
    linewidth = 1.2,
    color = "blue"
  ) +
  labs(
    title = "Box Volume by Cutout Size",
    subtitle = "Based on 36 x 48 inch paper",
    x = "Side Length of Cutout 'a' (inches)",
    y = "Volume V(a) (cubic inches)",
    caption = "Figure 1: Plot of the box volume function V(a) = (48-2a)(36-2a)a.",
    alt = "A line plot showing the volume of a box based on a cutout size 'a'. The x-axis ra
  ) +
  theme_minimal() # Use a minimal theme for a clean look

# Print the plot
print(box_plot_36x48)
```