

Activity 14 - Progress in STAT 184

Presenting progress through activity results

Andrew Mason

2025-11-09

1 Exploring the Armed Forces

I will use the wrangled data to explore whether sex and rank are independent of one another through a two-way frequency table.

Table 1: Army Enlisted Gender Distribution by Rank

Rank	Female	Male
Private	5662	29767
Private First Class	10229	43775
Corporal OR Specialist	15143	79234
Sergeant	10954	54803
Staff Sergeant	7363	49502
Sergeant First Class	4410	30264
First Sergeant OR Master Sergeant	1472	9482
Sergeant Major OR Command Sergeant Major	394	2865

1.1 Observations from the table

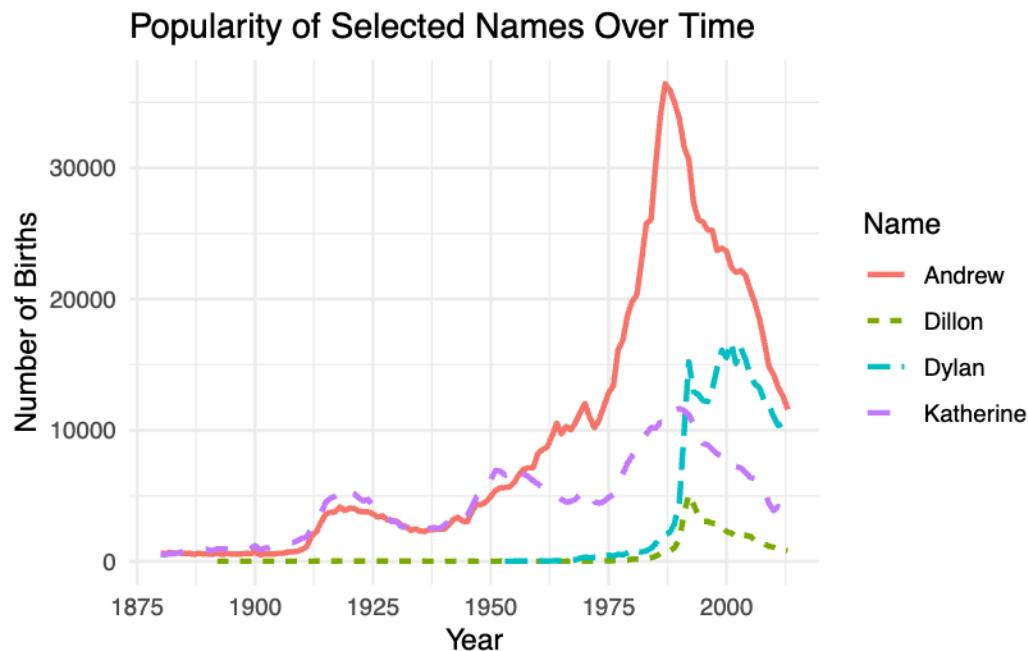
The table displays the gender distribution of enlisted personnel across Army ranks. From the data, we can observe a consistent pattern: the number of male soldiers significantly exceeds that of female soldiers at every enlisted rank. While both sexes are represented at all levels, the disparity becomes more pronounced in higher ranks such as Sergeant First Class and Sergeant Major or Command Sergeant Major.

This pattern suggests that sex and rank are not independent within the Army's enlisted personnel. If they were independent, the proportion of male and female soldiers would remain relatively constant across ranks. Instead, the decreasing proportion of women as rank increases indicates a relationship between sex and rank—men are more likely to occupy senior enlisted positions, whereas women are more concentrated in lower ranks.

Overall, this visualization highlights structural gender differences in career progression within the Army, demonstrating that sex and rank are associated rather than independent.

2 Popularity of Baby Names

To explore the popularity of some names, I've decided to focus on the popularity of the first names of myself and immediate family: Andrew, Dillon, Dylan, and Katherine. I will focus on the overall popularity of the name, without making a distinction based on the baby's sex.



2.1 Observations from the visualization

The chart tracks the number of U.S. births per year for four given names—Andrew, Dillon, Dylan, and Kathryn—from approximately 1880 through 2020. The visualization uses both color and line type to distinguish each name.

Trends:

Andrew (solid red): Displays a long-term upward trend beginning in the early 1900s, reaching a dramatic peak around the mid-1980s, followed by a gradual decline.

Dylan (dot-dashed teal): Shows minimal usage before 1960, then a sharp rise beginning in the 1980s, peaking near 2000, before tapering off slightly.

Dillon (dashed green): Remains nearly unused until the 1980s, then experiences a modest rise around 1995, but stays far below Dylan's popularity.

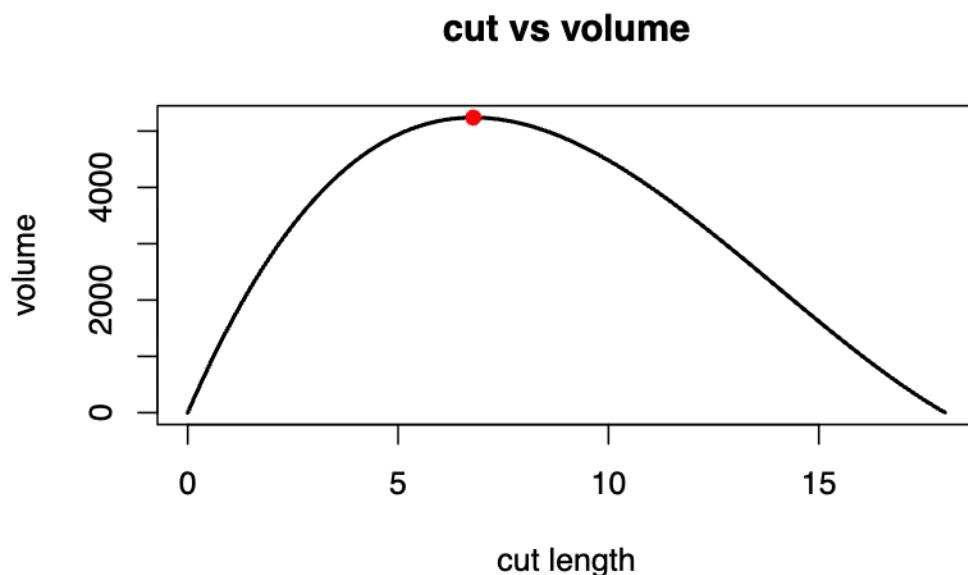
Kathryn (dotted purple): Peaks around the 1910s, declines mid-century, and rises again around the 1980s, though never reaching Andrew's or Dylan's numbers.

Visual design: Distinct line types and colors improve readability and allow overlapping patterns to be easily distinguished. The minimal theme and smooth temporal axis highlight the general trends without visual clutter.

3 The Box Problem

```
$maximum  
[1] 6.788902
```

```
$objective  
[1] 5239.819
```



3.1 Observations from the visualization

The chart, titled “cut vs volume,” visualizes how the volume of an open-top box changes with varying cut lengths from the corners of a rectangular sheet. The x-axis shows the cut length, and the y-axis represents the resulting box volume.

The curve rises rapidly from the origin, reaches a distinct peak, and then declines smoothly, forming a clear parabolic shape. A red point marks the maximum value, representing the optimal cut length that yields the largest box volume.

According to the solution metrics, the maximum volume occurs at a cut length of approximately 6.79 units, producing a box volume of about 5239.82 cubic units. This point reflects the balance between increasing height and decreasing base area—beyond this cut length, the box loses volume as the base becomes too small to offset the taller sides.

4 Self Reflection

I went into this course believing that learning R would be similar to learning Python, in the sense that most lessons would be utility-focused and based on learning programming basics. However, at

this point in the semester, I've learned that even though it uses similar syntax, R is a fundamentally unique language from Python, and comes with a key focus on creating beautiful data, beyond simply functional data.

Learning functions in R was the first step to this - definitions in this language are more formalized, and the visual look of the code is meant to be neat and readable by humans. R differs from Python in this way - while Python IS written in English, it is much more flexible on defining exactly what a parameter does. R not only has built in integrity for readability, but there is also a larger school of thought surrounding the aesthetics of R code.

The most important part of this class to me, however, has been learning the ins and outs of building effective and beautiful data visualizations. `{ggplot2}` is an old and powerful package with ongoing support and documentation, along with a number of other packages for data visualization, all of which have their own specific use cases. R as a whole seems to be primarily dedicated to creating traditionally useful, human-readable data visualizations, rather than bulk-processing big data.

I look forward to diving further into R as not only a programming language, but as a generalized school of thought into data analytics.

5 Code Appendix

5.1 U.S. Armed Forces Data

5.1.1 Step 1: Building Ranks DF for Rank Join

```
# Scrape Ranks from Github
ranks <- read_html(
  "https://neilhatfield.github.io/Stat184_PayGradeRanks.html") %>%
  html_elements(css = "table") %>%
  html_table()

ranks_df <- ranks[[1]]

# Cleaning Ranks DF
ranks_df[1, 1] <- "Type" # cell 1 value
ranks_head <- ranks_df[1, ] # column headers
names(ranks_df) <- ranks_head[1,] # header -> column name
ranks_df <- ranks_df[-c(1, 26), ] # remove irrelevant rows
ranks_df <- ranks_df %>%
  rename(Pay.Grade = `Pay Grade`) # remove bad name

ranks_clean <- ranks_df %>%
  dplyr::select(!Type) %>%
  pivot_longer(
    cols = !Pay.Grade,
    names_to = "Branch",
    values_to = "Rank"
```

```

) %>%
mutate(
  Rank = na_if(x = Rank, y = "--")
)

```

5.1.2 Step 2: Building USAF Data Frame

```

# Grab DF from googlesheets4
## headers
gs4_deauth()

raw_headers <- read_sheet(
  ss =
    "https://docs.google.com/spreadsheets/d/19xQnI1cBh6Jkw7eP8YQuuicM1VDF7Gr-nXCb5qbwb_E/edit?#gid=144000000"
  col_names = FALSE,
  n_max = 3
)

## DF
raw <- read_sheet(
  ss =
    "https://docs.google.com/spreadsheets/d/19xQnI1cBh6Jkw7eP8YQuuicM1VDF7Gr-nXCb5qbwb_E/edit?#gid=144000000"
  col_names = FALSE, # will define column names later
  skip = 3, # given headers are incomplete, will need to define later
  n_max = 28,
  na = c("N/A*")
)

```

5.1.3 Step 3: Data Wrangling for USAF Data Frame

```

# Wrangling + Cleaning
branch <- rep( # repeat due to 3 occurrences of each branch
  x = c("Army", "Navy", "Marine Corps", "Air Force", "Space Force", "Total"),
  each = 3
)

raw_head <- paste( # branch + headers
  c("Pay.Grade", branch),
  raw_headers[3,],
  sep = "."
)

raw_head[1] <- "Pay.Grade"

```

```
names(raw) <- raw_head # Applying headers to column names
```

5.1.4 Step 4: Tidying USAF Data Frames

```
# Tidying Data for first DF
df <- raw %>%
  pivot_longer(
    cols = -Pay.Grade,
    names_to = c('Branch', 'Gender'),
    names_pattern = '(Army|Navy|Marine\\.Corps
  |Air\\.Force|Space\\.Force|Total)\\.(Male|Female|Total)',
    values_to = 'Count'
  ) %>%
  filter(!str_detect(Pay.Grade, regex('total', ignore_case = TRUE))) %>%
  filter(!str_detect(Pay.Grade, regex('Source', ignore_case = TRUE))) %>%
  filter(!str_detect(Branch, regex('total', ignore_case = TRUE))) %>%
  filter(!str_detect(Gender, regex('total', ignore_case = TRUE))) %>%
  filter(!is.na(Count)) %>%
  mutate(
    Count = gsub(", ", "", Count),
    Count = as.integer(Count)
  )

# Merge Rank DF and Armed Forces DF
df1 <- left_join(
  x = df,
  y = ranks_clean,
  by = join_by(Pay.Grade == Pay.Grade, Branch == Branch)
)
```

5.1.5 Step 5: Creating frame where a case is an individual soldier

```
# Tidying Data for second DF
df2 <- df1 %>%
  uncount(weights = Count)
```

5.1.6 Step 6: Creating frequency table correlating Gender with Rank

```
# Create frequency table
gender_freq <- df2 %>%
  filter(
```

```

Pay.Grade %in% c('E1', 'E2', 'E3', 'E4', 'E5', 'E6', 'E7', 'E8', 'E9'),
Branch == "Army"
) %>%
mutate(
  Rank = factor(Rank, levels = unique(Rank))) %>%
tabyl(Rank, Gender)
print(
  knitr::kable(
    gender_freq,
    format = "latex",
    booktabs = TRUE,
    align = c("l", "r", "r"),
    caption = "Army Enlisted Gender Distribution by Rank"
) %>%
  kableExtra::kable_styling(latex_options = "hold_position")
)

```

5.2 Baby Names

5.2.1 Load and prepare data for visualization

```

# Load data
data(BabyNames, package='dcData')

# Filter for selected names (given in project description)
names_subset <- BabyNames %>%
  filter(name %in% c('Andrew', 'Dillon', 'Dylan', 'Katherine')) %>%
  group_by(year, name) %>%
  summarize(total = sum(count), .groups = "drop")

```

5.2.2 Create data visualization

```

# Visualization: popularity over time
print(
  ggplot(names_subset, aes(x = year, y = total,
                           color = name, linetype = name)) +
  geom_line(linewidth = 1) +
  labs(title = "Popularity of Selected Names Over Time",
       x = "Year",
       y = "Number of Births",
       color = "Name",
       alt = "A line chart titled 'Popularity of Selected Names Over Time.'\n      The x-axis shows years from about 1880 to 2020, and the y-axis shows

```

```

the number of births. Four lines represent the names
Andrew (solid red), Dillon (dashed green), Dylan (dot-dashed teal),
and Kathryn (dotted purple). Andrew peaks sharply around 1985,
Dylan rises in the 1990s, Kathryn shows moderate peaks mid-century
and late 20th century, and Dillon remains consistently low." ,
linetype = "Name") +
theme_minimal()
)

```

5.3 The Box Problem

5.3.1 Defining goals, functions

```

# Goal 1: Create a function to get the volume of a box
# Needs: side length of cutouts (cutLength), paper dimensions, Volume formula
# Defining paper dimensions
PAPERLEN <- 36
PAPERWID <- 48

# Applying to volume function
vol_calc <- function(cutLength){
  length <- PAPERLEN-(2*cutLength)
  width <- PAPERWID-(2*cutLength)
  volume <- length*width*cutLength
  return(volume)
}

# Plugging volume function into optimization
max <- optimize(vol_calc, c(0, (PAPERLEN/2)), maximum=TRUE)
print(max)

```

5.3.2 Creating data visualization

```

# Goal 2: Use plot() to create data visualization showing the relationship
# between cut and volume
curve(vol_calc, from=0, to=(PAPERLEN/2), col="black", lwd=2,
      xlab="cut length", ylab="volume", main="cut vs volume")
points(max$maximum, max$objective, col="red", pch=19)

```