STAA 551 Assignment 1

Matthew Stoebe

2024-08-29

Assignment Description

Use R Markdown to complete your assignment.

All problem numbers refer to Regression and Other Stories by Gelman, Hill, and Vehtari Submission is a PDF file upload. There is only one attempt for this assignment, so make sure you upload the correct file

- Chapter 2: 2.3, 2.9
- Chapter 3: 3.6–3.9

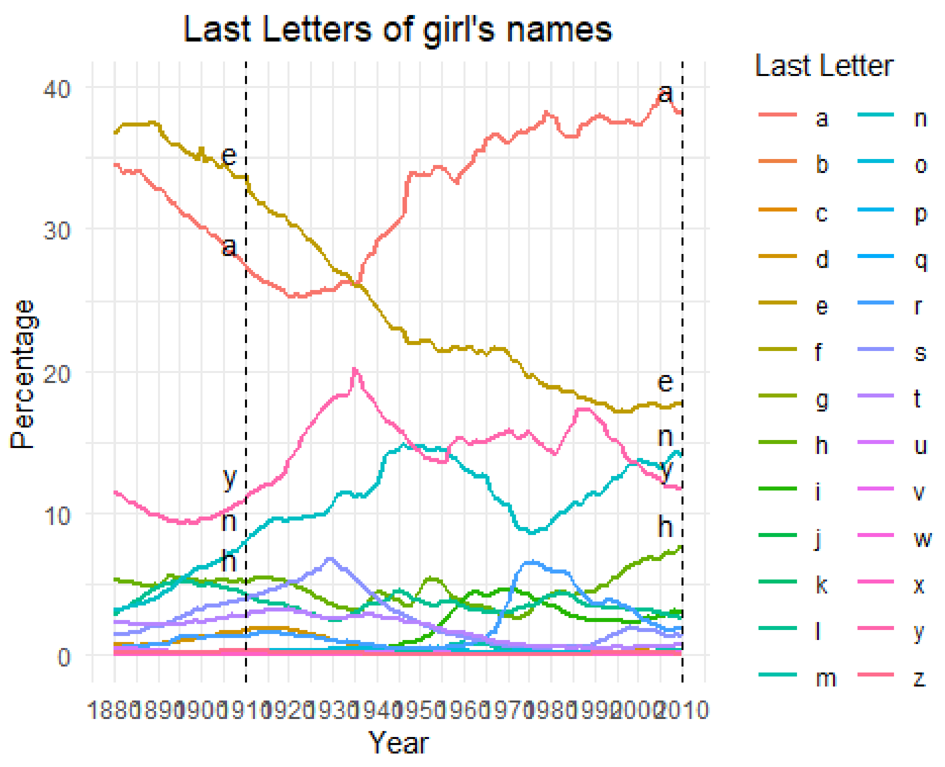
Question 2.3

```r
allnames <- read.csv("./ROS-Examples-master/Names/data/allnames_clean.csv")

female_data <- filter(allnames, sex == "F")

female_data$last_letter <- substr(female_data$name, nchar(female_data$name),
nchar(female_data$name))

grouped_data <- group_by(female_data, last_letter)


summarized_data <- summarise(grouped_data, across(matches("^X\\d"), sum,
na.rm = TRUE))
```

```
## Warning: There was 1 warning in `summarise()`.
## ℹ In argument: `across(matches("^X\\d"), sum, na.rm = TRUE)`.
## ℹ In group 1: `last_letter = "a"`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```r
total_sums <- colSums(summarized_data[, -1])  # Exclude the first column
(last_letter)

percentage_data <- summarized_data

for (col in names(summarized_data[-1])) {
  percentage_data[[col]] <- 100 * (percentage_data[[col]] / total_sums[col])
}


percentage_long <- pivot_longer(percentage_data, cols = starts_with("X"),
                                names_to = "year",
                                values_to = "percentage")
percentage_long <- percentage_long %>%
  mutate(year = as.numeric(sub("^X", "", year)))


top_labels_2010 <- percentage_long %>%
  filter(year == 2010) %>%
  arrange(desc(percentage)) %>%
  slice_head(n = 5)  # Get the top 3 lines based on percentage

top_labels_1910 <- percentage_long %>%
  filter(year == 1910) %>%
  arrange(desc(percentage)) %>%
  slice_head(n = 5)  # Get the top 3 lines based on percentage



ggplot(percentage_long, aes(x = year, y = percentage, color = last_letter,
group = last_letter)) +
  geom_line(size = 1) +
  labs(title = "Last Letters of girl's names",
       x = "Year",
       y = "Percentage",
       color = "Last Letter") +
  scale_x_continuous(breaks = seq(1880, 2010, by = 10)) +  # Show labels for
every 10 years
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +

  geom_text(data = top_labels_2010,
            aes(label = last_letter),
            vjust = -0.5, hjust = 1.5, color = "black") +

  # Optional: Add custom annotations or vertical lines as needed
  geom_vline(xintercept = 2010, linetype = "dashed", color = "black") +

  geom_text(data = top_labels_1910,
            aes(label = last_letter),
            vjust = -0.5, hjust = 1.5, color = "black") +

  # Optional: Add custom annotations or vertical lines as needed
  geom_vline(xintercept = 1910, linetype = "dashed", color = "black")
```

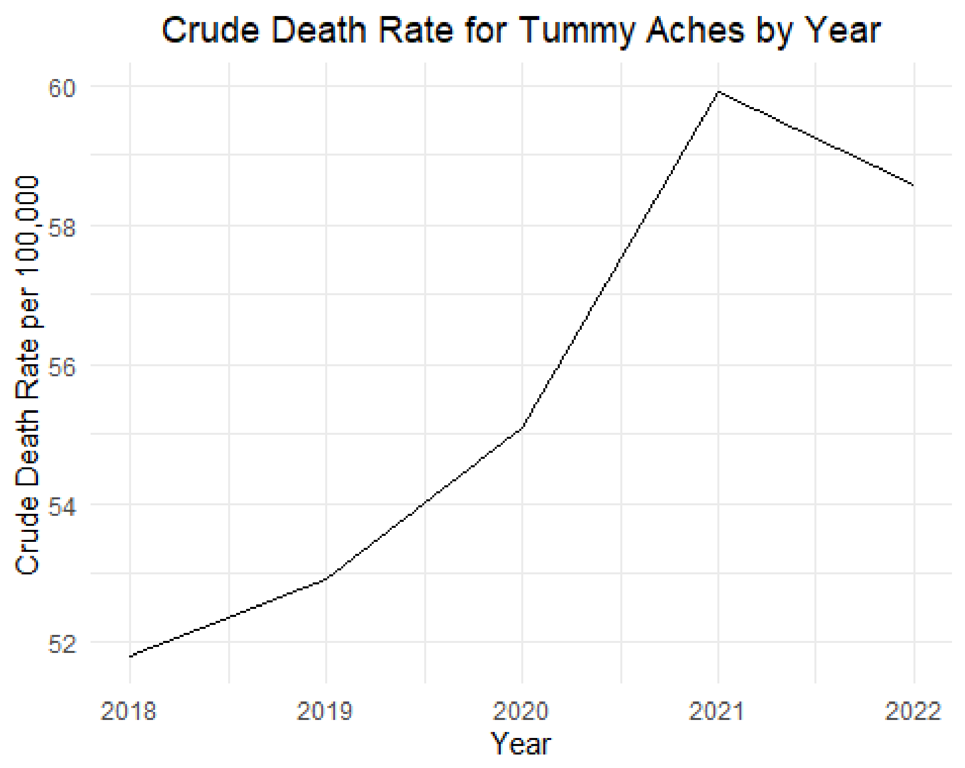
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



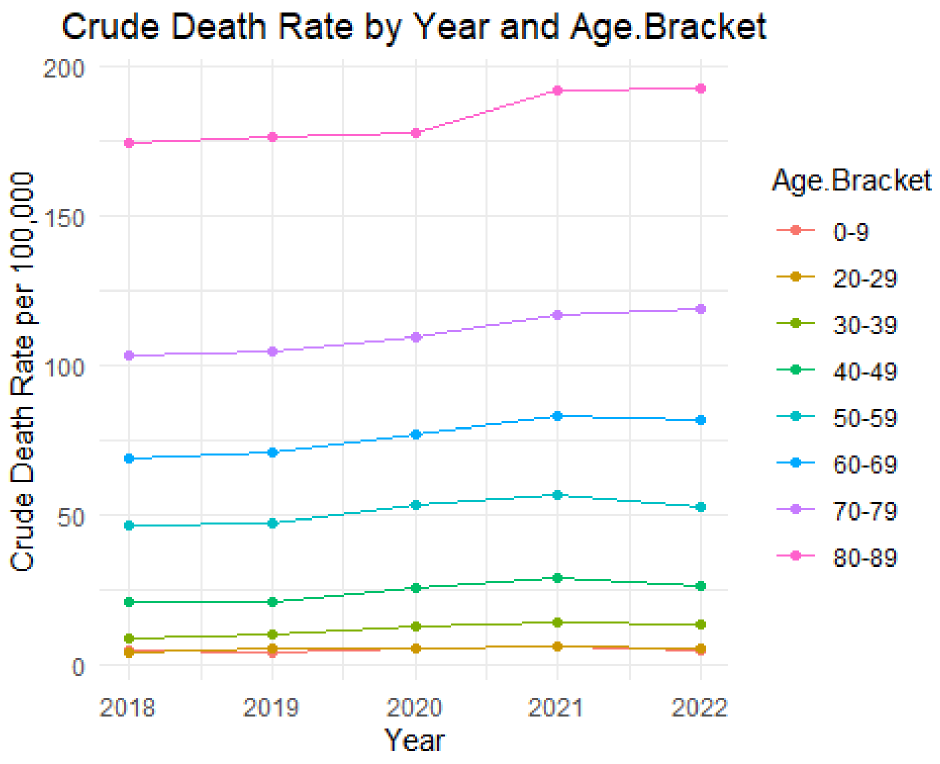
Question 2.9

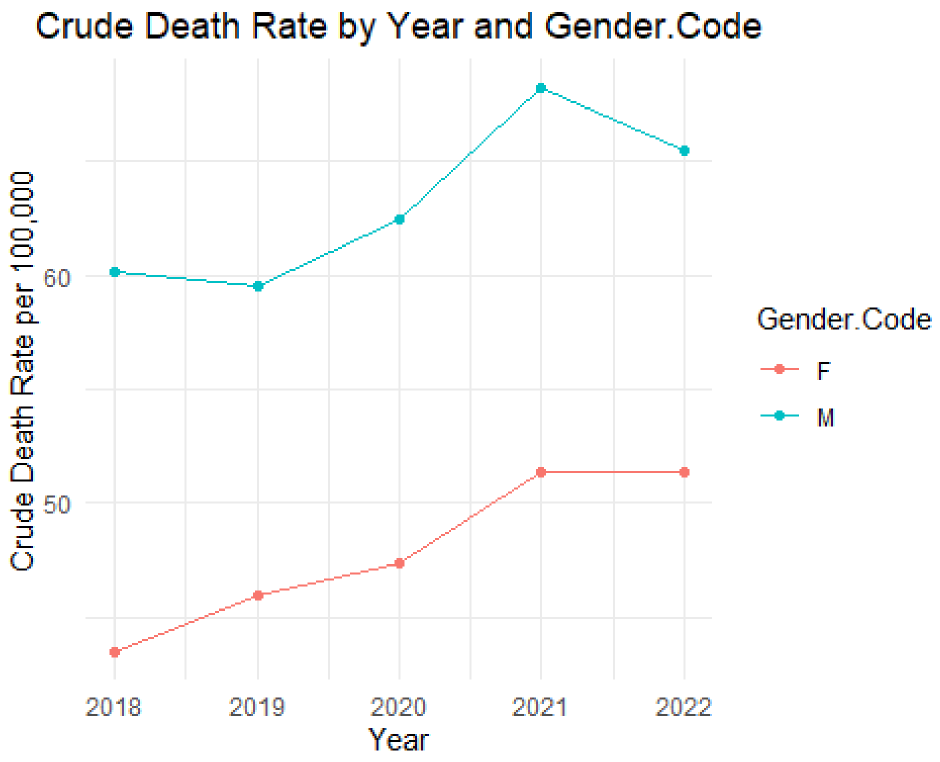
Graphing parallel time series: The mortality data in Section 2.4 are accessible from this site at the U.S. Centers for Disease Control and Prevention: wonder.cdc.gov. Download mortality data from this source but choose just one particular cause of death, and then make graphs similar to those in Section 2.4, breaking down trends in death rate by age, sex, and region of the country

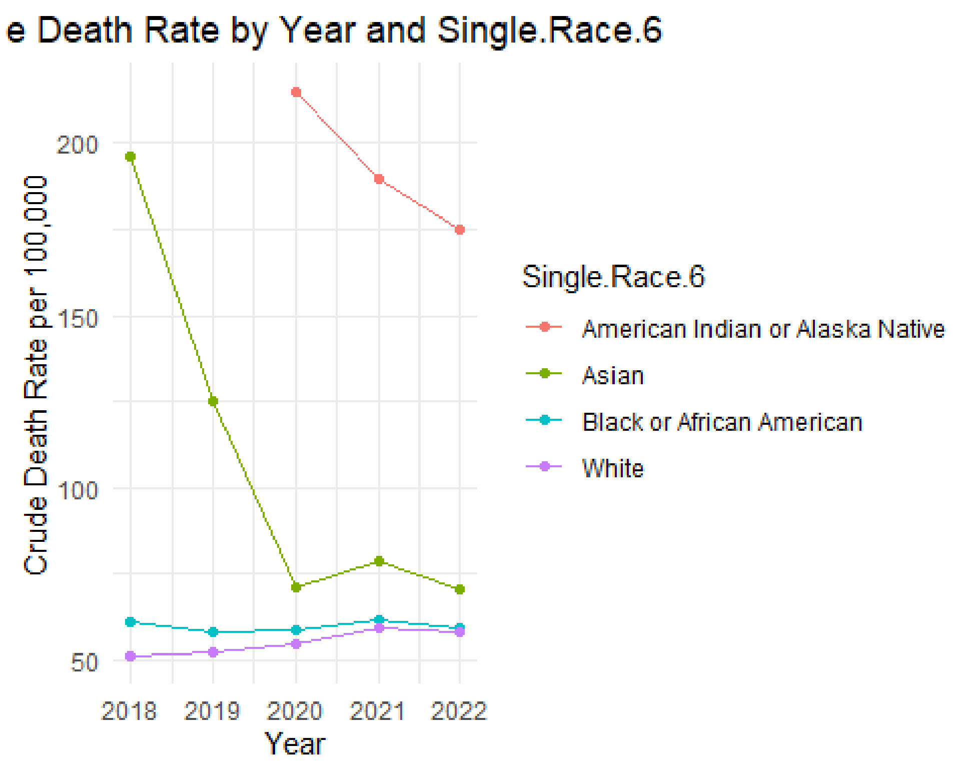
```
data_path <- "./Other
Data/Mortality/Underlying_Cause_of_Death_2018_2022_Single_Race.csv"

# Initial Data Prep
#data <- read.delim("C:\\Users\\mstoebe\\Downloads\\Underlying Cause of
Death, 2018-2022, Single Race (1).txt", sep = "\t", header = TRUE)
#filtered_data <- data[!is.na(data$`Year.Code`) & data$`Year.Code` != "", ]
#write.csv(filtered_data, data_path)

data <- read.csv(data_path)

data$Crude.Rate <- as.numeric(data$Crude.Rate)

## Warning: NAs introduced by coercion

filtered_data <- data[!is.na(data$Crude.Rate), ]

# Convert Population and Deaths to numeric, setting non-numeric values to NA
filtered_data$Population <- as.numeric(gsub(",", "",
filtered_data$Population))
filtered_data$Deaths <- as.numeric(gsub(",", "", filtered_data$Deaths))
filtered_data$Single.Year.Ages <- as.numeric(gsub("[^0-9]", "",
filtered_data$`Single.Year.Ages`))
filtered_data$Age.Bracket <- floor(filtered_data$Single.Year.Ages / 10) * 10
filtered_data$Age.Bracket <- factor(filtered_data$Age.Bracket,
                                    levels = seq(0, 90, by = 10),
                                    labels = paste(seq(0, 90, by = 10),
seq(9, 99, by = 10), sep = "-"))


summary_by_year <- aggregate(cbind(Deaths, Population) ~ Year, data =
filtered_data, sum, na.rm = TRUE)
summary_by_year$Crude.Rate <- (summary_by_year$Deaths /
summary_by_year$Population) * 100000


ggplot(summary_by_year, aes_string(x = "Year", y = "Crude.Rate")) +
  geom_line() +
  labs(title = "Crude Death Rate for Tummy Aches by Year",
       x = "Year",
       y = "Crude Death Rate per 100,000") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

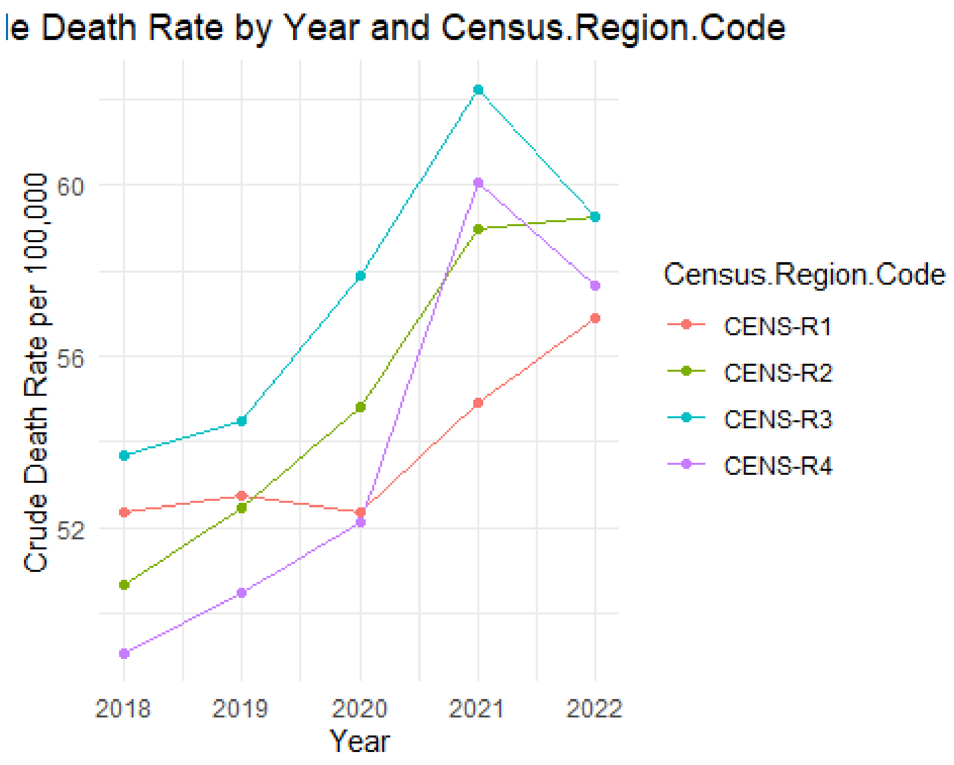


```
features <- c("Age.Bracket","Gender.Code", "Single.Race.6",
"Census.Region.Code" )


for (feature in features) {
  formula <- as.formula(paste("cbind(Deaths, Population) ~ Year +", feature))
  summary_by_feature <- aggregate(formula, data = filtered_data, sum, na.rm =
TRUE)
  summary_by_feature$Crude.Rate <- (summary_by_feature$Deaths /
summary_by_feature$Population) * 100000

  p <- ggplot(summary_by_feature, aes_string(x = "Year", y = "Crude.Rate",
color = feature)) +
    geom_line() +
    geom_point() +
    labs(title = paste("Crude Death Rate by Year and", feature),
         x = "Year",
         y = "Crude Death Rate per 100,000",
         color = feature) +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))


  # Print the plot
  print(p)
}
```


Crude Death Rate by Year and Age.Bracket
200
150
100
50
0
Crude Death Rate per 100,000
2018
2019
2020
2021
2022
Year
Age.Bracket
0-9
20-29
30-39
40-49
50-59
60-69
70-79
80-89

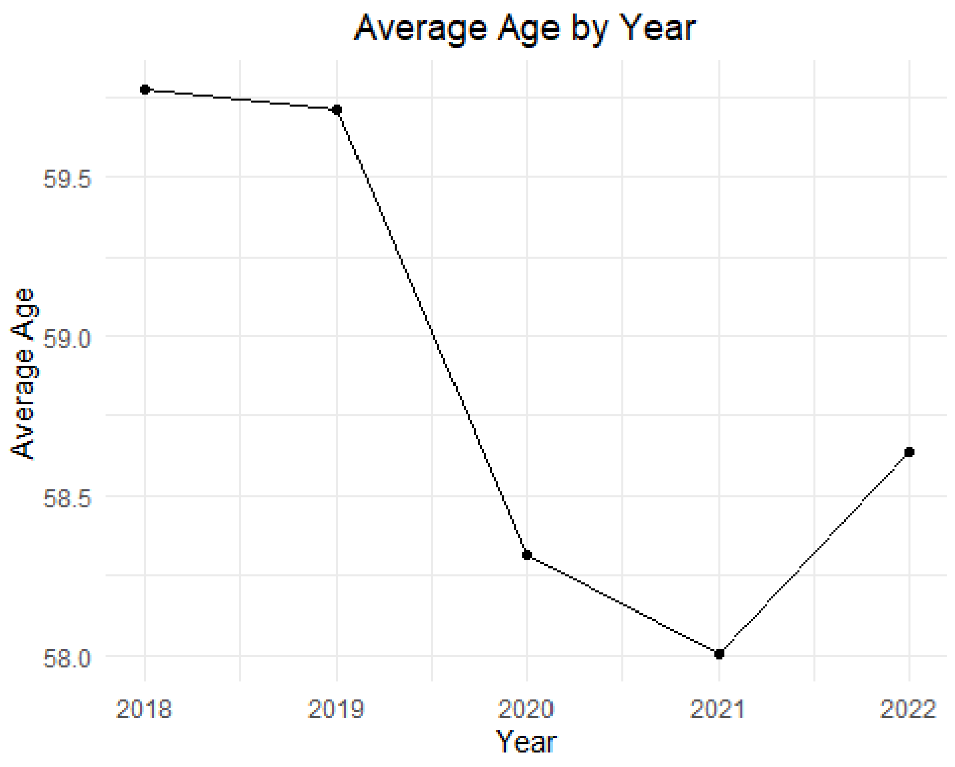

Crude Death Rate by Year and Gender.Code
60
50
Crude Death Rate per 100,000
2018
2019
2020
2021
2022
Year
Gender.Code
F
M





```
average_age_by_year <- aggregate(Single.Year.Ages ~ Year, data =
filtered_data, mean, na.rm = TRUE)
ggplot(average_age_by_year, aes(x = Year, y = Single.Year.Ages)) +
  geom_line() +
```

```r
  geom_point() +
  labs(title = "Average Age by Year",
       x = "Year",
       y = "Average Age") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

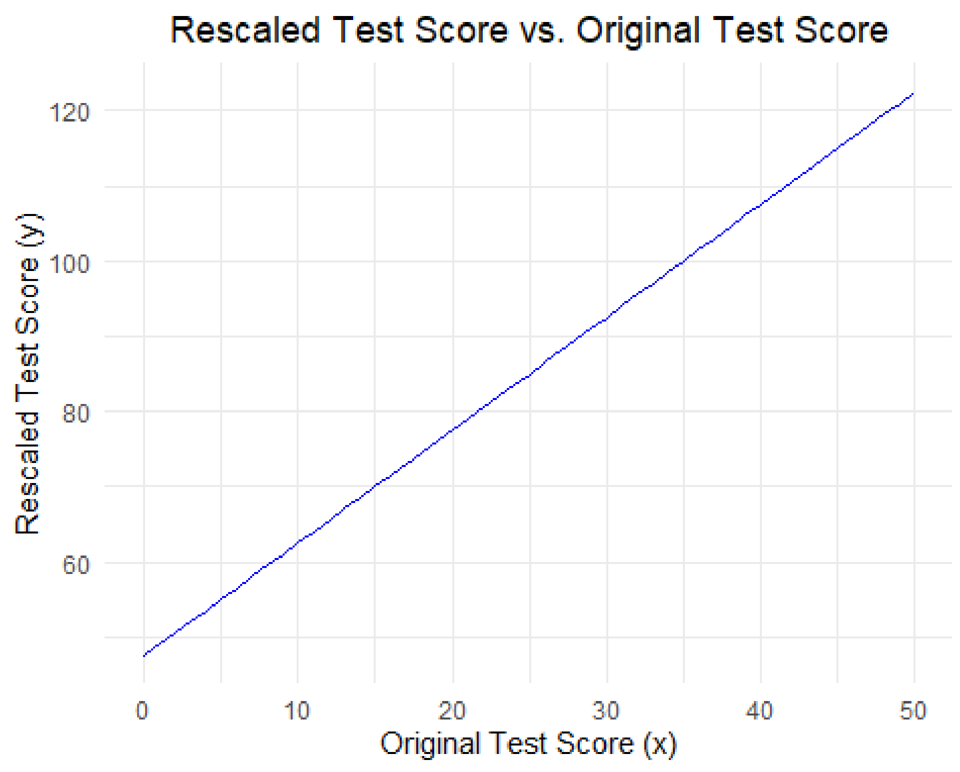


#Question 3.6

Linear transformations: A test is graded from 0 to 50, with an average score of 35 and a standard deviation of 10. For comparison to other tests, it would be convenient to rescale to a mean of 100 and standard deviation of 15. (a) Labeling the original test scores as x and the desired rescaled test score as y, come up with a linear transformation, that is, values of a and b so that the rescaled scores y = a + bx have a mean of 100 and a standard deviation of 15. (b) What is the range of possible values of this rescaled score y? (c) Plot the line showing y vs. x

```r
# Solution (a): Determine the linear transformation
x_mean <- 35
x_stdev <- 10
y_mean <- 100
y_stdev <- 15


b <- y_stdev / x_stdev
a <- y_mean - b * x_mean
```

```r
cat("The linear transformation is: y =", a, "+", b, "* x\n")

## The linear transformation is: y = 47.5 + 1.5 * x

# Solution (b): Determine the range of the new test scores
x_min <- 0
x_max <- 50

y_min <- a + b * x_min
y_max <- a + b * x_max

cat("The range of the rescaled test scores is from", y_min, "to", y_max,
"\n")

## The range of the rescaled test scores is from 47.5 to 122.5

# Solution (c): Plot y vs. x
data <- data.frame(x = seq(0, 50, by = 1))

# Apply the linear transformation to get y values
data$y <- a + b * data$x

# Plot y vs. x with centered title
ggplot(data, aes(x = x, y = y)) +
  geom_line(color = "blue") +
  labs(title = "Rescaled Test Score vs. Original Test Score",
       x = "Original Test Score (x)",
       y = "Rescaled Test Score (y)") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



#Question 3.7

Solution

You could also use -1.5 for your slope because the equation for slope is y_stdev = abs(b) * x_stdev

In 3.6, we assumed that the transformation would be postive, but it is also possible to do a negative transformation. The primary issue with this is that the interpretation of the scores changes as the lowest test scores now become the highest and the highest test scores become the lowest.

#Question 3.8

We know the correlation means and standard deviation which means that we can get the summed mean and standard deviation.

The new mean = (x_mean + y_mean) /2 = (69.1 + 63.7) /2 = 66.4

New STDEV =

```r
mean_x <- 69.1  # Mean height of husbands
mean_y <- 63.7  # Mean height of wives
sd_x <- 2.9     # Standard deviation of husbands' heights
sd_y <- 2.7     # Standard deviation of wives' heights
rho <- 0.3      # Correlation between husbands' and wives' heights
```

```
# Mean of the average height
new_mean <- (mean_x + mean_y) / 2

var <- sd_x^2 + sd_y^2 + 2*rho*sd_x*sd_y

new_sd = sqrt(var)

cat("Average height for husbands and wives:", new_mean, "inches\n")
```

```
## Average height for husbands and wives: 66.4 inches
```

```
cat("Standard deviation of the average height for husbands and wives:",
new_sd, "inches\n")
```

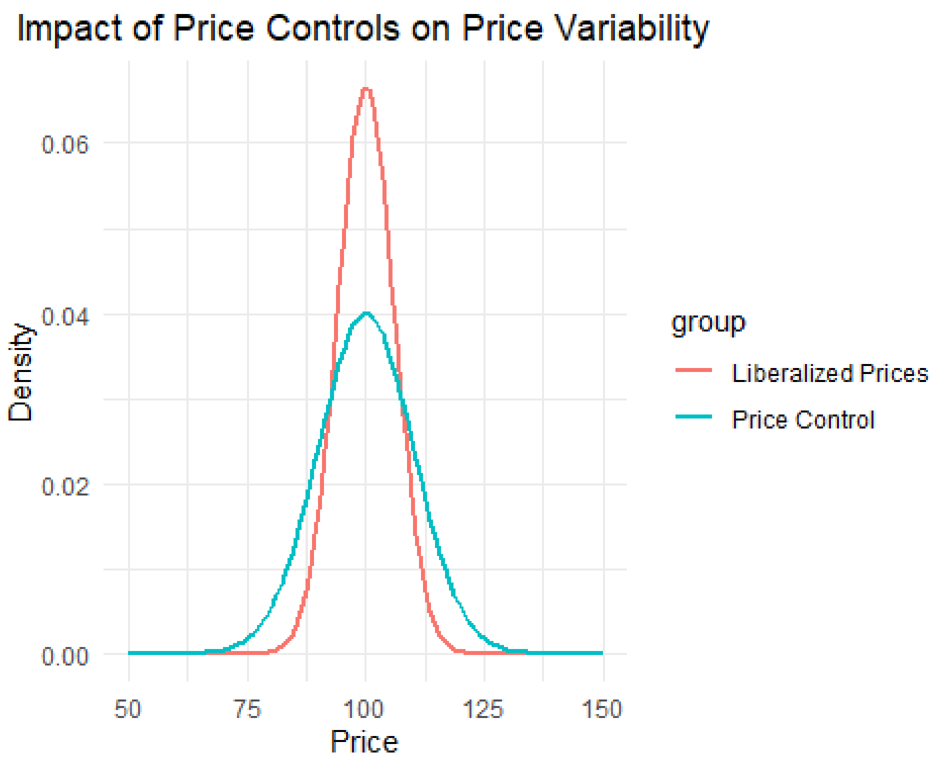
```
## Standard deviation of the average height for husbands and wives: 4.516415
inches
```

#Question 3.9

Answered using the study below on how price controlls affected price volitility
https://link.springer.com/article/10.1007/s00181-020-01953-w

```
# Load the necessary library
library(ggplot2)

# Define parameters
control_mean <- 100
treated_mean <- 100
control_sd <- 10
treated_sd <- control_sd * 0.6  # 60% reduction in variability for
liberalized prices
inflation_impact <- 0

# Adjust control SD for inflation
control_sd_inflation <- control_sd + inflation_impact

# Generate sequences of prices
x <- seq(50, 150, length.out = 100)

# Calculate the normal distribution for both groups
control_density <- dnorm(x, mean = control_mean, sd = control_sd_inflation)
treated_density <- dnorm(x, mean = treated_mean, sd = treated_sd)

# Create a data frame for plotting
data <- data.frame(
  x = c(x, x),
  density = c(control_density, treated_density),
  group = factor(rep(c("Price Control", "Liberalized Prices"), each = 100))
)
```

```r
# Plot the distributions
ggplot(data, aes(x = x, y = density, color = group)) +
  geom_line(size = 1) +
  labs(title = "Impact of Price Controls on Price Variability",
       x = "Price",
       y = "Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```


Impact of Price Controls on Price Variability
Density
Price
group
Liberalized Prices
Price Control