Assignment 2

Ariz Kazani

2024-07-16

Assignment 2

Name: Ariz Kazani

Student ID: 101311311

Notes

```
# Libraries

# NOTE: if you do not have any of the below libraries installed, un-comment the line and run it
# install.packages("rmarkdown")
library(rmarkdown)

# install.packages("ggplot2")
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.1
```

```
# install.packages("patchwork")
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.4.1
```

```
# install.packages("dplyr")
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.1

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# install.packages("tidyr")
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.4.1
```

```r
# install.packages("gapminder")
library(gapminder)
```

```
## Warning: package 'gapminder' was built under R version 4.4.1
```

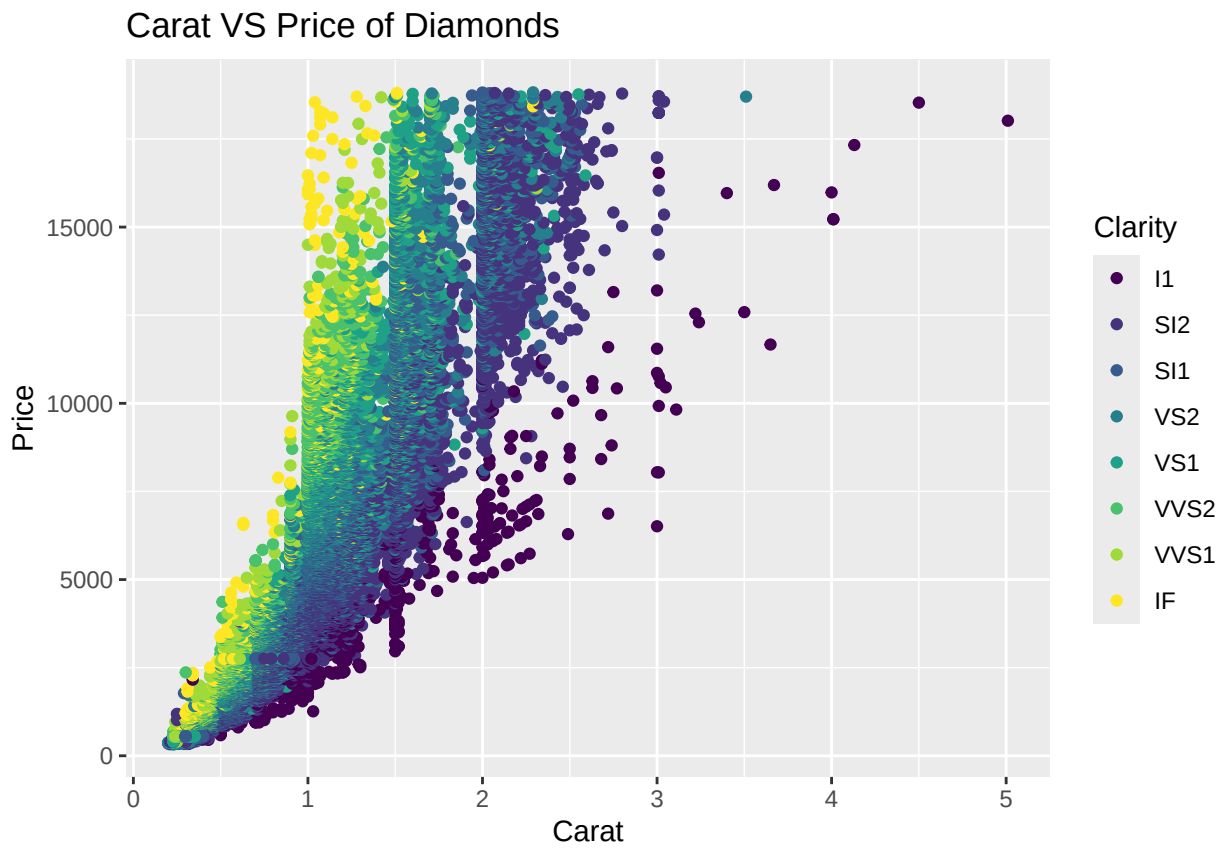
Solutions

---

1. Advanced ggplot2 Visualizations

A. Load the diamonds dataset from the ggplot2 package. Create a scatter plot of carat vs price with points colored by clarity.

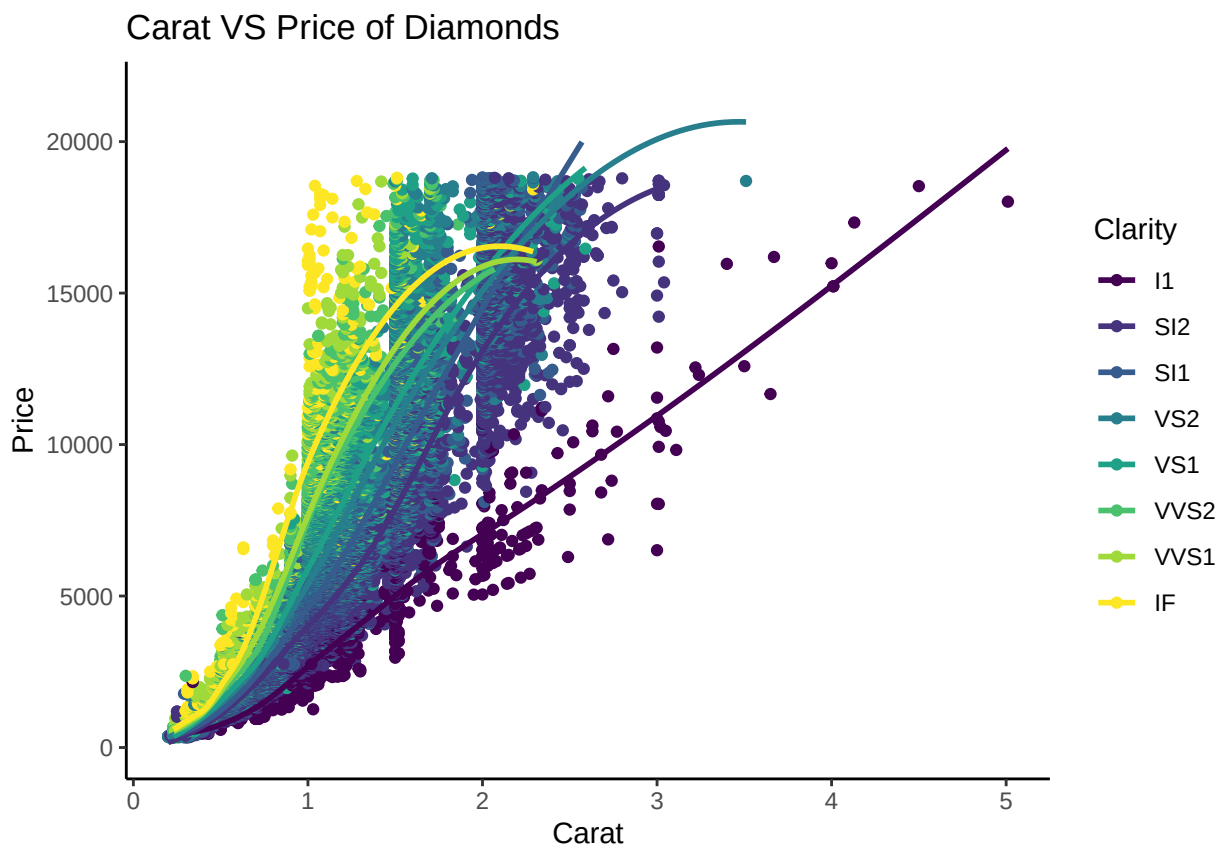
```r
data("diamonds")
scatPlot <- ggplot(diamonds, aes(x = carat, y = price, colour = clarity)) +
  geom_point() +
  labs(
    x = "Carat",
    y = "Price",
    title = "Carat VS Price of Diamonds",
    colour = "Clarity"
  )

scatPlot
```



B. Modify the scatter plot to include a smoothing line (e.g., LOESS) and customize the theme for better readability.

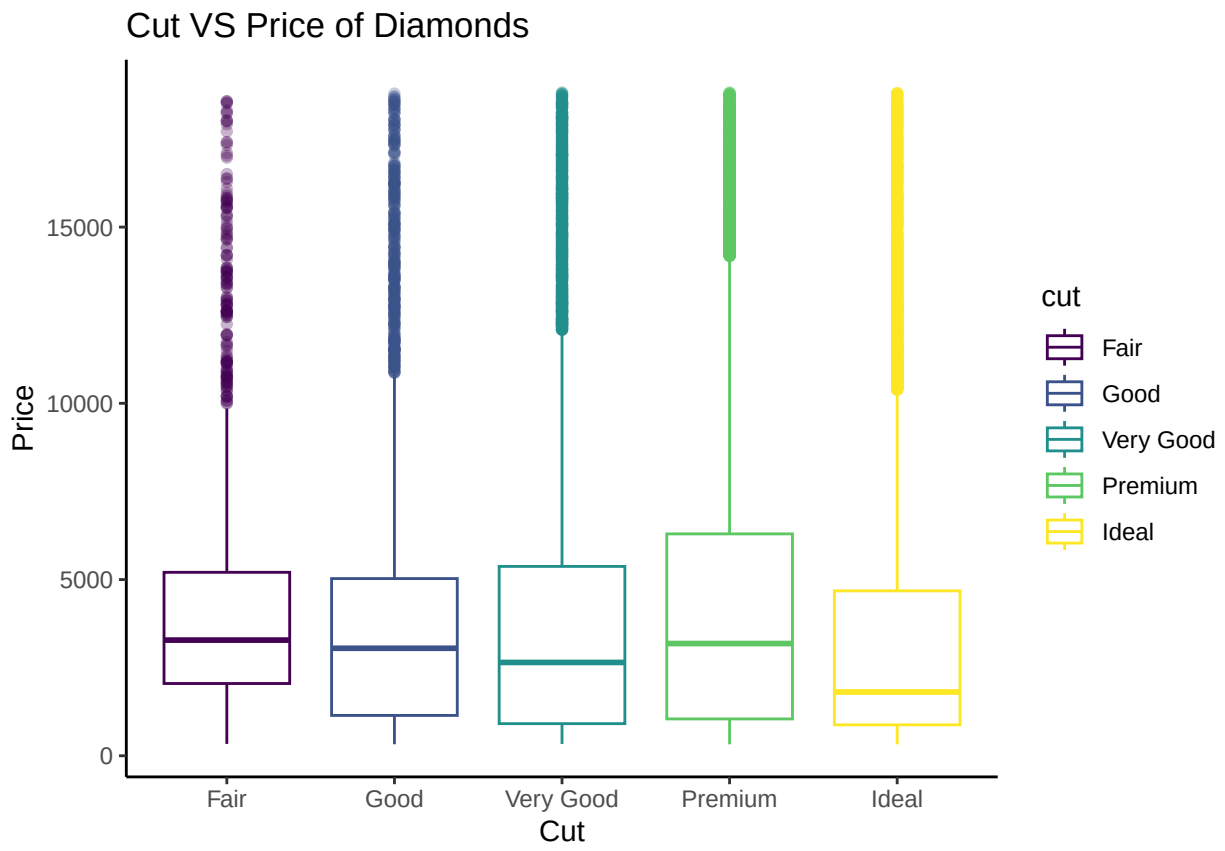
```
scatPlot <- scatPlot +
  geom_smooth(fill = NA, method = "loess") +
  theme_classic()

scatPlot
```

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



C. Create a boxplot of price by cut, with different fill colors for each cut.

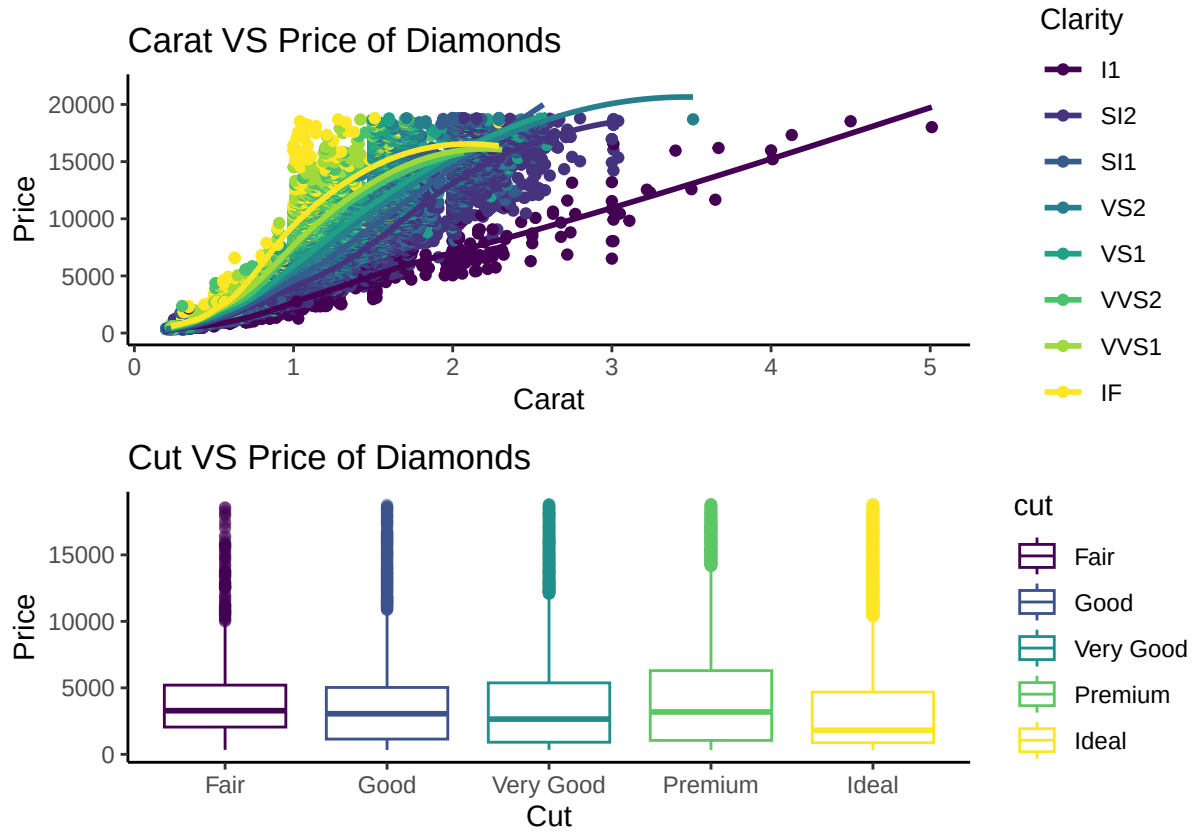
```
boxPlot <- ggplot(diamonds, aes(x = cut, y = price, colour = cut)) +
  geom_boxplot(alpha = 0.3) +
  labs(x = "Cut", y = "Price", title = "Cut VS Price of Diamonds") +
  theme_classic()

boxPlot
```



D. Combine the scatter plot and boxplot into a single visualization using patchwork.

```
combinedPlot <- scatPlot / boxPlot
combinedPlot
```

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



---

2. Advanced Group Manipulations

A. Load the mtcars dataset. Group the data by the number of cylinders and calculate the mean mpg for each group.

```
data("mtcars")

mtcarsDS <- mtcars

meanMpg <- tapply(mtcarsDS$mpg, mtcarsDS$cyl, mean)

meanMpg
```

```
##        4        6        8
## 26.66364 19.74286 15.10000
```

B. Add a column to the original dataset indicating whether each car's mpg is above or below the mean mpg of its cylinder group.

```
mtcarsDS$posMean <- ifelse(mtcarsDS$mpg > meanMpg[as.character(mtcarsDS$cyl)],
                           "above",
                           ifelse(mtcarsDS$mpg < meanMpg[as.character(mtcarsDS$cyl)], "below", "same"))

head(mtcarsDS)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb posMean
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4   above
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4   above
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1   below
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1   above
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2   above
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1   below
```

C. Create a summary table showing the mean and median hp and wt for each combination of cyl and gear.

```
summaryTable <- mtcars %>%
  group_by(cyl, gear) %>%
  summarise(
    meanHp = mean(hp),
    medianHp = median(hp),
    meanWt = mean(wt),
    medianWt = median(wt),
  )
```

```
## `summarise()` has grouped output by 'cyl'. You can override using the `.groups`
## argument.
```

```
summaryTable
```

```
## # A tibble: 8 x 6
## # Groups:   cyl [3]
##     cyl  gear meanHp medianHp meanWt medianWt
##   <dbl> <dbl>  <dbl>    <dbl>  <dbl>    <dbl>
## 1     4     3    97       97    2.46     2.46
## 2     4     4    76       66    2.38     2.26
## 3     4     5   102      102    1.83     1.83
## 4     6     3   108.     108.   3.34     3.34
## 5     6     4   116.     116.   3.09     3.16
## 6     6     5   175      175    2.77     2.77
## 7     8     3   194.     180    4.10     3.81
## 8     8     5   300.     300.   3.37     3.37
```

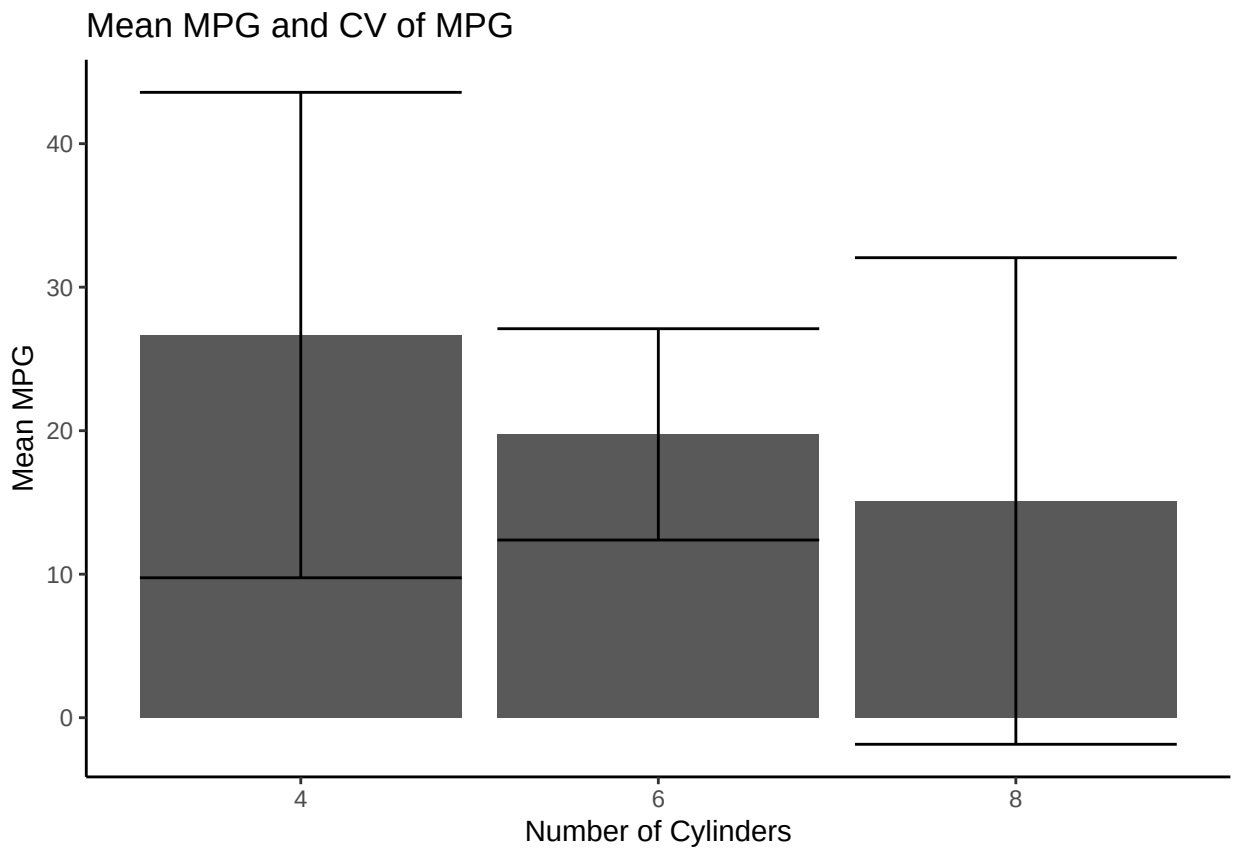
D. Write a function to calculate the coefficient of variation (CV) for a given numeric column and apply this function to mpg, hp, and wt for each cylinder group.

```r
getCV <- function(x) {
  cv <- sd(x) / mean(x) * 100
  return(cv)
}

cvs <- mtcars %>%
  group_by(cyl) %>%
  summarise(cvMpg = getCV(mpg),
            cvHp = getCV(hp),
            cvWt = getCV(wt))

cvs
```

```
## # A tibble: 3 x 4
##     cyl cvMpg  cvHp  cvWt
##   <dbl> <dbl> <dbl> <dbl>
## 1     4 16.9   25.3  24.9
## 2     6  7.36  19.8  11.4
## 3     8 17.0   24.4  19.0
```

E. Plot the mean mpg and CV of mpg for each cylinder group using a bar plot with error bars.

```r
plotData <- mtcars %>%
  group_by(cyl) %>%
  summarise(meanMpg = mean(mpg), cvMpg = getCV(mpg))

ggplot(plotData, aes(x = factor(cyl), y = meanMpg)) +
  geom_bar(stat = "identity") +
  geom_errorbar(aes(ymin = meanMpg - cvMpg, ymax = meanMpg + cvMpg), ) +
  labs(x = "Number of Cylinders", y = "Mean MPG", title = "Mean MPG and CV of MPG") +
  theme_classic()
```



---

3. Data Reshaping with tidyr

A. Load the airquality dataset. Reshape the dataset from wide to long format, using gather() for the measurements (Ozone, Solar.R, Wind, Temp).

```
data("airquality")
airqualityData <- airquality
airqualityData <- airqualityData %>% gather(key = "variable", value = "value", Ozone:Temp)

head(airqualityData)
```

```
##   Month Day variable value
## 1     5   1    Ozone    41
## 2     5   2    Ozone    36
## 3     5   3    Ozone    12
## 4     5   4    Ozone    18
## 5     5   5    Ozone    NA
## 6     5   6    Ozone    28
```

B. Reshape the dataset back to wide format using spread().

```
airqualityData <- airqualityData %>% spread(key = variable, value = value)

head(airqualityData)
```

```
##   Month Day Ozone Solar.R Temp Wind
## 1     5   1    41     190   67  7.4
## 2     5   2    36     118   72  8.0
## 3     5   3    12     149   74 12.6
## 4     5   4    18     313   62 11.5
## 5     5   5    NA      NA   56 14.3
## 6     5   6    28      NA   66 14.9
```

C. Use separate() to split the Month column into Month and Day columns (if it were combined), and then recombine them using unite().

```
# since its not combine the following code is commented out
# airqualityData <- airqualityData %>% separate(Date, c("Month", "Day"), sep="-")

airqualityData <- airqualityData %>% unite("Date", Month, Day, sep = "-")

head(airqualityData)
```

```
##   Date Ozone Solar.R Temp Wind
## 1  5-1    41     190   67  7.4
## 2  5-2    36     118   72  8.0
## 3  5-3    12     149   74 12.6
## 4  5-4    18     313   62 11.5
## 5  5-5    NA      NA   56 14.3
## 6  5-6    28      NA   66 14.9
```

D. Create a summary table showing the average values for each variable by month.

```
airqualityData <- airqualityData %>% separate(Date, c("Month", "Day"), sep =
                                                "-")

aqdSummary <- airqualityData %>%
  group_by(Month) %>%
  summarise(
    meanOzone = mean(Ozone , na.rm = TRUE),
    meanSolar.R = mean(Solar.R , na.rm = TRUE),
    meanTemp = mean(Temp , na.rm = TRUE),
    meanWind = mean(Wind , na.rm = TRUE),
  )

aqdSummary
```

```
## # A tibble: 5 x 5
##   Month meanOzone meanSolar.R meanTemp meanWind
##   <chr>     <dbl>       <dbl>    <dbl>    <dbl>
## 1 5          23.6        181.     65.5    11.6
## 2 6          29.4        190.     79.1    10.3
## 3 7          59.1        216.     83.9     8.94
## 4 8          60.0        172.     84.0     8.79
## 5 9          31.4        167.     76.9    10.2
```

---

4. Introduction to Probability

A. Simulate rolling a fair six-sided die 1000 times. Calculate the empirical probability of each outcome.

```
simulateDiceRole <- function(n) {
  outcome <- sample(1:6, n, replace = TRUE)
  return(outcome)
}

outcome <- simulateDiceRole(1000)

outcome
```

```
##    [1] 2 3 4 4 4 5 2 6 6 4 5 1 1 3 2 2 4 4 1 2 1 5 4 5 4 4 3 4 6 1 1 6 2 1 2 6 6
##   [38] 3 3 1 3 4 3 1 2 5 1 2 1 4 3 1 5 2 1 5 1 3 1 1 2 3 6 1 6 4 5 3 4 3 1 6 6 3
##   [75] 4 6 3 4 6 1 1 2 4 1 6 2 2 6 4 5 6 5 6 2 5 4 3 4 1 5 6 4 6 5 1 5 2 1 4 5 6
##  [112] 5 2 2 4 4 1 5 2 2 2 3 1 5 3 2 3 4 5 2 1 1 2 6 2 1 2 4 6 6 3 4 2 2 5 4 4 1
##  [149] 6 6 4 5 3 5 2 2 2 6 2 5 4 3 3 5 4 6 3 4 4 1 2 2 2 6 4 6 4 4 4 5 1 3 5 5 4
##  [186] 2 6 5 3 5 2 3 6 1 4 1 4 2 5 6 2 6 3 5 6 3 2 4 5 4 3 1 2 2 2 5 4 1 2 1 4 1
##  [223] 6 1 5 1 1 1 5 1 4 6 3 3 6 5 5 4 4 5 5 2 1 4 3 4 4 3 5 4 2 1 1 5 2 6 2 5 2
##  [260] 3 1 5 5 2 6 1 6 1 3 1 4 2 4 6 1 6 4 1 4 4 1 4 1 3 3 2 5 1 6 4 6 6 2 6 3 6
##  [297] 2 5 2 3 4 3 1 1 2 6 4 6 6 1 6 1 1 2 6 6 3 1 3 5 3 3 5 5 6 4 2 4 1 5 1 1 6
##  [334] 6 4 4 6 6 2 5 4 2 2 5 1 2 6 5 2 4 2 6 6 3 5 6 1 1 6 6 6 1 5 5 3 5 3 2 6 4
##  [371] 2 4 1 3 4 5 4 5 6 3 1 1 6 4 4 5 2 2 6 6 1 6 1 1 5 5 1 1 2 4 1 3 3 3 3 5 1
##  [408] 1 4 5 2 2 6 3 3 3 6 3 1 5 5 3 4 1 2 5 1 5 6 6 3 3 2 4 2 3 3 5 6 2 6 1 6 5
##  [445] 5 6 6 1 1 2 5 1 4 6 1 4 5 2 4 3 1 4 3 2 5 5 2 1 2 5 2 6 2 6 6 4 6 4 6 5 2
##  [482] 6 6 4 2 2 3 4 5 6 2 4 6 3 6 2 3 1 2 4 2 5 5 5 2 4 1 2 6 6 2 1 6 1 4 2 2 4
##  [519] 6 3 6 1 6 5 1 2 3 4 5 2 6 3 6 3 4 3 5 4 4 1 5 5 3 6 4 4 5 4 2 1 6 2 6 4 6
##  [556] 5 5 1 1 6 6 6 1 1 1 3 3 1 1 5 1 3 4 6 1 3 1 1 6 5 6 3 2 1 6 2 3 2 6 6 6 5
##  [593] 5 2 1 3 5 6 3 6 4 2 1 1 6 6 4 4 5 4 2 6 1 2 1 5 2 3 1 2 4 5 5 2 5 1 6 6 3
##  [630] 4 4 4 5 3 3 6 6 1 4 5 1 5 1 2 1 2 6 3 4 2 6 6 3 2 1 3 6 5 5 3 1 4 1 6 1 5
##  [667] 3 5 1 4 1 2 4 4 4 4 1 3 2 3 1 5 6 4 3 1 3 4 4 5 6 1 6 2 5 2 2 6 2 4 5 1 6
##  [704] 2 6 4 6 5 4 1 2 2 4 1 1 1 1 6 2 4 5 6 1 6 1 5 1 2 3 3 1 2 3 1 1 5 6 3 3 4
##  [741] 6 3 6 2 1 3 5 1 4 6 2 4 4 1 1 5 1 2 5 3 5 6 1 4 5 1 5 6 2 3 5 5 2 6 6 4 4
##  [778] 1 1 3 5 4 1 1 5 3 2 4 2 3 3 2 3 4 6 1 4 3 1 1 6 1 4 2 1 4 6 6 3 4 1 1 2 2
##  [815] 4 6 6 5 1 3 3 2 3 3 4 2 3 2 3 1 4 5 6 5 5 1 5 4 5 6 6 1 6 1 5 6 3 4 4 2 3
##  [852] 5 2 4 4 3 4 6 3 4 3 4 5 4 2 6 6 2 2 1 3 1 2 5 5 2 5 6 4 6 3 1 2 2 3 1 3 1
##  [889] 2 2 5 5 4 3 6 6 1 2 5 3 6 1 3 1 5 2 4 4 2 5 3 6 4 3 3 3 3 4 3 5 4 4 3 2 1
##  [926] 2 2 6 2 5 4 5 2 4 4 3 4 4 1 3 6 1 3 4 4 1 3 1 4 2 5 6 2 3 4 3 5 1 1 3 3 4
##  [963] 2 5 5 6 2 1 5 3 2 1 3 2 2 4 2 1 6 1 3 2 6 5 6 6 4 2 5 5 5 4 5 4 3 6 3 4 2
```

```
## [1000] 3
```

```
EP <- table(outcome) / 1000

EP
```

```
## outcome
##     1     2     3     4     5     6
## 0.184 0.168 0.149 0.172 0.154 0.173
```

B. Simulate drawing a card from a standard deck of 52 cards 1000 times. Calculate the empirical probability of drawing an Ace.

```
# lets assume each suit goes from ace to king, 1-13 respectivly
simulateCardDraw <- function(n) {
  outcome <- sample(1:52, n, replace = TRUE)
  return(outcome)
}


outcome <- simulateCardDraw(1000)

outcome
```

```
##    [1] 48  9 19 11  3  3 15  3 12  1 37 46 25 14 51 12 46 20 39 28 45 49 50  1
##   [25]  1 23 12 18  1  7 35 43 25  3 38 10 20 19 36 19 14  9  8 50 17 48 38 24
##   [49]  2 36 11 17 23 48 31 52  8 49 21 48 30 22 14 24  8  7 39 34 48 40 13  1
##   [73] 12 24 23 28  5 38  3  6 12 33 49 44  4 35 43  5 29 15 31  3 15 20  6 45
##   [97] 37 40 22 19 29 24 27 33 46 41 31 15  1 36 41 41 32 48 21 16 38 16 19 47
##  [121] 10  1 21 12 19 27 36  3 13 21 20 52 11 33 36 50 38 21 13 51 48 13 31 33
##  [145] 38 36 47 37 29 26 13 48 51 46  2 29 39 46 31 22 28 46  4 45 44 31 18 41
##  [169] 20  4  1 42 29 23 15 41  8 42  1 22 29 26 37 30 32 45 39 24 42 38 40 18
##  [193] 52 25 41 10 35 37  6 33 18 47  1 11 45 35 47 17 43 33  6 52 12  7 35 26
##  [217] 15  4 18 23 21 35  1 17 33 50 26 50 52 28 47 52  8 13 22 36 33 43 25 44
##  [241]  4  7 52 13 44 11 10 19 21  9 10 29 16 15 32 43 33 45 38 15 31 23 41  1
##  [265] 11 30 38  9  7 15  7 42  5  6 21 27 28 12 37 39 11  7 23 28 51  5 37 26
##  [289]  4 13 51 12  4 25 14 29 18 37 43 15 40 24  5 47 27 35 19 18 28 32 49  8
##  [313] 22 16 24 34  7 10 28  5 26 11 12 10 11 35  9  4  9 21 17 48 37 17 50 48
##  [337] 33 18 49  9 12 28 28 29 21 39 34 40 43 30 17  7  6 10  2 51  2 35 27 24
##  [361] 43 51  5 36  6 49 33  6 48 21 44  7 51 14 21 13  4 24 17 17 51 24 30 40
##  [385] 38 38 26 39 44 36 26 51  3 48 14 45 21 49 28 44 47 37  2  2 35  7 13  6
##  [409] 42 27  5  6  9 44  7 33  5 38 32 41  4 20 52 26 15 20 52 15 33 46 42 13
##  [433] 15 31 31 27 27  8 29 28 11  2 24 16 48 11 38 44 15 22  6 49 52 11 32 14
##  [457] 47 42  4 29 39 16 39 22 29 31 28  4 44 19 24 15 15  8  2 22  9 36 33 35
##  [481] 21 51 35 40 13 50 40 20 38  1 11 26 38 47 12  1  3 15 41 47 22 44 14 38
##  [505]  3 47 37 47 15 11 43 29 21 27 43 42 26 31 42 49  4 48 31  9 13 32  6 51
##  [529] 22 26 19 33 45 14 41 48 40 18 30 33  1  5  6 25 14 46 48  4 23 48 25 44
##  [553] 20 31 16 11 10 16 45 37 22 27 32 40 21 40  2  7 21 10 29 21 36 34 39 13
##  [577] 30 42 28 18 18 49 31 50 15  2 44 42 26 29 45 42  8 10 52  1 49 26 46  4
##  [601]  7  9 21  8 23 31 12 10 45 42 42 19 24 15  6 13  3  7 51 52 52 18 30 28
##  [625] 24  7 41 41 25 35  2 13 36 28 24 30 27 14  3 42  8  5 10 17 45 15 14 25
```

```
##  [649] 29 32 41 37 14 17 42 47 21 12 23 17 33 15  4  2 34 42 37 11 40 21  3 52
##  [673]  3  8 21 16 12  7 50 20 36 48 36 22 36 12 34 43 11 11 23 13  2 49 27 33
##  [697] 38 12 19  3 11 35 39 40 17  3 24 25 28 19 29 49 13 39 50 21 16  3 17 22
##  [721] 16 23 50 19 11 11 17  3 26  6 49 52 35 17 37 44 35 22  6 34 29 52 29  2
##  [745] 50 36 16 39  8  6  2 35 11 40 11  3 18 16 37 24  5 17 15 21 25 30 44 41
##  [769] 28 31 17 17  2 44 24  2 38 20 32 13 35 25 43 24 49  6  7  4 38 24 30 16
##  [793] 42 52  5 21 48 47 39  7  2  5  5 29 43 32 16 12 13 44 48 16 44 26 41 19
##  [817] 28 18 10 20 48 35 42 32 38 38  4 47 32 45 46 39 29  2 26  9  9 24 48  9
##  [841]  8  3  3 41 43 40 49 43 24 25 28 12 46 31 41 36 27 13 27 48  3 10 24  9
##  [865] 22 38 37 11 43 12 25 14  9 35 20 47  1 45  4 43 18 14 31  6 41  9 18 45
##  [889]  3  6 11 37 49 21 14 15  4 10  3 23 22 19 45 25 14 47 27 42 51 39 16 45
##  [913] 40 16 11 40 14  6 47 22 32 40 50 30 32 14 26 43 10  3 48 30  9 18  6 34
##  [937] 38 32  8 22 13 42 16 31 37  3 27 44 50  4 25 10 35  5  9 34 41 31 51 13
##  [961] 43 15 19  8 42  2 47  2 25 23 21 46 10 23 29 26 19 15 40 18 17  2 32 36
##  [985] 37 51  5 10 24 51  1  7 34 29 41 14 40 43 35 23
```

```r
#find the card value, from 1-13
outcome <- ifelse(outcome %% 13 == 0, 13, outcome %% 13)

EP <- sum(outcome == 1) / 1000

cat("Empirical probability of drawing an Ace:", EP)
```

```
## Empirical probability of drawing an Ace: 0.074
```

C. Use the binomial distribution to calculate the probability of getting exactly 5 heads in 10 flips of a fair coin. Repeat for getting 5 or more heads.

```r
prob5Heads <- dbinom(x = 5, size = 10, prob = 0.5)
cat("The probability of getting exactly 5 heads in 10 flips of a fair coin is:",
    prob5Heads,
    "\n")
```

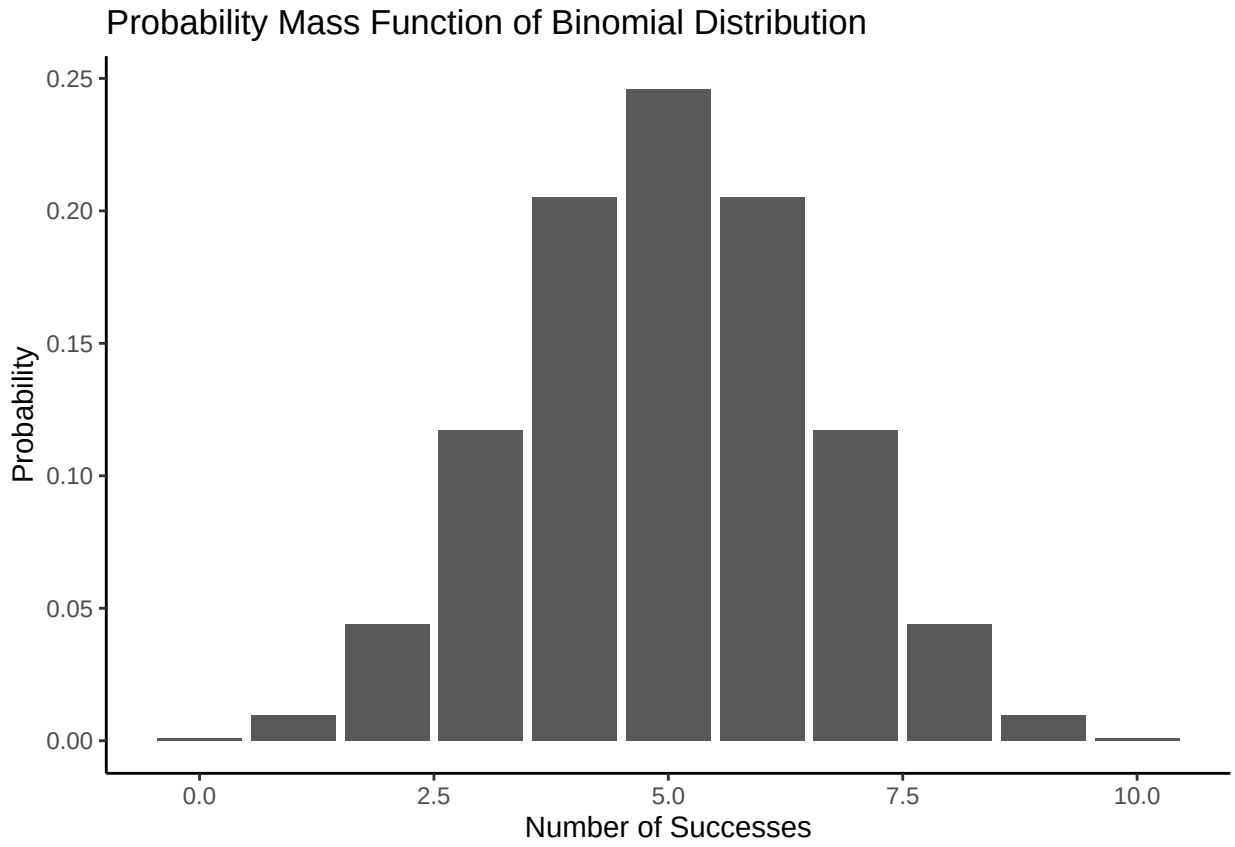
```
## The probability of getting exactly 5 heads in 10 flips of a fair coin is: 0.2460938
```

```r
prob5MoreHeads <- pbinom(4,
                         size = 10,
                         prob = 0.5,
                         lower.tail = FALSE)
cat("The probability of 5 or more heads in 10 flips of a fair coin is:",
    prob5MoreHeads,
    "\n")
```

```
## The probability of 5 or more heads in 10 flips of a fair coin is: 0.6230469
```

D. Generate a plot showing the probability mass function (PMF) of a binomial distribution with parameters n = 10 and p = 0.5.

```r
x <- 0:10

pmf <- dbinom(x = x, size = 10, prob = 0.5)

df <- data.frame(x, pmf)

ggplot(df, aes(x = x, y = pmf)) +
  geom_bar(stat = "identity") +
  labs(title = "Probability Mass Function of Binomial Distribution", x = "Number of Successes", y = "Pr
  theme_classic()
```



---

5. Advanced Data Manipulation and Visualization

A. Load the iris dataset and create a summary table showing the mean, median, and standard deviation of each numerical variable grouped by Species.

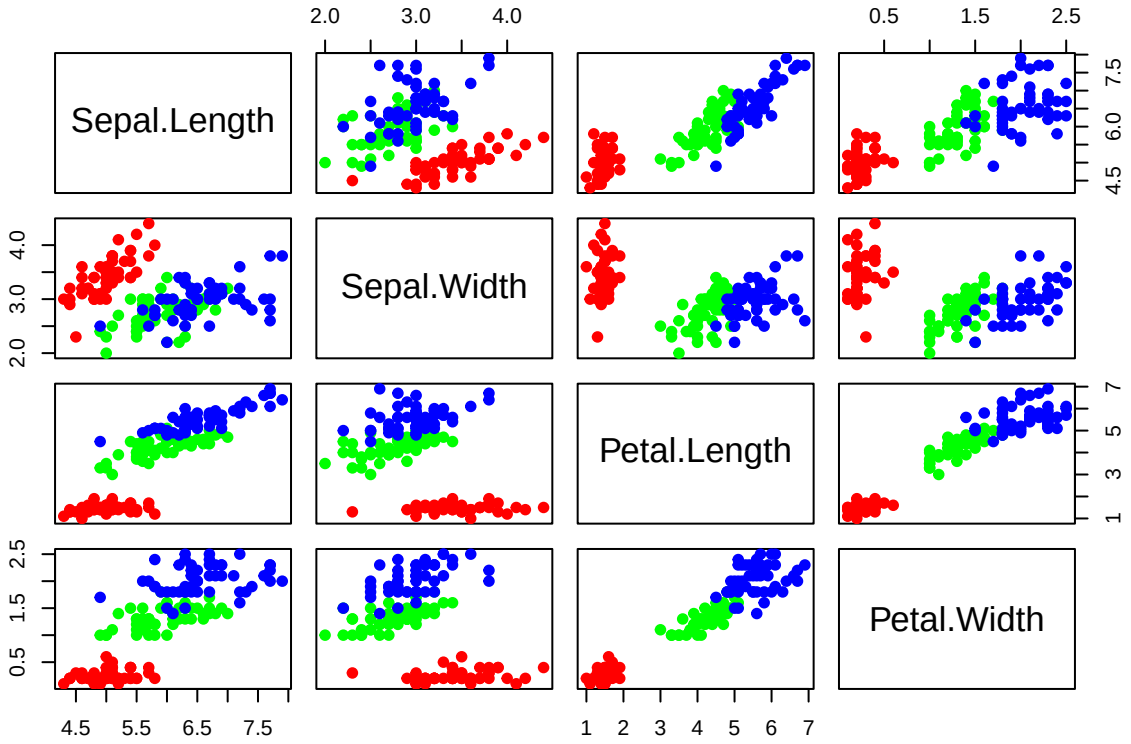
```r
data(iris)
irisData <- iris
irisData <- irisData %>% group_by(Species) %>% summarise(
  meanSepal.Length = mean(Sepal.Length),
```

```
  medianSepal.Length = median(Sepal.Length),
  sdSepal.Length = sd(Sepal.Length),

  meanSepal.Width = mean(Sepal.Width),
  medianSepal.Width = median(Sepal.Width),
  sdSepal.Width = sd(Sepal.Width),

  meanPetal.Length = mean(Petal.Length),
  medianPetal.Length = median(Petal.Length),
  sdPetal.Length = sd(Petal.Length),

  meanPetal.Width = mean(Petal.Width),
  medianPetal.Width = median(Petal.Width),
  sdPetal.Width = sd(Petal.Width),
)

irisData
```

```
## # A tibble: 3 x 13
##   Species    meanSepal.Length medianSepal.Length sdSepal.Length meanSepal.Width
##   <fct>                 <dbl>              <dbl>          <dbl>           <dbl>
## 1 setosa                 5.01                5           0.352            3.43
## 2 versicolor             5.94                5.9         0.516            2.77
## 3 virginica              6.59                6.5         0.636            2.97
## # i 8 more variables: medianSepal.Width <dbl>, sdSepal.Width <dbl>,
## #   meanPetal.Length <dbl>, medianPetal.Length <dbl>, sdPetal.Length <dbl>,
## #   meanPetal.Width <dbl>, medianPetal.Width <dbl>, sdPetal.Width <dbl>
```

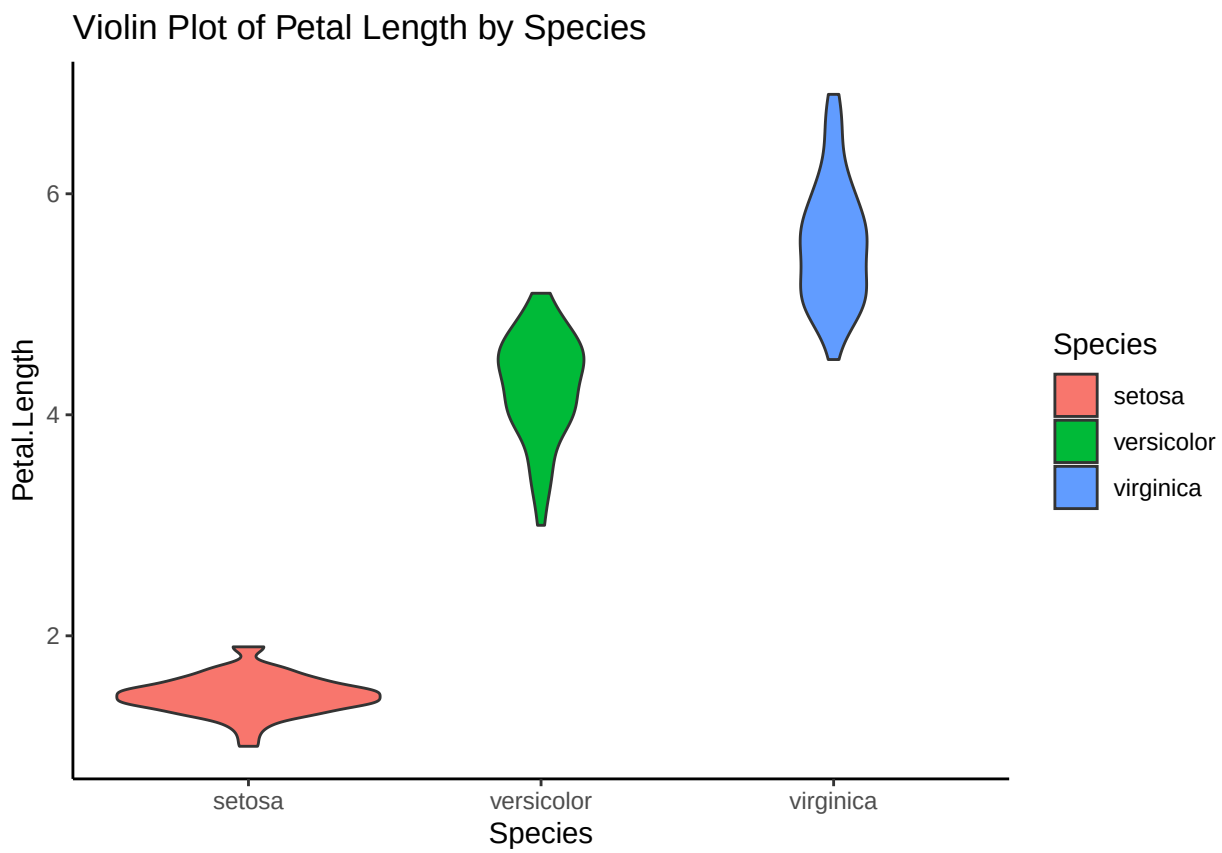
B. Create a pairwise scatter plot matrix using the pairs() function for the iris dataset colored by Species.

```
colours <- rainbow(length(unique(iris$Species)))
pairs(iris[1:4], pch = 19, col = colours[iris$Species])
```



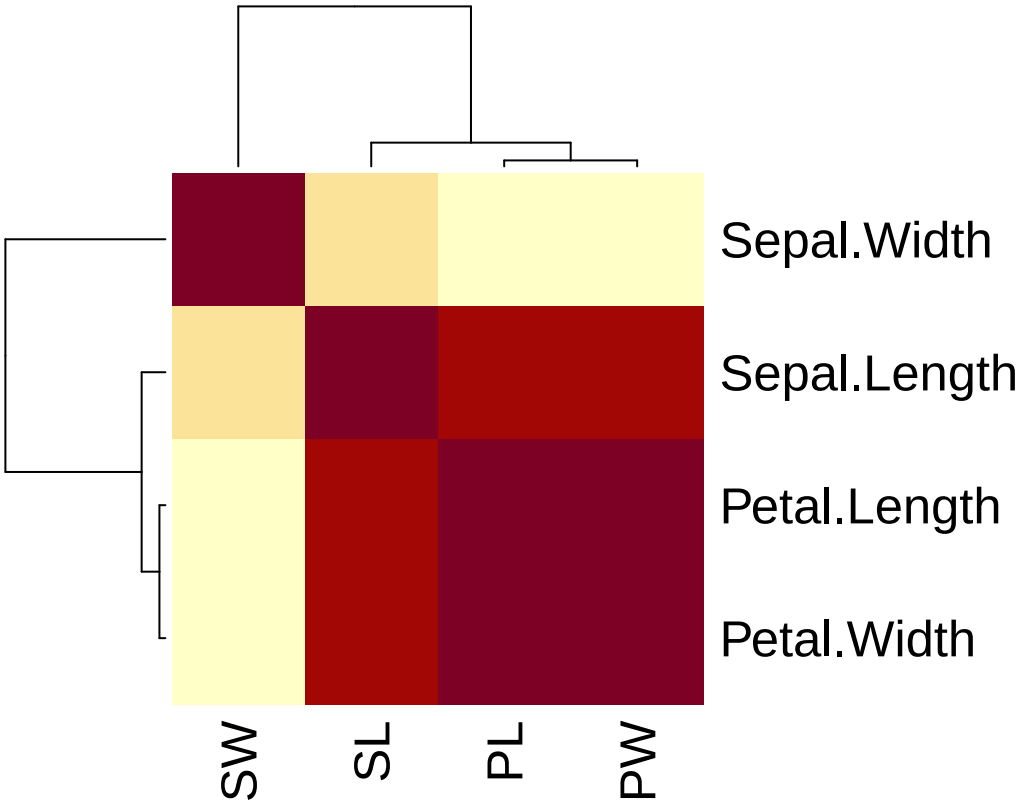
C. Use ggplot2 to create a violin plot for Petal.Length grouped by Species.

```
ggplot(iris, aes(x = Species, y = Petal.Length, fill = Species)) +
  geom_violin() +
  labs(title = "Violin Plot of Petal Length by Species") +
  theme_classic()
```



D. Create a heatmap of the correlation matrix for the numerical variables in the iris dataset.

```
correlationMatrix <- cor(iris[, 1:4])

heatmap(correlationMatrix,
        symm = TRUE,
        labCol = c("SL", "SW", "PL", "PW"))
```



E. Write a short analysis (5-7 sentences) interpreting the results from the summary table, scatter plot matrix, violin plot, and heatmap.

For all three species we can see that the mean and median sepal and petal length and width are similar, indicating a normal distribution. The virginica has the highest mean sepal length and the setosa has the smallest with the revers being true for the length. The petal for the petal length and width the virginica is the highest, followed by versicolor,and lastly setosa. The pairwise plot simple confirst the observations we saw with the data for the sepal and petal lengths, it also provides a clearer view of some of the points, we can see the the virginica and versicolor's points are close to one another and overlap with the setosa being clearly further apart. From the violin plot we can see that the setosa has a very normal distribution where versicolor and virginica are slightly skewed left and right respectively. From the heat map we can see that there is a very strong coralation between Petal Length and Petal Width and a moderate coralation between Sepal Length and Petal length and Petal width. There is also a very small coralation between Sepal Length and Width.

---

6. Data Reshaping and Aggregation

A. Load the gapminder dataset from the gapminder package. Reshape the dataset to long format, focusing on the variables year and gdpPercap.

```r
data(gapminder)

gapminderDS <- gapminder


gapminderDS <- gapminderDS %>%
  gather(key = "variable", value = "value", year, gdpPercap)

head(gapminderDS)
```

```
## # A tibble: 6 x 6
##   country     continent lifeExp      pop variable value
##   <fct>       <fct>       <dbl>    <int> <chr>    <dbl>
## 1 Afghanistan Asia         28.8  8425333 year      1952
## 2 Afghanistan Asia         30.3  9240934 year      1957
## 3 Afghanistan Asia         32.0 10267083 year      1962
## 4 Afghanistan Asia         34.0 11537966 year      1967
## 5 Afghanistan Asia         36.1 13079460 year      1972
## 6 Afghanistan Asia         38.4 14880372 year      1977
```

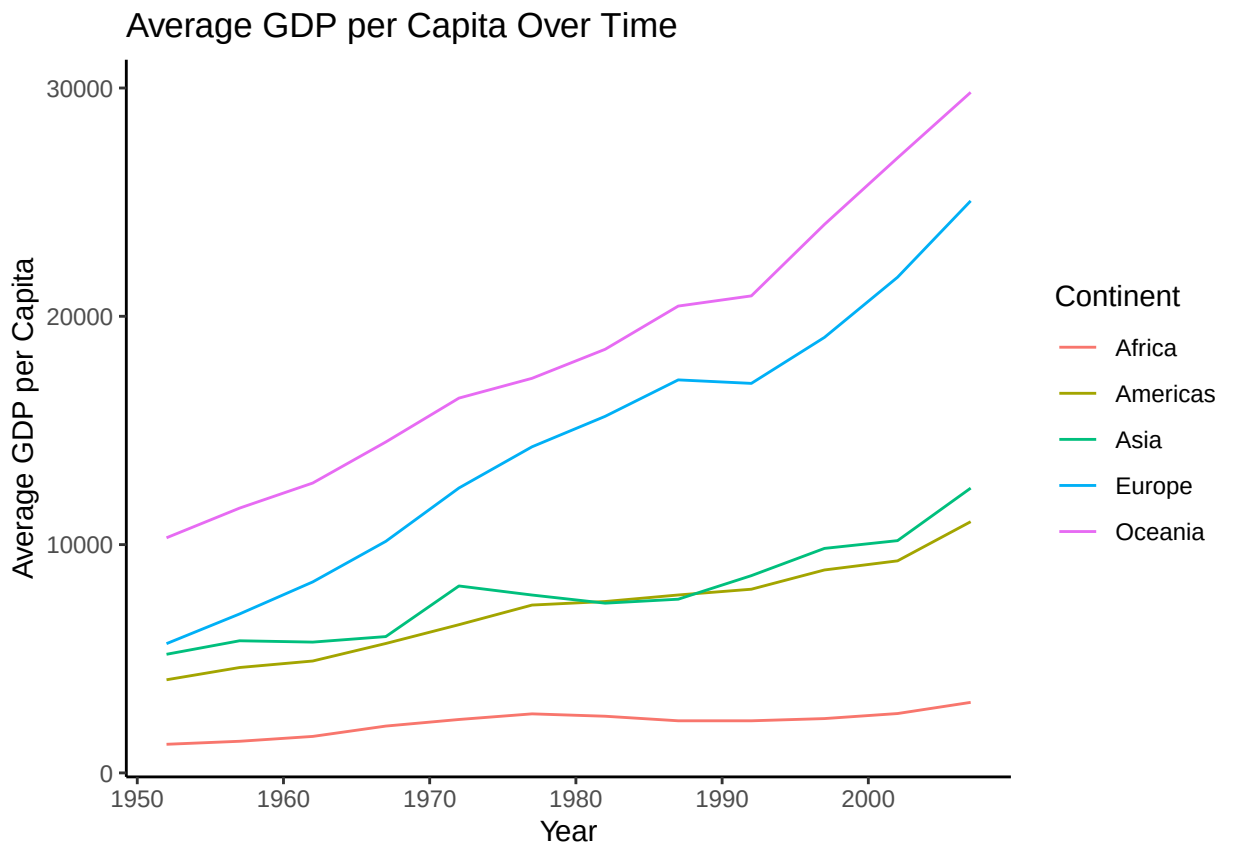
B. Aggregate the data to calculate the average gdpPercap by continent and year.

```r
avgGDP <- aggregate(gdpPercap ~ continent + year, data = gapminder, FUN = mean, na.rm = TRUE)

head(avgGDP)
```

```
##   continent year gdpPercap
## 1    Africa 1952  1252.572
## 2  Americas 1952  4079.063
## 3      Asia 1952  5195.484
## 4    Europe 1952  5661.057
## 5   Oceania 1952 10298.086
## 6    Africa 1957  1385.236
```

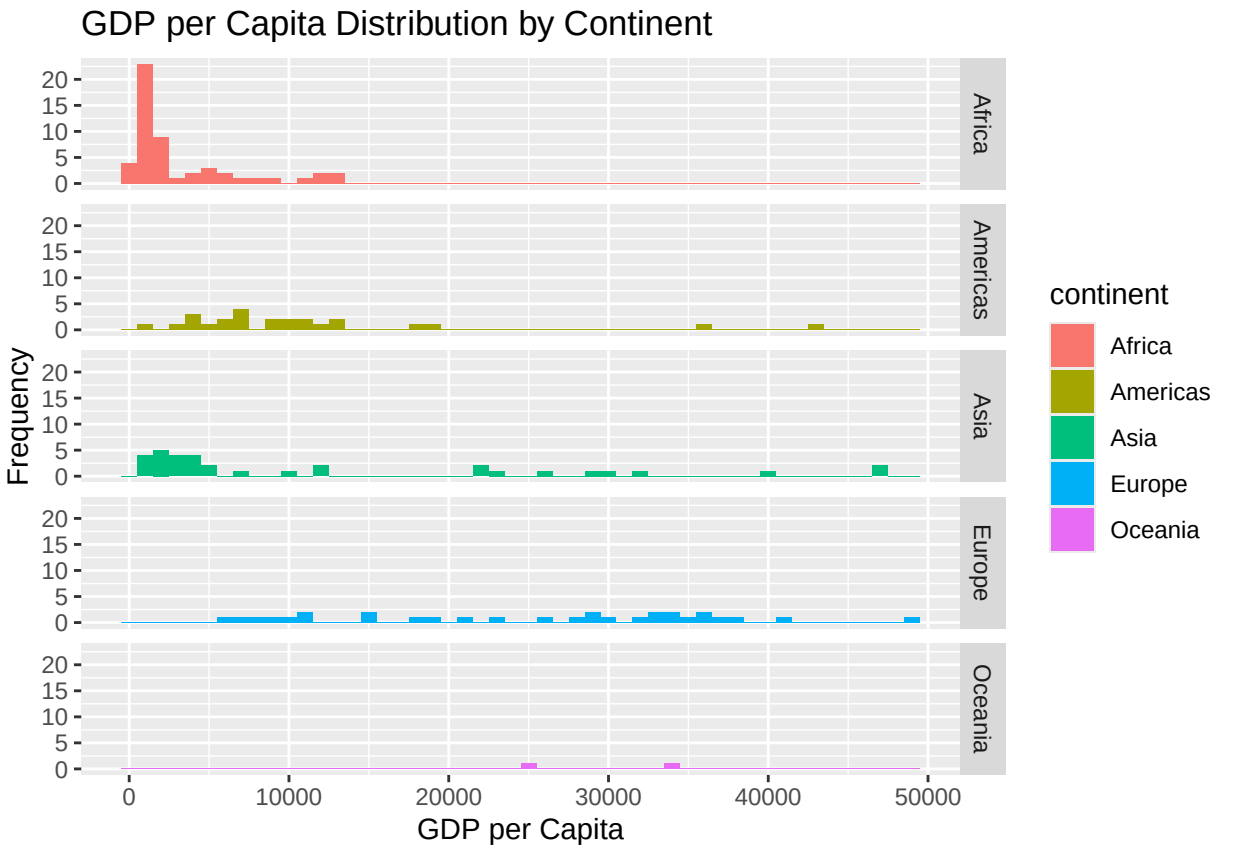
C. Create a line plot of the average gdpPercap over time for each continent.

```r
ggplot(avgGDP, aes(x = year, y = gdpPercap, color = continent)) +
  geom_line() +
  labs(
    title = "Average GDP per Capita Over Time",
    x = "Year",
    y = "Average GDP per Capita",
    color = "Continent"
  ) +
  theme_classic()
```



D. Create a faceted plot showing gdpPercap distributions by continent for the most recent year in the dataset.

```
maxYear <- max(gapminder$year)
gapminderDS <- gapminder %>%
  filter(year == maxYear)

ggplot(gapminderDS, aes(x = gdpPercap, fill = continent)) +
  geom_histogram(binwidth = 1000) +
  facet_grid(continent ~ .) +
  labs(
    x = "GDP per Capita",
    y = "Frequency",
    title = paste("GDP per Capita Distribution by Continent")
  )
```



E. Write a detailed report (6-8 sentences) analyzing the trends and patterns observed in the plots.

From the data we can see that as the years increase the life expectancy and GDP increase. We can also see that Africa's GDP per capita has not increased much and Eurome, and Asia's increased a Lot. Africas GDP per capita is very low for all the countries. The Americas, Asia and Europe have very spread out. Europe has the highest country with GDP per capita. Ocianas lowest GDP per capita is higher than Africas highest GDP per capita. The GDP of the Americas and Asia is comparable.

---

7. Probability

A local fraternity is conducting a raffle where 50 tickets are to be sold, one per customer. There are three prizes to be awarded. If the four organizers of the raffle each buy one ticket, what is the probability that the four organizers win

A. all of the prizes?

R Solution:

```
numTotalWin <- choose(50, 3)

numOrganizersWin <- choose(4, 3)
```

```r
otherWin <- choose(46, 0)

pAllWin <- (numOrganizersWin * otherWin) / numTotalWin

cat("The Probability that all of the prizes are won by the organizers is",
    pAllWin,
    "\n")
```

```
## The Probability that all of the prizes are won by the organizers is 0.0002040816
```

B. exactly two of the prizes?

```r
numOrganizersWin <- choose(4, 2)
otherWin <- choose(46, 1)

pAllWin <- (numOrganizersWin * otherWin) / numTotalWin

cat("The Probability that exactly two of the prizes are won by the organizers is",
    pAllWin,
    "\n")
```

```
## The Probability that exactly two of the prizes are won by the organizers is 0.01408163
```

C. exactly one of the prizes?

```r
numOrganizersWin <- choose(4, 1)
otherWin <- choose(46, 2)

pAllWin <- (numOrganizersWin * otherWin) / numTotalWin

cat("The Probability that exactly one of the prizes are won by the organizers is",
    pAllWin,
    "\n")
```

```
## The Probability that exactly one of the prizes are won by the organizers is 0.2112245
```

D. none of the prizes?

```r
numOrganizersWin <- choose(4, 0)
otherWin <- choose(46, 3)

pAllWin <- (numOrganizersWin * otherWin) / numTotalWin

cat("The Probability that exactly one of the prizes are won by the organizers is",
    pAllWin,
    "\n")
```

```
## The Probability that exactly one of the prizes are won by the organizers is 0.7744898
```