###########################################################################

#Data Visualization

###########################################################################

# bar graph

counts <- table(mtcars$gear)

barplot(counts,main = "Simple Bar Plot",

xlab = "Improvement",

ylab = "Frequency",

legend = rownames(counts),col = c("Red", "Yellow", "Green"))

A screenshot of a cell phone

Description automatically generated

# stacked bar graph

counts <- table(mtcars$vs,mtcars$gear)

barplot(counts,main = "Car Distribution By Gears and VS",

xlab = "Number Of Gears",

ylab = "Frequency",

legend = rownames(counts),col = c("grey", "cornflowerblue"))

A screenshot of a cell phone

Description automatically generated

# grouped bar chart

counts <- table(mtcars$vs,mtcars$gear)

barplot(counts,main = "Car Distribution By Gears and VS",

xlab = "Number Of Gears",

ylab = "Frequency",

legend = rownames(counts),col = c("grey", "cornflowerblue"),beside = T)

A screenshot of a computer

Description automatically generated

# pie chart

slices <- c(10,12,4,16,8)

pct <- round(slices/sum(slices) \* 100)

lbls <- paste(c("India","USA","Australia","France","Germany"), " ", pct, "%",sep = "")

pie(slices,labels = lbls,col = rainbow(5) , main = "Country Pie Chart With Percentages")

A close up of a logo

Description automatically generated

# 3-D pie chart

# install.packages("plotrix",dependencies = T)

library(plotrix)

pie3D(slices,labels = lbls,explode = 0.0,main = "3D Pie Chart View Of Countries")

A close up of a logo

Description automatically generated

# histogram

hist\_mpg <- mtcars$mpg # miles per gallon data

hist(hist\_mpg,breaks = 8,col = "darkgreen") # break controls the number of bins

A picture containing clock

Description automatically generated

# Kernel Density Plot

hist\_mpg <- mtcars$mpg

density\_data <- density(hist\_mpg)

plot(density\_data,main = "Kernel Density Of Miles Per Gallon")

polygon(density\_data,col = "green",border = "black")

A close up of a logo

Description automatically generated

# Line Chart

weight <- c(2.5,2.8,3.2,4.8,5.1,5.9,6.8,7.1,7.8,8.1)

months <- c(0,1,2,3,4,5,6,7,8,9)

plot(months,weight,type = "b",col = "red", main = "Baby Weight Chart")

A close up of a map

Description automatically generated

# Heat Map

install.packages("clf",dependencies = T)

Warning in install.packages :

package ‘clf’ is not available (for R version 3.6.3)

library(clf)

clf heatmap(spreads\_small,1:15,labels\_small)

A screenshot of a cell phone

Description automatically generated

# Word Cloud

install.packages("wordcloud",dependencies = T)

library(wordcloud)

# There are a few ways to customize it.

# scale: This is used to indicate the range of sizes of the words.

# max.words and min.freq: These parameters are used to limit the number of words plotted.

# max.words will plot the specified number of words and discard least frequent terms,

whereas, min.freq will discard all terms whose frequency is below the specified value.

# random.order: By setting this to FALSE, we make it so that the words with the highest frequency are plotted first.If we don’t set this, it will plot the words in a random order, and the highest frequency words may not necessarily appear in the center.

# rot.per: This value determines the fraction of words that are plotted vertically.

# colors: The default value is black. If you want to use different colors based on frequency, you can specify a vector of colors, or use one of the pre-defined color palettes.You can find a list [here](https://www.sthda.com/sthda/RDoc/images/rcolorbrewer.png).

data <- read.csv('Advertising.csv', header = T)

head(data)

wordcloud(words = data$TV,freq = data$Newspaper, min.freq = 2,max.words = 100,random.order = FALSE,rot.per = 0.35,colors = brewer.pal(8,"Spectral")) A picture containing food

Description automatically generated A picture containing food

Description automatically generated

# ggplot2

# It breaks up graphs into semantic components such as scales and layers.

# The frequency need not be calculated

library(ggplot2)

ggplot(hsb,aes(x=read)) + geom\_bar() # creating a bar plot with just one variable with bars

ggplot(hsb,aes(x=read)) + geom\_density() # creating a kernel density plan with one variable with a curve line

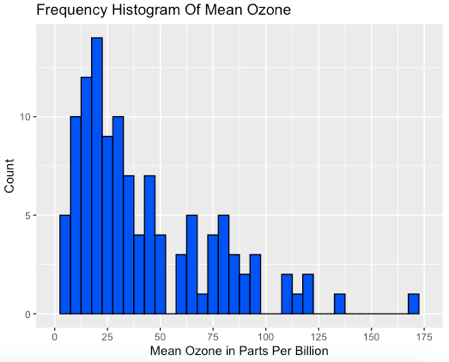
# Creating a Histogram using “airquality” dataset:

ggplot(airquality,aes(x = Ozone)) + geom\_histogram(aes(y = ..count..),

binwidth = 5,color = "black",fill = "blue") +

scale\_x\_continuous(name = "Mean Ozone in Parts Per Billion",breaks = seq(0, 175, 25), limits = c(0,175)) +

scale\_y\_continuous(name = "Count") + ggtitle("Frequency Histogram Of Mean Ozone")



# Creating a box plot using “airquality” dataset:

airquality$Month <- factor(airquality$Month,labels = c("May","June","July","Aug","Sept"))

ggplot(airquality,aes(x = Month,y = Ozone)) + geom\_boxplot(color = "black",fill = "blue") +

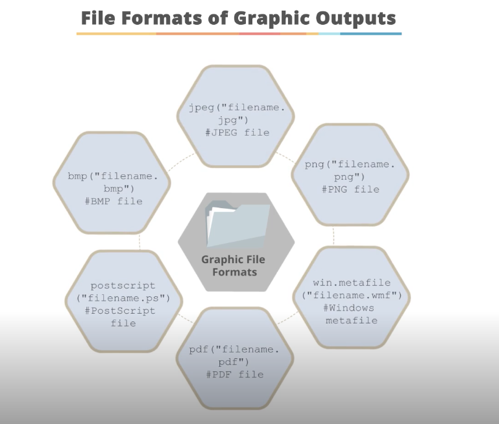
scale\_y\_continuous(name = "Mean Ozone in \n part per Billion",breaks = seq(0, 175, 25), limits = c(0,175)) +

scale\_x\_discrete(name = "Month") + ggtitle("Box Plot Of Mean Ozone By Month")

A close up of a logo

Description automatically generated

# Saving a Graphic Output as a File

# To save a graphic output as a file,following code can be used  A screenshot of a social media post

Description automatically generated

jpeg(“myplot.jpg”)

counts <- table(mtcars$gear)

barplot(counts)

dev.off() # The dev.off() function returns the control back to the terminal.