# ED308:

# Parametric Inference Lab File

## Quantitive Economics and Data Science

**NAME: ANAMIKA KUMARI**
**ROLL NO: IED/10022/21**

# 1 Sampling Distribution

## Code:

```
#Sampling Distribution
library(dplyr)
library(ggplot2)
ames <- read.csv("http://bit.ly/315N5R5")
glimpse(ames) # you need dplyr to use this function
area <- ames$Gr.Liv.Area
price <- ames$SalePrice
head(area, n=10) #show first 10 observations
head(price, n=10) #show first 10 observations
length(area) #how many observations in the vector?
any(is.na(area)) #is there any NA in the vector area?
area.pop.sd<-sqrt(sum((area - mean(area))^2)/(2930)) # Population standard deviation
area.pop.sd
summary(area)
hist(area,
    main = "Histogram of above ground living area",
    xlab = "Above ground living area (sq.ft.)",
)
area <- ames$Gr.Liv.Area # create new dataset containing only variable 'Gr.Liv.Area' from dataset 'ames'

samp1 <- sample(area, 50) #take a random sample of 50 observations from the dataset 'area'

mean(samp1) # mean of the sample distribution for area. Note difference from population mean.
area <- ames$Gr.Liv.Area

sample_means50 <- rep(NA, 5000) #initialise a vector

for(i in 1:5000){   # use of a loop function to draw a random sample 5000 times
  samp <- sample(area, 50)
  sample_means50[i] <- mean(samp)
}

hist(sample_means50, breaks = 25,
    main = "Sampling distribution of sample mean for Above ground living area",
    xlab = "Means (sq.ft.)") #Histogram of the 5000 samples (sampling distribution of the samples mean)
```

## Output:

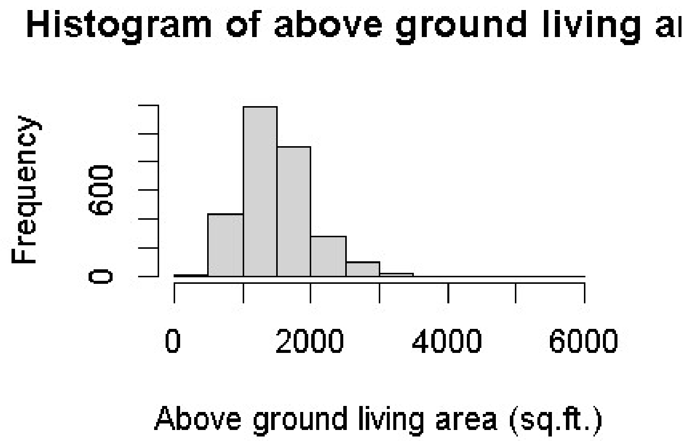
```
> library(dplyr)
> ames <- read.csv("http://bit.ly/315N5R5")
> glimpse(ames) # you need dplyr to use this function
Rows: 2,930
```

```
Columns: 82
$ Order <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 2…
$ PID <int> 526301100, 526350040, 526351010, 526353030, 527105
010, 527105030, 527127…
$ MS.SubClass <int> 20, 20, 20, 20, 60, 60, 120, 120, 120, 60, 60, 20,
60, 20, 120, 60, 50, …
$ MS.Zoning <chr> "RL", "RH", "RL", "RL", "RL", "RL", "RL", "RL", "R
L", "RL", "RL", "RL", …
$ Lot.Frontage <int> 141, 80, 81, 93, 74, 78, 41, 43, 39, 60, 75, NA, 6
3, 85, NA, 47, 152, 88…
$ Lot.Area <int> 31770, 11622, 14267, 11160, 13830, 9978, 4920, 500
5, 5389, 7500, 10000, …
$ Street <chr> "Pave", "Pave", "Pave", "Pave", "Pave", "Pave", "P
ave", "Pave", "Pave", …
$ Alley <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
, NA, NA, NA, NA, NA, …
$ Lot.Shape <chr> "IR1", "Reg", "IR1", "Reg", "IR1", "IR1", "Reg", "
IR1", "IR1", "Reg", "I…
$ Land.Contour <chr> "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "
HLS", "Lvl", "Lvl", "L…
$ Utilities <chr> "AllPub", "AllPub", "AllPub", "AllPub", "AllPub",
"AllPub", "AllPub", "A…
$ Lot.Config <chr> "Corner", "Inside", "Corner", "Corner", "Inside",
"Inside", "Inside", "I…
$ Land.Slope <chr> "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "
Gtl", "Gtl", "Gtl", "G…
$ Neighborhood <chr> "NAmes", "NAmes", "NAmes", "NAmes", "Gilbert", "Gi
lbert", "StoneBr", "St…
$ Condition.1 <chr> "Norm", "Feedr", "Norm", "Norm", "Norm", "Norm", "
Norm", "Norm", "Norm",…
$ Condition.2 <chr> "Norm", "Norm", "Norm", "Norm", "Norm", "Norm", "N
orm", "Norm", "Norm", …
$ Bldg.Type <chr> "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "T
wnhsE", "TwnhsE", "Twn…
$ House.Style <chr> "1Story", "1Story", "1Story", "1Story", "2Story",
"2Story", "1Story", "1…
$ Overall.Qual <int> 6, 5, 6, 7, 5, 6, 8, 8, 8, 7, 6, 6, 6, 7, 8, 8, 8,
9, 4, 6, 6, 7, 7, 6, …
$ Overall.Cond <int> 5, 6, 6, 5, 5, 6, 5, 5, 5, 5, 5, 7, 5, 5, 5, 5, 7,
2, 5, 6, 6, 6, 5, 7, …
$ Year.Built <int> 1960, 1961, 1958, 1968, 1997, 1998, 2001, 1992, 19
95, 1999, 1993, 1992, …
$ Year.Remod.Add <int> 1960, 1961, 1958, 1968, 1998, 1998, 2001, 1992,
19
96, 1999, 1994, 2007, …
$ Roof.Style <chr> "Hip", "Gable", "Hip", "Hip", "Gable", "Gable", "G
able", "Gable", "Gable…
$ Roof.Matl <chr> "CompShg", "CompShg", "CompShg", "CompShg", "CompS
hg", "CompShg", "CompS…
$ Exterior.1st <chr> "BrkFace", "VinylSd", "Wd Sdng", "BrkFace", "Vinyl
Sd", "VinylSd", "CemntB…
$ Exterior.2nd <chr> "Plywood", "VinylSd", "Wd Sdng", "BrkFace", "Vinyl
Sd", "VinylSd", "Cment…
$ Mas.Vnr.Type <chr> "Stone", "None", "BrkFace", "None", "None", "BrkFa
ce", "None", "None", "…
$ Mas.Vnr.Area <int> 112, 0, 108, 0, 0, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0,
603, 0, 350, 0, 119, 4…
$ Exter.Qual <chr> "TA", "TA", "TA", "Gd", "TA", "TA", "Gd", "Gd", "G
d", "TA", "TA", "TA", …
$ Exter.Cond <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", "T
A", "TA", "TA", "Gd", …
$ Foundation <chr> "CBlock", "CBlock", "CBlock", "CBlock", "PConc", "
PConc", "PConc", "PCon…
$ Bsmt.Qual <chr> "TA", "TA", "TA", "TA", "Gd", "TA", "Gd", "Gd", "G
d", "TA", "Gd", "Gd", …
$ Bsmt.Cond <chr> "Gd", "TA", "TA", "TA", "TA", "TA", "TA", "TA", "T
A", "TA", "TA", "TA", …
$ Bsmt.Exposure <chr> "Gd", "No", "No", "No", "No", "No", "Mn", "No", "N
o", "No", "No", "No", …
```

```
$ BsmtFin.Type.1 <chr> "BLQ", "Rec", "ALQ", "ALQ", "GLQ", "GLQ", "GLQ",
"

ALQ", "GLQ", "Unf", "U…
$ BsmtFin.SF.1 <int> 639, 468, 923, 1065, 791, 602, 616, 263, 1180, 0,
0, 935, 0, 637, 368, 1…
$ BsmtFin.Type.2 <chr> "Unf", "LwQ", "Unf", "Unf", "Unf", "Unf", "Unf",
"

Unf", "Unf", "Unf", "U…
$ BsmtFin.SF.2 <int> 0, 144, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1120,
0, 0, 0, 0, 163, 0, 16…
$ Bsmt.Unf.SF <int> 441, 270, 406, 1045, 137, 324, 722, 1017, 415, 994
, 763, 233, 789, 663, …
$ Total.Bsmt.SF <int> 1080, 882, 1329, 2110, 928, 926, 1338, 1280, 1595,
994, 763, 1168, 789, …
$ Heating <chr> "GasA", "GasA", "GasA", "GasA", "GasA", "GasA", "G
asA", "GasA", "GasA", …
$ Heating.QC <chr> "Fa", "TA", "TA", "Ex", "Gd", "Ex", "Ex", "Ex", "E
x", "Gd", "Gd", "Ex", …
$ Central.Air <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y",
"Y", "Y", "Y", "Y", "Y…
$ Electrical <chr> "SBrkr", "SBrkr", "SBrkr", "SBrkr", "SBrkr", "SBrk
r", "SBrkr", "SBrkr", …
$ X1st.Flr.SF <int> 1656, 896, 1329, 2110, 928, 926, 1338, 1280, 1616,
1028, 763, 1187, 789,…
$ X2nd.Flr.SF <int> 0, 0, 0, 0, 701, 678, 0, 0, 0, 776, 892, 0, 676, 0
, 0, 1589, 672, 0, 0, …
$ Low.Qual.Fin.SF <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
0, 0, 0, 0, 0, 0, 0, …
$ Gr.Liv.Area <int> 1656, 896, 1329, 2110, 1629, 1604, 1338, 1280, 161
6, 1804, 1655, 1187, 1…
$ Bsmt.Full.Bath <int> 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0,
1, 0, 1, 0, 1, 1, 1, …
$ Bsmt.Half.Bath <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, …
$ Full.Bath <int> 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 3, 2,
1, 1, 2, 2, 2, 2, 1, …
$ Half.Bath <int> 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0,
1, 0, 0, 0, 0, 1, 0, …
$ Bedroom.AbvGr <int> 3, 2, 3, 3, 3, 3, 2, 2, 2, 3, 3, 3, 3, 2, 1, 4, 4,
1, 2, 3, 3, 3, 3, 2, …
$ Kitchen.AbvGr <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, …
$ Kitchen.Qual <chr> "TA", "TA", "Gd", "Ex", "TA", "Gd", "Gd", "Gd", "G
d", "Gd", "TA", "TA", …
$ TotRms.AbvGrd <int> 7, 5, 6, 8, 6, 7, 6, 5, 5, 7, 7, 6, 7, 5, 4, 12, 8
, 8, 4, 7, 7, 6, 7, 5,…
$ Functional <chr> "Typ", "Typ", "Typ", "Typ", "Typ", "Typ", "Typ", "
Typ", "Typ", "Typ", "T…
$ Fireplaces <int> 2, 0, 0, 2, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0,
1, 0, 2, 1, 2, 0, 1, …
$ Fireplace.Qu <chr> "Gd", NA, NA, "TA", "TA", "Gd", NA, NA, "TA", "TA"
, "TA", NA, "Gd", "Po"…
$ Garage.Type <chr> "Attchd", "Attchd", "Attchd", "Attchd", "Attchd",
"Attchd", "Attchd", "A…
$ Garage.Yr.Blt <int> 1960, 1961, 1958, 1968, 1997, 1998, 2001, 1992, 19
95, 1999, 1993, 1992, …
$ Garage.Finish <chr> "Fin", "Unf", "Unf", "Fin", "Fin", "Fin", "Fin", "
RFn", "RFn", "Fin", "F…
$ Garage.Cars <int> 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2,
3, 2, 2, 2, 2, 2, 2, …
$ Garage.Area <int> 528, 730, 312, 522, 482, 470, 582, 506, 608, 442,
440, 420, 393, 506, 52…
$ Garage.Qual <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", "T
A", "TA", "TA", "TA", …
$ Garage.Cond <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", "T
A", "TA", "TA", "TA", …
$ Paved.Drive <chr> "P", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y",
"Y", "Y", "Y", "Y", "Y…
$ Wood.Deck.SF <int> 210, 140, 393, 0, 212, 360, 0, 0, 237, 140, 157, 4
83, 0, 192, 0, 503, 32…
```

```
$ Open.Porch.SF <int> 62, 0, 36, 0, 34, 36, 0, 82, 152, 60, 84, 21, 75,
0, 54, 36, 12, 0, 0, 0…
$ Enclosed.Porch <int> 0, 0, 0, 0, 0, 0, 170, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0…
$ X3Ssn.Porch <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, …
$ Screen.Porch <int> 0, 120, 0, 0, 0, 0, 0, 144, 0, 0, 0, 0, 0, 0, 140,
210, 0, 0, 0, 0, 0, 0…
$ Pool.Area <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, …
$ Pool.QC <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
, NA, NA, NA, NA, NA, …
$ Fence <chr> NA, "MnPrv", NA, NA, "MnPrv", NA, NA, NA, NA, NA,
NA, "GdPrv", NA, NA, N…
$ Misc.Feature <chr> NA, NA, "Gar2", NA, NA, NA, NA, NA, NA, NA, NA, "S
hed", NA, NA, NA, NA, …
$ Misc.Val <int> 0, 0, 12500, 0, 0, 0, 0, 0, 0, 0, 0, 500, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, …
$ Mo.Sold <int> 5, 6, 6, 4, 3, 6, 4, 1, 3, 6, 4, 3, 5, 2, 6, 6, 6,
6, 6, 2, 1, 1, 1, 3, …
$ Yr.Sold <int> 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 20
10, 2010, 2010, 2010, …
$ Sale.Type <chr> "WD ", "WD ", "WD ", "WD ", "WD ", "WD ", "WD ", "
WD ", "WD ", "WD ", "W…
$ Sale.Condition <chr> "Normal", "Normal", "Normal", "Normal", "Normal",
"Normal", "Normal", "N…
$ SalePrice <int> 215000, 105000, 172000, 244000, 189900, 195500, 21
3500, 191500, 236500, …
> area <- ames$Gr.Liv.Area
> price <- ames$SalePrice
> head(area, n=10) #show first 10 observations
[1] 1656 896 1329 2110 1629 1604 1338 1280 1616 1804
> head(price, n=10) #show first 10 observations
[1] 215000 105000 172000 244000 189900 195500 213500 191500 236500
189000
> length(area) #how many observations in the vector?
[1] 2930
> any(is.na(area)) #is there any NA in the vector area?
[1] FALSE
> area.pop.sd<-sqrt(sum((area - mean(area))^2)/(2930)) # Population
standa
rd deviation
> area.pop.sd
[1] 505.4226
> summary(area)
Min. 1st Qu. Median Mean 3rd Qu. Max.
334 1126 1442 1500 1743 5642
> hist(area,
+ main = "Histogram of above ground living area",
+ xlab = "Above ground living area (sq.ft.)",
+ )
```

## Plot:



```
> area <- ames$Gr.Liv.Area # create new dataset containing only variable
'
Gr.Liv.Area' from dataset 'ames'
> samp1 <- sample(area, 50) #take a random sample of 50 observations
from
the dataset 'area'
> mean(samp1) # mean of the sample distribution for area. Note
difference
from population mean.
[1] 1471.26
> samp2 <- sample(area, 1000)
> mean(samp2)
[1] 1506.203
> samp3 <- sample(area, 1500)
> mean(samp3)
[1] 1503.226
> samp4 <- sample(area, 2000)
> mean(samp4)
[1] 1503.557
```

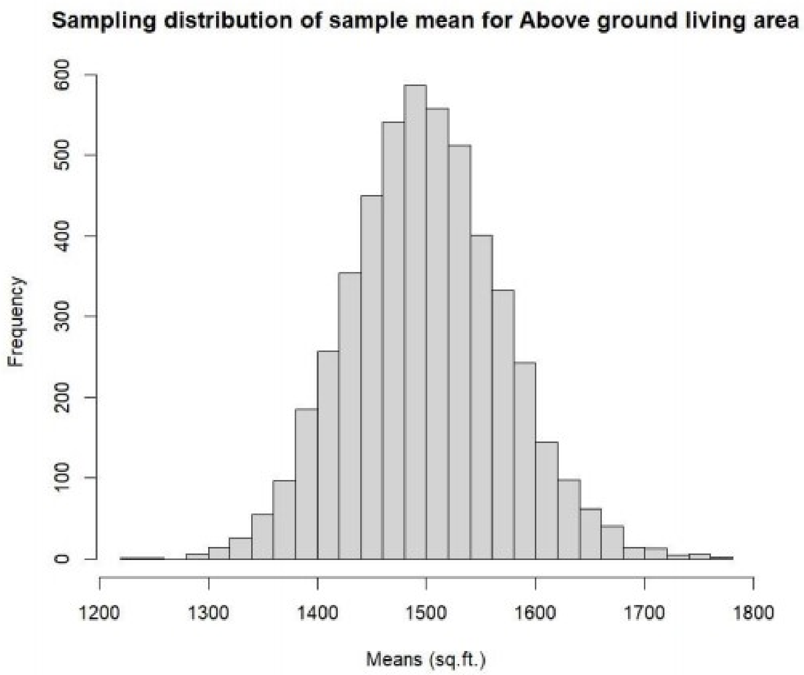
# 2 Sample Size and Sampling Distribution code:

```
#2 Sample Size and Sampling Distribution
area <- ames$Gr.Liv.Area
area
sample_means50 <- rep(NA, 5000)
#to compute sampling distribution
for(i in 1:5000){
  samp <- sample(area, 50)
  sample_means50[i] <- mean(samp)
}
#Estimating the avg living area in homes in ames
hist(sample_means50)
#To get a sense of the effect that sample size has
#on our distribution, let’s build up two more sampling distributions:
#one based on a sample size of 10 and another based on a sample size of 100
#from a population size of 5000.
area <- ames$Gr.Liv.Area
```

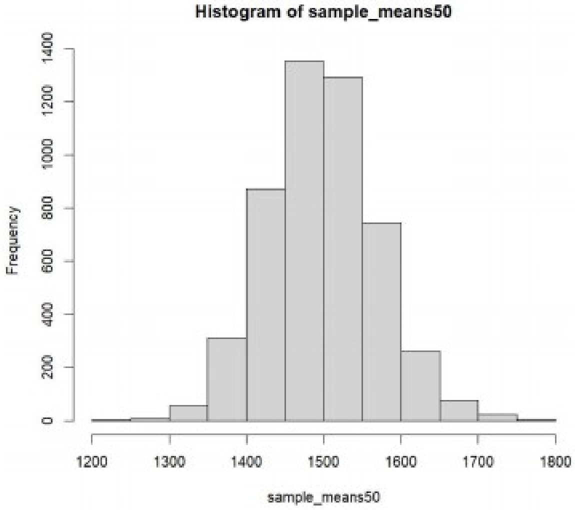
```
sample_means10 <- rep(NA, 5000)
sample_means10
sample_means100 <- rep(NA, 5000)
sample_means100
for(i in 1:5000){
  samp <- sample(area, 10)
  sample_means10[i] <- mean(samp)
  samp <- sample(area, 100)
  sample_means100[i] <- mean(samp)
}

#To see the effect that different sample sizes
#have on the sampling distribution,
#let's plot the three distributions on top of one another.
area <- ames$Gr.Liv.Area

sample_means10 <- rep(NA, 5000)
sample_means10
sample_means50 <- rep(NA, 5000)
sample_means50
sample_means100 <- rep(NA, 5000)
sample_means100
for(i in 1:5000){
  samp <- sample(area, 10)
  sample_means10[i] <- mean(samp)
  samp <- sample(area, 50)
  sample_means50[i] <- mean(samp)
  samp <- sample(area, 100)
  sample_means100[i] <- mean(samp)
}

par(mfrow = c(3, 1))  # this creates 3 rows and 1 column for graphs

xlimits <- range(sample_means10)
xlimits
hist(sample_means10,  breaks = 25, xlim = xlimits)

hist(sample_means50,  breaks = 25, xlim = xlimits)

hist(sample_means100, breaks = 25, xlim = xlimits)
```

# Output:

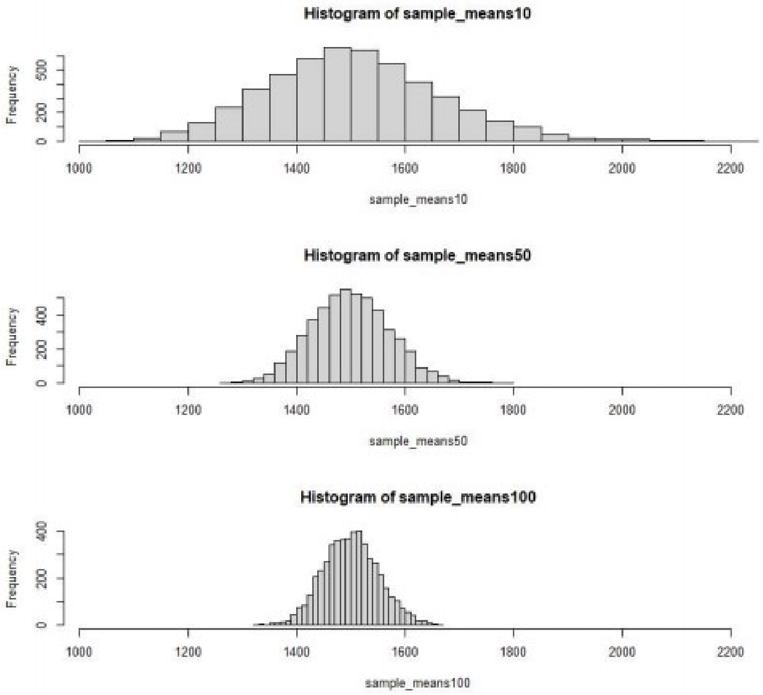
```
> #Sampling Distribution of Means
> area <- ames$Gr.Liv.Area
> sample_means50 <- rep(NA, 5000) #initialise a vector
> for(i in 1:5000){ # use of a loop function to draw a random sample 500
0 times
+ samp <- sample(area, 50)
+ sample_means50[i] <- mean(samp)
+ }
> hist(sample_means50, breaks = 25,
+ main = "Sampling distribution of sample mean for Above ground livin
g area",
+ xlab = "Means (sq.ft.)") #Histogram of the 5000 samples (sampling d
istribution of the samples mean)
```



```
> area <- ames$Gr.Liv.Area
> sample_means50 <- rep(NA, 5000)
> for(i in 1:5000){
+ samp <- sample(area, 50)
+ sample_means50[i] <- mean(samp)
+ }
> hist(sample_means50)
> sample_means10 <- rep(NA, 5000)
> sample_means100 <- rep(NA, 5000)
> for(i in 1:5000){
+ samp <- sample(area, 10)
+ sample_means10[i] <- mean(samp)
+ samp <- sample(area, 100)
+ sample_means100[i] <- mean(samp)
+ }
```



```
> sample_means10 <- rep(NA, 5000)
> sample_means50 <- rep(NA, 5000)
> sample_means100 <- rep(NA, 5000)
> for(i in 1:5000){
+ samp <- sample(area, 10)
+ sample_means10[i] <- mean(samp)
+ samp <- sample(area, 50)
+ sample_means50[i] <- mean(samp)
+ samp <- sample(area, 100)
+ sample_means100[i] <- mean(samp)
+ }
> par(mfrow = c(3, 1)) # this creates 3 rows and 1 column for graphs
> xlimits <- range(sample_means10)
> hist(sample_means10, breaks = 25, xlim = xlimits)
> hist(sample_means50, breaks = 25, xlim = xlimits)
> hist(sample_means100, breaks = 25, xlim = xlimits)
>
```



# ED308: Parametric Inference Lab 2

## Part One: Correlation:

## code:

Define x and y as vector of same length and then:

```
my_data <- mtcars

x = my_data$mpg

y = my_data$cyl

cor(x, y, method = c("pearson", "kendall", "spearman"))

cor.test(x, y, method=c("pearson", "kendall", "spearman"))



cor(x, y,  method = "pearson", use = "complete.obs")



my_data <- mtcars

head(my_data, 6)

summary(my_data)

install.packages("ggpubr")

library("ggpubr")

ggscatter(my_data, x = "mpg", y = "wt", add = "reg.line", conf.int = TRUE, cor.coef = TRUE, cor.method = "pearson",  xlab = "Miles/(US) gallon", ylab = "Weight (1000 lbs)")
```

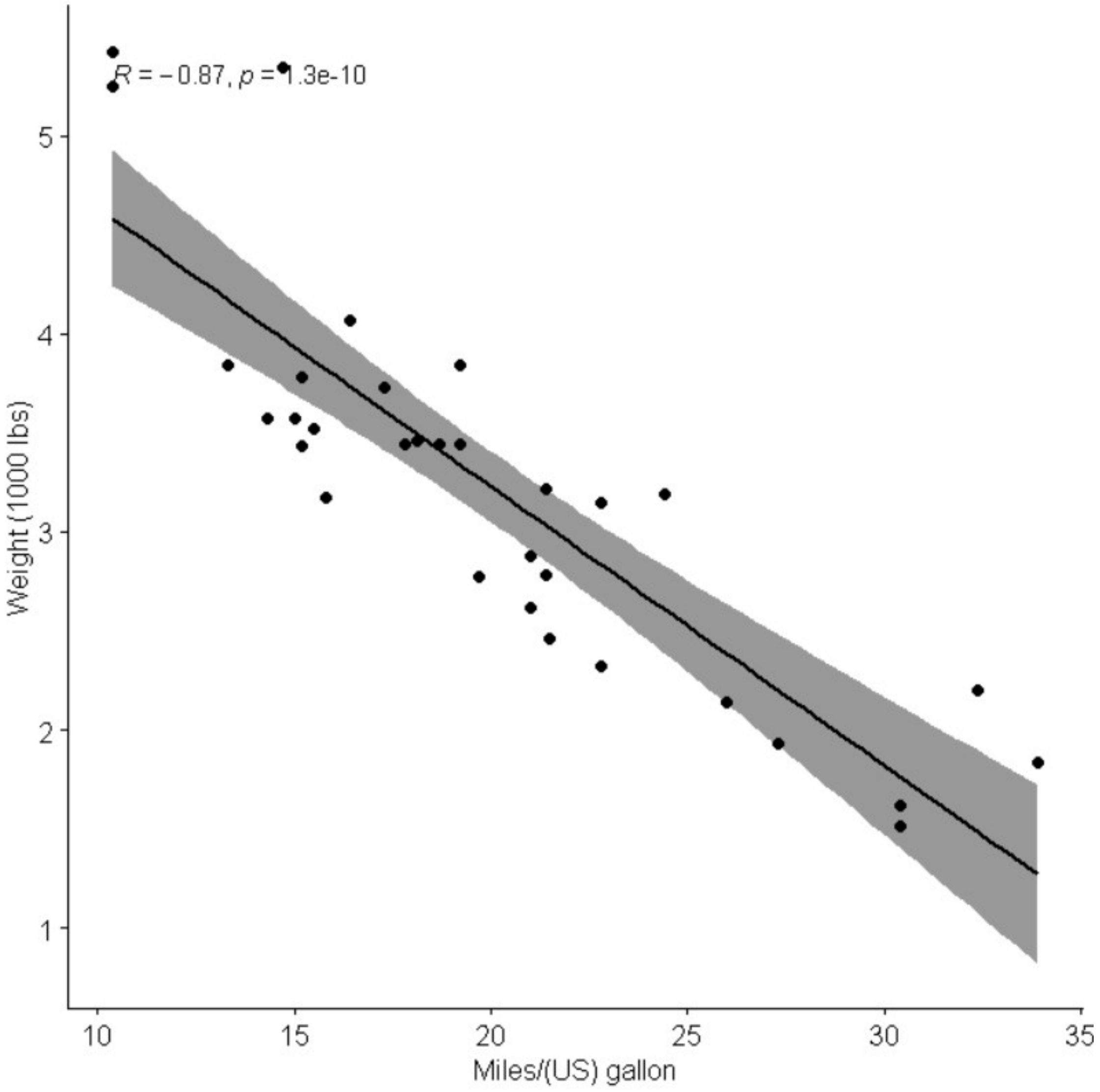
## output:

```
> my_data = mtcars
> x = my_data$mpg
> y = my_data$cyl
> cor(x, y, method = c("pearson", "kendall", "spearman"))
[1] -0.852162
> cor.test(x, y, method=c("pearson", "kendall", "spearman"))

	Pearson's product-moment correlation

data:  x and y
t = -8.9197, df = 30, p-value = 6.113e-10
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.9257694 -0.7163171
sample estimates:
      cor 
-0.852162 

> cor(x, y,  method = "pearson", use = "complete.obs")
[1] -0.852162
> my_data <- mtcars
> head(my_data, 6)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
```

```
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
> summary(my_data)
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
> library(ggpubr)
Error in library(ggpubr) : there is no package called ‘ggpubr’
> install.packages("ggpubr")
Installing package into ‘C:/Users/hp/AppData/Local/R/win-library/4.3’
(as ‘lib’ is unspecified)
--- Please select a CRAN mirror for use in this session ---
also installing the dependencies ‘stringr’, ‘ggrepel’, ‘ggsci’, ‘tidyr’, ‘purrr’, ‘cowplot’, ‘ggsignif’, ‘polynom’, ‘rstatix’


  There is a binary version available but the source version is later:
        binary source needs_compilation
stringr  1.5.0  1.5.1             FALSE

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/ggrepel_0.9.4.zip'
Content type 'application/zip' length 607494 bytes (593 KB)
downloaded 593 KB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/ggsci_3.0.0.zip'
Content type 'application/zip' length 2434431 bytes (2.3 MB)
downloaded 2.3 MB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/tidyr_1.3.0.zip'
Content type 'application/zip' length 1281140 bytes (1.2 MB)
downloaded 1.2 MB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/purrr_1.0.2.zip'
Content type 'application/zip' length 499378 bytes (487 KB)
downloaded 487 KB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/cowplot_1.1.1.zip'
Content type 'application/zip' length 1375189 bytes (1.3 MB)
downloaded 1.3 MB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/ggsignif_0.6.4.zip'
Content type 'application/zip' length 601530 bytes (587 KB)
downloaded 587 KB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/polynom_1.4-1.zip'
Content type 'application/zip' length 404763 bytes (395 KB)
downloaded 395 KB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/rstatix_0.7.2.zip'
Content type 'application/zip' length 607670 bytes (593 KB)
downloaded 593 KB
```

```
trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/ggpubr_0.6.0.zip'
Content type 'application/zip' length 2087746 bytes (2.0 MB)
downloaded 2.0 MB

package ‘ggrepel’ successfully unpacked and MD5 sums checked
package ‘ggsci’ successfully unpacked and MD5 sums checked
package ‘tidyr’ successfully unpacked and MD5 sums checked
package ‘purrr’ successfully unpacked and MD5 sums checked
package ‘cowplot’ successfully unpacked and MD5 sums checked
package ‘ggsignif’ successfully unpacked and MD5 sums checked
package ‘polynom’ successfully unpacked and MD5 sums checked
package ‘rstatix’ successfully unpacked and MD5 sums checked
package ‘ggpubr’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\hp\AppData\Local\Temp\RtmpWi7ezD\downloaded_packages
installing the source package ‘stringr’

trying URL 'https://www.stats.bris.ac.uk/R/src/contrib/stringr_1.5.1.tar.gz'
Content type 'application/x-gzip' length 176599 bytes (172 KB)
downloaded 172 KB

* installing *source* package 'stringr' ...
** package 'stringr' successfully unpacked and MD5 sums checked
** using staged installation
** R
** data
*** moving datasets to lazyload DB
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
*** copying figures
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (stringr)

The downloaded source packages are in
        ‘C:\Users\hp\AppData\Local\Temp\RtmpWi7ezD\downloaded_packages’
> library(ggpubr)
> ggscatter(my_data, x = "mpg", y = "wt", add = "reg.line", conf.int = TRUE, cor.coef = TRUE, cor.method = "pearson",  xlab =
"Miles/(US) gallon", ylab = "Weight (1000 lbs)")
```

>



# Shapiro-Wilk normality test for mpg and for wt

**Code:**

```
# Shapiro-Wilk normality test for mpg
shapiro.test(my_data$mpg)

# Shapiro-Wilk normality test for wt
shapiro.test(my_data$wt)

result <- cor.test(my_data$wt, my_data$mpg, method = "pearson")
result

res2 <- cor.test(my_data$wt, my_data$mpg,  method="kendall")
res2

res3 <-cor.test(my_data$wt, my_data$mpg,  method = "spearman")
res3

# Load data
data("mtcars")
my_data <- mtcars[, c(1,3,4,5,6,7)]
# print the first 6 rows
head(my_data, 6)
```

```
res <- cor(my_data)
round(res, 2)
cor(my_data, use = "complete.obs")
# Install Hmisc package:
install.packages("Hmisc")
library("Hmisc")
res2 <- rcorr(as.matrix(my_data))
res2
```

output:

```
> # Shapiro-Wilk normality test for mpg
> shapiro.test(my_data$mpg)

	Shapiro-Wilk normality test

data:  my_data$mpg
W = 0.94756, p-value = 0.1229

>
> # Shapiro-Wilk normality test for wt
> shapiro.test(my_data$wt)

	Shapiro-Wilk normality test

data:  my_data$wt
W = 0.94326, p-value = 0.09265

> result <- cor.test(my_data$wt, my_data$mpg, method = "pearson")
> result

	Pearson's product-moment correlation

data:  my_data$wt and my_data$mpg
t = -9.559, df = 30, p-value = 1.294e-10
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.9338264 -0.7440872
sample estimates:
       cor 
-0.8676594 

>
> res2 <- cor.test(my_data$wt, my_data$mpg,  method="kendall")
Warning message:
In cor.test.default(my_data$wt, my_data$mpg, method = "kendall") :
  Cannot compute exact p-value with ties
> res2

	Kendall's rank correlation tau

data:  my_data$wt and my_data$mpg
z = -5.7981, p-value = 6.706e-09
alternative hypothesis: true tau is not equal to 0
sample estimates:
       tau 
-0.7278321 

> res3 <-cor.test(my_data$wt, my_data$mpg,  method = "spearman")
Warning message:
In cor.test.default(my_data$wt, my_data$mpg, method = "spearman") :
  Cannot compute exact p-value with ties
> res3
```

```
	Spearman's rank correlation rho

data:  my_data$wt and my_data$mpg
S = 10292, p-value = 1.488e-11
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho 
-0.886422 

> # Load data
> data("mtcars")
> my_data <- mtcars[, c(1,3,4,5,6,7)]
> # print the first 6 rows
> head(my_data, 6)
                   mpg disp  hp drat    wt  qsec
Mazda RX4         21.0  160 110 3.90 2.620 16.46
Mazda RX4 Wag     21.0  160 110 3.90 2.875 17.02
Datsun 710        22.8  108  93 3.85 2.320 18.61
Hornet 4 Drive    21.4  258 110 3.08 3.215 19.44
Hornet Sportabout 18.7  360 175 3.15 3.440 17.02
Valiant           18.1  225 105 2.76 3.460 20.22
> res <- cor(my_data)
> round(res, 2)
       mpg  disp    hp  drat    wt  qsec
mpg   1.00 -0.85 -0.78  0.68 -0.87  0.42
disp -0.85  1.00  0.79 -0.71  0.89 -0.43
hp   -0.78  0.79  1.00 -0.45  0.66 -0.71
drat  0.68 -0.71 -0.45  1.00 -0.71  0.09
wt   -0.87  0.89  0.66 -0.71  1.00 -0.17
qsec  0.42 -0.43 -0.71  0.09 -0.17  1.00
> cor(my_data, use = "complete.obs")
            mpg       disp         hp        drat         wt        qsec
mpg   1.0000000 -0.8475514 -0.7761684  0.68117191 -0.8676594  0.41868403
disp -0.8475514  1.0000000  0.7909486 -0.71021393  0.8879799 -0.43369788
hp   -0.7761684  0.7909486  1.0000000 -0.44875912  0.6587479 -0.70822339
drat  0.6811719 -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120476
wt   -0.8676594  0.8879799  0.6587479 -0.71244065  1.0000000 -0.17471588
qsec  0.4186840 -0.4336979 -0.7082234  0.09120476 -0.1747159  1.00000000
> install.packages("Hmisc")
Installing package into ‘C:/Users/hp/AppData/Local/R/win-library/4.3’
(as ‘lib’ is unspecified)
also installing the dependencies ‘checkmate’, ‘htmlwidgets’, ‘rstudioapi’, ‘htmlTable’, ‘viridis’

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/checkmate_2.3.0.zip'
Content type 'application/zip' length 751570 bytes (733 KB)
downloaded 733 KB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/htmlwidgets_1.6.2.zip'
Content type 'application/zip' length 811027 bytes (792 KB)
downloaded 792 KB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/rstudioapi_0.15.0.zip'
Content type 'application/zip' length 319298 bytes (311 KB)
downloaded 311 KB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/htmlTable_2.4.2.zip'
Content type 'application/zip' length 422708 bytes (412 KB)
downloaded 412 KB

trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/viridis_0.6.4.zip'
Content type 'application/zip' length 3029380 bytes (2.9 MB)
downloaded 2.9 MB
```

```
trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/Hmisc_5.1-1.zip'
Content type 'application/zip' length 3539838 bytes (3.4 MB)
downloaded 3.4 MB

package ‘checkmate’ successfully unpacked and MD5 sums checked
package ‘htmlwidgets’ successfully unpacked and MD5 sums checked
package ‘rstudioapi’ successfully unpacked and MD5 sums checked
package ‘htmlTable’ successfully unpacked and MD5 sums checked
package ‘viridis’ successfully unpacked and MD5 sums checked
package ‘Hmisc’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
	C:\Users\hp\AppData\Local\Temp\RtmpWi7ezD\downloaded_packages
> library("Hmisc")

Attaching package: ‘Hmisc’

The following objects are masked from ‘package:dplyr’:

    src, summarize

The following objects are masked from ‘package:base’:

    format.pval, units

> res2 <- rcorr(as.matrix(my_data))
> res2
       mpg  disp    hp  drat    wt  qsec
mpg   1.00 -0.85 -0.78  0.68 -0.87  0.42
disp -0.85  1.00  0.79 -0.71  0.89 -0.43
hp   -0.78  0.79  1.00 -0.45  0.66 -0.71
drat  0.68 -0.71 -0.45  1.00 -0.71  0.09
wt   -0.87  0.89  0.66 -0.71  1.00 -0.17
qsec  0.42 -0.43 -0.71  0.09 -0.17  1.00

n= 32


P
     mpg    disp   hp     drat   wt     qsec  
mpg         0.0000 0.0000 0.0000 0.0000 0.0171
disp 0.0000        0.0000 0.0000 0.0000 0.0131
hp   0.0000 0.0000        0.0100 0.0000 0.0000
drat 0.0000 0.0000 0.0100        0.0000 0.6196
wt   0.0000 0.0000 0.0000 0.0000        0.3389
qsec 0.0171 0.0131 0.0000 0.6196 0.3389       
>
```

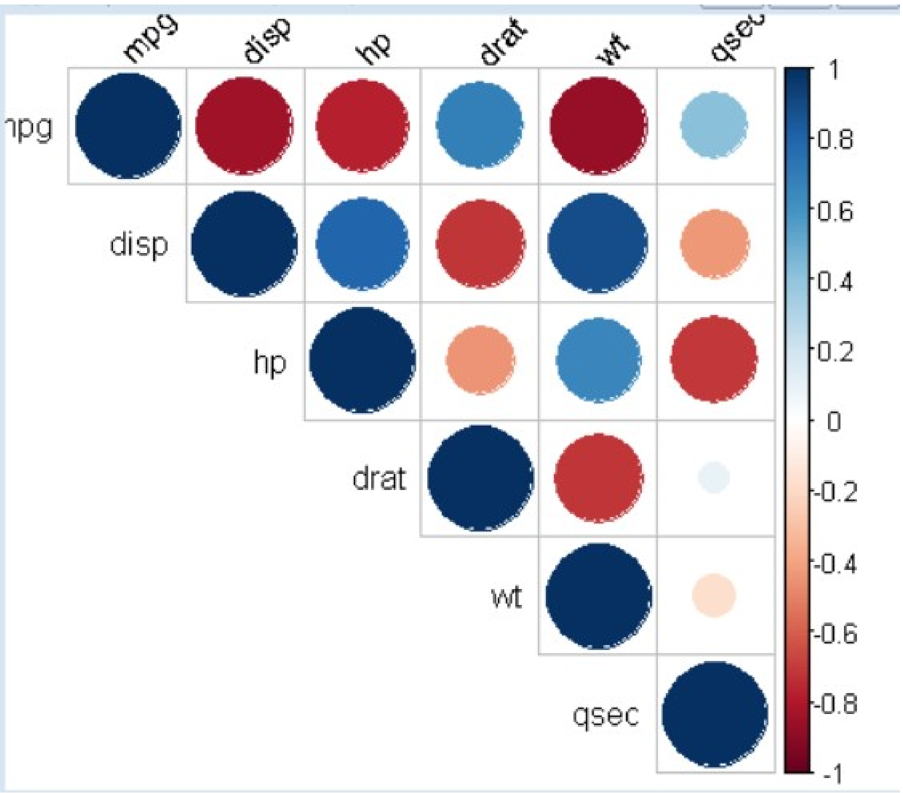
Code:

```
install.packages("corrplot")
library(corrplot)
corrplot(res, type = "upper", tl.col = "black", tl.srt = 45)
```

output:

```
> install.packages("corrplot")
Installing package into ‘C:/Users/hp/AppData/Local/R/win-library/4.3’
(as ‘lib’ is unspecified)
trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/corrplot_0.92.zip'
Content type 'application/zip' length 3844891 bytes (3.7 MB)
downloaded 3.7 MB

package ‘corrplot’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
```

```
        C:\Users\hp\AppData\Local\Temp\RtmpWi7ezD\downloaded_packages
> library(corrplot)
corrplot 0.92 loaded
> corrplot(res, type = "upper", tl.col = "black", tl.srt = 45)
>
```



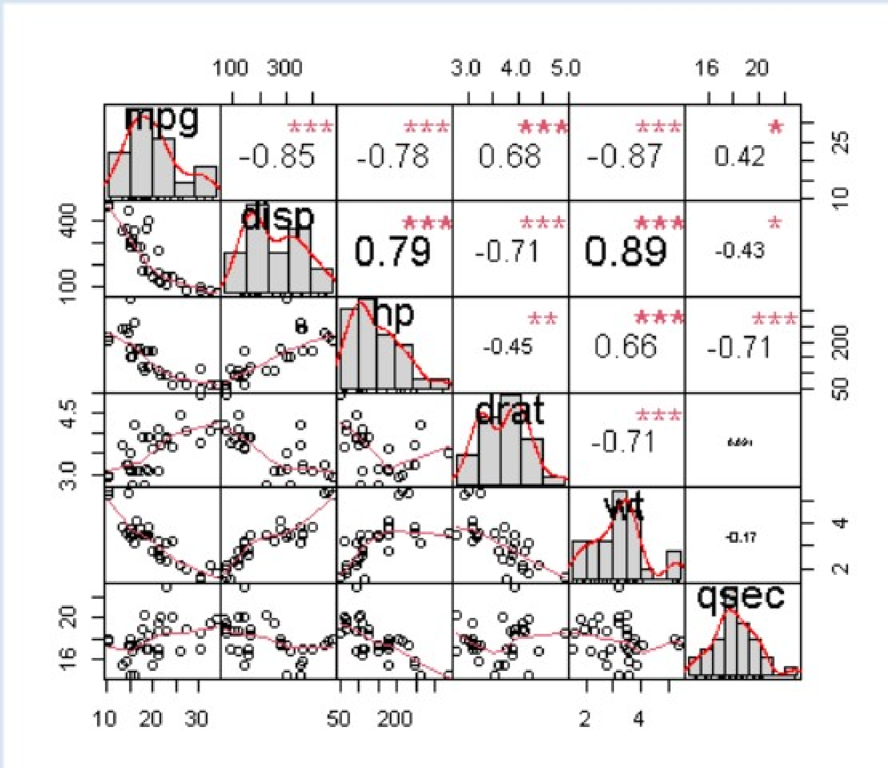
code:

```
install.packages("PerformanceAnalytics")
library("PerformanceAnalytics")
my_data <- mtcars[, c(1,3,4,5,6,7)]
chart.Correlation(my_data, histogram=TRUE, pch=19)
```

output:

```
> install.packages("PerformanceAnalytics")
Installing package into ‘C:/Users/hp/AppData/Local/R/win-library/4.3’
(as ‘lib’ is unspecified)
trying URL 'https://www.stats.bris.ac.uk/R/bin/windows/contrib/4.3/PerformanceAnalytics_2.0.4.zip'
Content type 'application/zip' length 3115183 bytes (3.0 MB)
downloaded 3.0 MB

package ‘PerformanceAnalytics’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\hp\AppData\Local\Temp\RtmpWi7ezD\downloaded_packages
> library("PerformanceAnalytics")
Loading required package: xts
Loading required package: zoo

Attaching package: ‘zoo’
```

```
The following objects are masked from ‘package:base’:

    as.Date, as.Date.numeric


######################### Warning from 'xts' package ##########################
#                                                                             #
# The dplyr lag() function breaks how base R's lag() function is supposed to  #
# work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
# source() into this session won't work correctly.                            #
#                                                                             #
# Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
# conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
# dplyr from breaking base R's lag() function.                                #
#                                                                             #
# Code in packages is not affected. It's protected by R's namespace mechanism #
# Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
#                                                                             #
###############################################################################

Attaching package: ‘xts’

The following objects are masked from ‘package:dplyr’:

    first, last


Attaching package: ‘PerformanceAnalytics’

The following object is masked from ‘package:graphics’:

    legend

> my_data <- mtcars[, c(1,3,4,5,6,7)]
> chart.Correlation(my_data, histogram=TRUE, pch=19)
There were 15 warnings (use warnings() to see them)
>
```



## Part Two: Comparing Means

code:

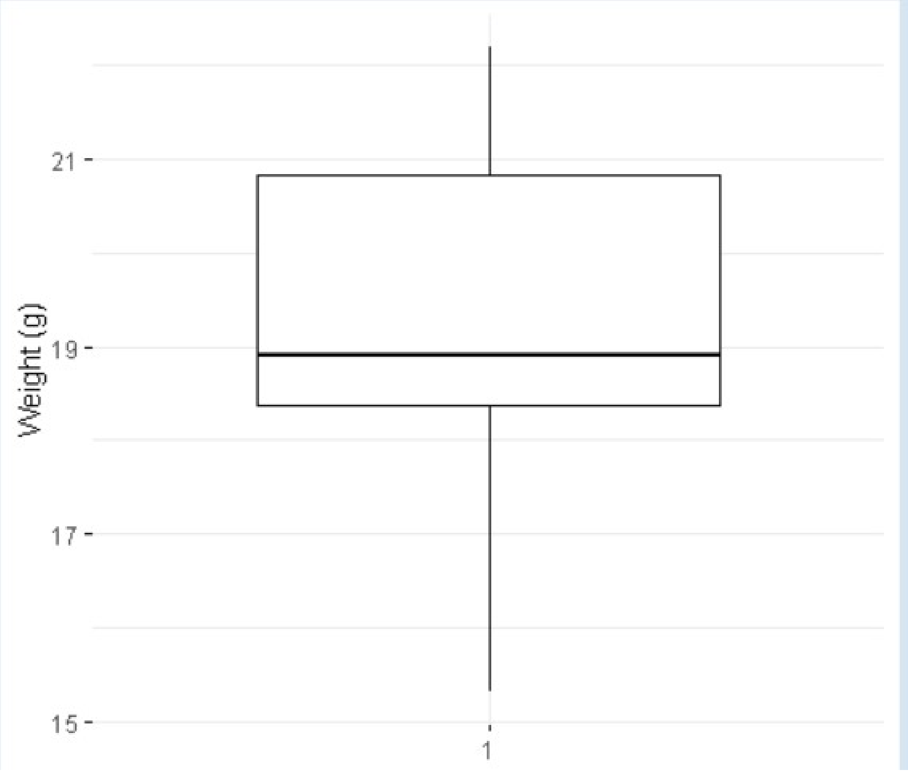
```
#Part Two: Comparing Means

t.test(x, mu = 0, alternative = "two.sided")

set.seed(1234)
my_data <- data.frame(
  name = paste0(rep("M_", 10), 1:10),
  weight = round(rnorm(10, 20, 2), 1))

# Print the data
my_data

# Statistical summaries of weight
summary(my_data$weight)

library(ggpubr)
ggboxplot(my_data$weight,
          ylab = "Weight (g)", xlab = FALSE,
          ggtheme = theme_minimal())
```

output:

```
> my_data <- mtcars[, c(1,3,4,5,6,7)]
> chart.Correlation(my_data, histogram=TRUE, pch=19)
There were 15 warnings (use warnings() to see them)
> t.test(x, mu = 0, alternative = "two.sided")

	One Sample t-test
```

```
data:  x
t = 18.857, df = 31, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 17.91768 22.26357
sample estimates:
mean of x
 20.09062

>
> set.seed(1234)
> my_data <- data.frame(
+   name = paste0(rep("M_", 10), 1:10),
+   weight = round(rnorm(10, 20, 2), 1))
>
> # Print the data
> my_data
   name weight
1   M_1   17.6
2   M_2   20.6
3   M_3   22.2
4   M_4   15.3
5   M_5   20.9
6   M_6   21.0
7   M_7   18.9
8   M_8   18.9
9   M_9   18.9
10 M_10   18.2
> # Statistical summaries of weight
> summary(my_data$weight)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  15.30   18.38   18.90   19.25   20.82   22.20 
> library(ggpubr)
> ggboxplot(my_data$weight,
+           ylab = "Weight (g)", xlab = FALSE,
+           ggtheme = theme_minimal())
>
```

Plot:



code:

```
shapiro.test(my_data$weight)

# One-sample t-test
mean(my_data$weight)

# two sided test (Ho: the mean is equal to 25g)
res <- t.test(my_data$weight, mu = 25)
# Printing the results
res

# one-sided test (Ho: the mean is greater than or equal to to 25g)
t.test(my_data$weight, mu = 25,
          alternative = "less")

wilcox.test(x, mu = 0, alternative = "two.sided")

# One-sample wilcoxon test
res <- wilcox.test(my_data$weight, mu = 25)
# Printing the results
res
```

## Part Three: Comparing Means of two Independent Groups

```
# R function to compute unpaired two-samples t-test
t.test(x, y, alternative = "two.sided", var.equal = FALSE)

# Data in two numeric vectors
women_weight <- c(38.9, 61.2, 73.3, 21.8, 63.4, 64.6, 48.4, 48.8, 48.5)
men_weight <- c(67.8, 60, 63.4, 76, 89.4, 73.3, 67.3, 61.3, 62.4)
```

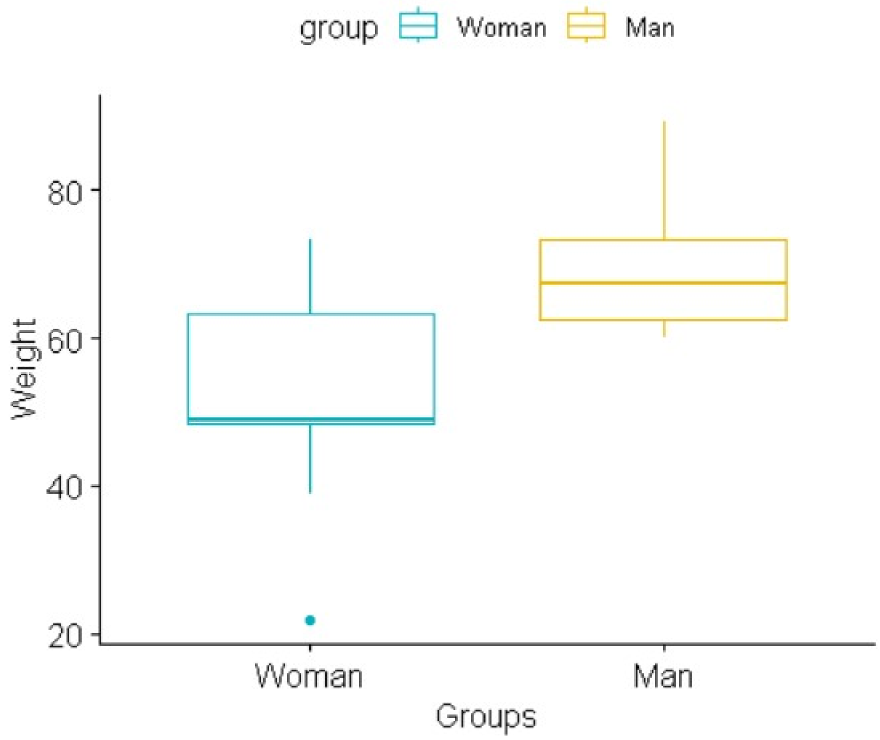
```
# Create a data frame
my_data <- data.frame(
                group = rep(c("Woman", "Man"), each = 9),
                weight = c(women_weight,  men_weight)
                )
print(my_data)

# Computing summary statistics by groups

library(dplyr)
group_by(my_data, group) %>%
  summarise(
    count = n(),
    mean = mean(weight, na.rm = TRUE),
    sd = sd(weight, na.rm = TRUE)
  )

# Plot weight by group and color by group
library("ggpubr")
ggboxplot(my_data, x = "group", y = "weight",
          color = "group", palette = c("#00AFBB", "#E7B800"),
        ylab = "Weight", xlab = "Groups")
```

**Output:**

```
> shapiro.test(my_data$weight)

      Shapiro-Wilk normality test

data:  my_data$weight
W = 0.9526, p-value = 0.6993

> # One-sample t-test
> mean(my_data$weight)
[1] 19.25
> # two sided test (Ho: the mean is equal to 25g)
> res <- t.test(my_data$weight, mu = 25)
> # Printing the results
> res

      One Sample t-test

data:  my_data$weight
t = -9.0783, df = 9, p-value = 7.953e-06
alternative hypothesis: true mean is not equal to 25
95 percent confidence interval:
 17.8172 20.6828
sample estimates:
mean of x
   19.25

> # one-sided test (Ho: the mean is greater than or equal to to 25g)
> t.test(my_data$weight, mu = 25,
+             alternative = "less")

      One Sample t-test

data:  my_data$weight
t = -9.0783, df = 9, p-value = 3.977e-06
alternative hypothesis: true mean is less than 25
95 percent confidence interval:
    -Inf 20.41105
sample estimates:
mean of x
   19.25
```

```
>
> wilcox.test(x, mu = 0, alternative = "two.sided")

        Wilcoxon signed rank test with continuity correction

data:  x
V = 528, p-value = 8.311e-07
alternative hypothesis: true location is not equal to 0

Warning message:
In wilcox.test.default(x, mu = 0, alternative = "two.sided") :
  cannot compute exact p-value with ties
> # One-sample wilcoxon test
> res <- wilcox.test(my_data$weight, mu = 25)
Warning message:
In wilcox.test.default(my_data$weight, mu = 25) :
  cannot compute exact p-value with ties
> # Printing the results
> res

        Wilcoxon signed rank test with continuity correction

data:  my_data$weight
V = 0, p-value = 0.005793
alternative hypothesis: true location is not equal to 25

>
> Part Three: Comparing Means of two Independent Groups
Error: unexpected symbol in "Part Three"
> t.test(x, y, alternative = "two.sided", var.equal = FALSE)

        Welch Two Sample t-test

data:  x and y
t = 12.512, df = 36.402, p-value = 9.508e-15
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 11.65034 16.15591
sample estimates:
mean of x mean of y
 20.09062   6.18750

> # Data in two numeric vectors
> women_weight <- c(38.9, 61.2, 73.3, 21.8, 63.4, 64.6, 48.4, 48.8, 48.5)
> men_weight <- c(67.8, 60, 63.4, 76, 89.4, 73.3, 67.3, 61.3, 62.4)
> # Create a data frame
> # Create a data frame
> my_data <- data.frame(
+                 group = rep(c("Woman", "Man"), each = 9),
+                 weight = c(women_weight,  men_weight)
+                 )
> print(my_data)
   group weight
1  Woman   38.9
2  Woman   61.2
3  Woman   73.3
4  Woman   21.8
5  Woman   63.4
6  Woman   64.6
7  Woman   48.4
8  Woman   48.8
9  Woman   48.5
10   Man   67.8
11   Man   60.0
12   Man   63.4
13   Man   76.0
14   Man   89.4
15   Man   73.3
16   Man   67.3
17   Man   61.3
18   Man   62.4
```

```
> # Computing summary statistics by groups
> library(dplyr)
> group_by(my_data, group) %>%
+   summarise(
+     count = n(),
+     mean = mean(weight, na.rm = TRUE),
+     sd = sd(weight, na.rm = TRUE)
+   )
# A tibble: 2 × 4
  group count  mean    sd
  <chr> <int> <dbl> <dbl>
1 Man       9  69.0  9.38
2 Woman     9  52.1 15.6
>
> # Plot weight by group and color by group
> library("ggpubr")
> ggboxplot(my_data, x = "group", y = "weight",
+          color = "group", palette = c("#00AFBB", "#E7B800"),
+         ylab = "Weight", xlab = "Groups")
>
Plot:
```



## Shapiro-Wilk normality test for Men's weights and for women’s weight

```
code:
# Shapiro-Wilk normality test for Men's weights
with(my_data, shapiro.test(weight[group == "Man"]))

# Shapiro-Wilk normality test for Women's weights
with(my_data, shapiro.test(weight[group == "Woman"]))

res.ftest <- var.test(weight ~ group, data = my_data)
res.ftest

t.test(women_weight, men_weight, var.equal = TRUE, alternative="less")

t.test(weight ~ group, data = my_data, var.equal = TRUE, alternative = "greater")
```

```
wilcox.test(x, y, alternative = "two.sided")

library(dplyr)
group_by(my_data, group) %>%
  summarise(
    count = n(),
    median = median(weight, na.rm = TRUE),
    IQR = IQR(weight, na.rm = TRUE)
  )

res <- wilcox.test(women_weight, men_weight)
Res
```

output:

```
> with(my_data, shapiro.test(weight[group == "Man"]))

        Shapiro-Wilk normality test

data:  weight[group == "Man"]
W = 0.86425, p-value = 0.1066

> # Shapiro-Wilk normality test for Women's weights
> with(my_data, shapiro.test(weight[group == "Woman"]))

        Shapiro-Wilk normality test

data:  weight[group == "Woman"]
W = 0.94266, p-value = 0.6101

>
> res.ftest <- var.test(weight ~ group, data = my_data)
> res.ftest

        F test to compare two variances

data:  weight by group
F = 0.36134, num df = 8, denom df = 8, p-value = 0.1714
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.08150656 1.60191315
sample estimates:
ratio of variances
         0.3613398

> t.test(women_weight, men_weight, var.equal = TRUE, alternative="less")

        Two Sample t-test

data:  women_weight and men_weight
t = -2.7842, df = 16, p-value = 0.006633
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -6.298536
sample estimates:
```

```
mean of x mean of y 
 52.10000  68.98889 

> 
> t.test(weight ~ group, data = my_data, var.equal = TRUE, alternative = "greater")

	Two Sample t-test

data:  weight by group
t = 2.7842, df = 16, p-value = 0.006633
alternative hypothesis: true difference in means between group Man and group Woman is greater than 0
95 percent confidence interval:
 6.298536      Inf
sample estimates:
  mean in group Man mean in group Woman 
           68.98889            52.10000 

> 
> wilcox.test(x, y, alternative = "two.sided")

	Wilcoxon rank sum test with continuity correction

data:  x and y
W = 1024, p-value = 4.306e-12
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(x, y, alternative = "two.sided") :
  cannot compute exact p-value with ties
> 
> library(dplyr)
> group_by(my_data, group) %>%
+   summarise(
+     count = n(),
+     median = median(weight, na.rm = TRUE),
+     IQR = IQR(weight, na.rm = TRUE)
+   )
# A tibble: 2 × 4
  group count median   IQR
  <chr> <int>  <dbl> <dbl>
1 Man       9   67.3  10.9
2 Woman     9   48.8  15  
> 
> res <- wilcox.test(women_weight, men_weight)
Warning message:
In wilcox.test.default(women_weight, men_weight) :
  cannot compute exact p-value with ties
> res

	Wilcoxon rank sum test with continuity correction

data:  women_weight and men_weight
W = 15, p-value = 0.02712
alternative hypothesis: true location shift is not equal to 0
```

**End-Anamika kumari.**