Computational Stats

Tiago dos Santos

2018-10-16

Contents

To use the lab computers, the access credentials are:

usr: enc pwd: Ecom*2018

1 Lesson 1

```
x <- 3+5

ls()
```

```
## [1] "LatexOrOther" "datasetsDir"  "fig_basePath" "x"
```

1.1 Start by creating a vector

```
y <- c(2,5,9,8)

y[1:3]
```

```
## [1] 2 5 9
```

```
y[c(1,3)]
```

```
## [1] 2 9
```

1.1.0.1 Get the elements 1,2,3 from the vector

```r
y[1:3]
```

```
## [1] 2 5 9
```

1.1.0.2 Get the elements 1,3 from the vector

```r
y[c(1,3)]
```

```
## [1] 2 9
```

1.1.0.3 Get an array from 0 to 1, with a 0.001 step

```r
y <- 1:1000/1000
y <- seq(0,1,0.001)
```

1.1.0.4 Which values are lower than 0.008?

```r
isValueLowerThan <- y < 0.008
y[isValueLowerThan]
```

```
## [1] 0.000 0.001 0.002 0.003 0.004 0.005 0.006 0.007
```

```r
idxs <- which(y<0.08)
y[idxs]
```

```
##  [1] 0.000 0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 0.010
## [12] 0.011 0.012 0.013 0.014 0.015 0.016 0.017 0.018 0.019 0.020 0.021
## [23] 0.022 0.023 0.024 0.025 0.026 0.027 0.028 0.029 0.030 0.031 0.032
## [34] 0.033 0.034 0.035 0.036 0.037 0.038 0.039 0.040 0.041 0.042 0.043
## [45] 0.044 0.045 0.046 0.047 0.048 0.049 0.050 0.051 0.052 0.053 0.054
## [56] 0.055 0.056 0.057 0.058 0.059 0.060 0.061 0.062 0.063 0.064 0.065
## [67] 0.066 0.067 0.068 0.069 0.070 0.071 0.072 0.073 0.074 0.075 0.076
## [78] 0.077 0.078 0.079
```

1.1.0.5 Creating objects by repetition

```r
colors <- c("amarelo","verde","vermelho","azul")

rep(colors, 5)
```

```
##  [1] "amarelo"  "verde"    "vermelho" "azul"     "amarelo"  "verde"
##  [7] "vermelho" "azul"     "amarelo"  "verde"    "vermelho" "azul"
## [13] "amarelo"  "verde"    "vermelho" "azul"     "amarelo"  "verde"
## [19] "vermelho" "azul"
```

```r
print("===")
```

```
## [1] "==="
```

```r
rep(10,5)
```

```
## [1] 10 10 10 10 10
```

1.2 Now a Matrix!

```
M <- matrix(1:9, ncol=3)
M
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Transposing the Matrix

```
t(M)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

Accessing the Matrix

```
M[1,2]
```

```
## [1] 4
```

```
M[1,]
```

```
## [1] 1 4 7
```

```
M[,2]
```

```
## [1] 4 5 6
```

Matrix Operation

```
M2 <- t(M)

M+M2 # valuewise add
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    6   10   14
## [3,]   10   14   18
```

```
M*M2 # valuewise multiplication
```

```
##      [,1] [,2] [,3]
## [1,]    1    8   21
## [2,]    8   25   48
## [3,]   21   48   81
```

```
M%*%M2 # Matricial Multiplication
```

```
##      [,1] [,2] [,3]
## [1,]   66   78   90
## [2,]   78   93  108
## [3,]   90  108  126
```

1.2.0.1 Joining Matrixes

Matrix Operation

```
cbind(M,M2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    4    7    1    2    3
## [2,]    2    5    8    4    5    6
## [3,]    3    6    9    7    8    9
```

```
rbind(M,M2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
## [4,]    1    2    3
## [5,]    4    5    6
## [6,]    7    8    9
```

1.2.0.2 Inverting a matrix

```
#solve(M) # M must not be singular
```

1.3 DataFrames

```
y <- 1:10
y2 <- 11:20
y3 <- letters[1:10]

d1 <- data.frame(y,y2,y3)

d1
```

```
##     y y2 y3
## 1   1 11  a
## 2   2 12  b
## 3   3 13  c
## 4   4 14  d
## 5   5 15  e
## 6   6 16  f
## 7   7 17  g
## 8   8 18  h
## 9   9 19  i
## 10 10 20  j
```

1.4 Reading a Tab Separated File

```
emp <- read.table(file.path(datasetsDir,"empresas.txt"), header=F)
knitr::kable(head(emp))
```

V1

V2

V3

V4

V5

Soflor

2

5

10

3

Florinha

3

10

22

7

Flora

5

30

55

18

Floflo

2

5

12

4

Fazflor

3

15

28

8

Comercflor

2

10

18

5

```r
dim(emp)
```

```
## [1] 40  5
```

```r
names(emp) <- c("nome","n.socios","c.social","vmm","n.emp")
knitr::kable(head(emp))
```

nome

n.socios

c.social

vmm

n.emp

Soflor

2

5

10

3

Florinha

3

10

22

7

Flora

5

30

55

18

Floflo

2

5

12

4

Fazflor

3

15

28

8

Comercflor

2

10

18

5

```
emp$n.socios
```

```
##  [1] 2 3 5 2 3 2 3 4 6 5 2 3 2 3 2 3 3 2 5 2 2 3 3 2 2 2 2 4 4 3 2 2 4 2 2
## [36] 2 3 3 3 2
```

```
emp[,2]
```

```
##  [1] 2 3 5 2 3 2 3 4 6 5 2 3 2 3 2 3 3 2 5 2 2 3 3 2 2 2 2 4 4 3 2 2 4 2 2
## [36] 2 3 3 3 2
```

1.5 Generating data

```
set.seed(5)

emp$ant <- round(rnorm(dim(emp)[1],10,1))
```

1.6 Getting insights

```
summary(emp)
```

```
##          nome       n.socios       c.social          vmm        
##  Alecrim   : 1   Min.   :2.00   Min.   : 5.00   Min.   :  5.00  
##  Beijaflor : 1   1st Qu.:2.00   1st Qu.: 5.00   1st Qu.: 11.00  
##  Caflor    : 1   Median :3.00   Median :10.00   Median : 19.00  
##  Comercflor: 1   Mean   :2.85   Mean   :11.72   Mean   : 24.48  
##  Cravinho  : 1   3rd Qu.:3.00   3rd Qu.:15.00   3rd Qu.: 31.00  
##  Cravo     : 1   Max.   :6.00   Max.   :50.00   Max.   :100.00  
##  (Other)   :34                                                  
##      n.emp             ant    
##  Min.   : 2.000   Min.   : 8  
##  1st Qu.: 3.000   1st Qu.: 9  
##  Median : 5.500   Median :10  
##  Mean   : 6.225   Mean   :10  
##  3rd Qu.: 9.000   3rd Qu.:11  
##  Max.   :18.000   Max.   :12  
## 
```

```
mean(emp$n.socios)
```

```
## [1] 2.85
```

```
sd(emp$n.socios)
```

```
## [1] 1.051251
```

```
tapply(emp$vmm, emp$n.emp, mean) # vmm mean by number of employes
```

```
##          2          3          4          5          6          7 
##   8.714286  10.875000  12.666667  18.000000  22.000000  23.000000 
##          8          9         10         11         14         15 
##  28.000000  32.250000  45.000000  61.000000  45.000000 100.000000 
##         16         18 
##  55.000000  55.000000 
```

```r
tapply(emp$vmm, emp$n.emp, sd) # vmm sd by number of employes
```

```
##        2        3        4        5        6        7        8        9
## 2.627691 1.457738 1.154701 0.000000 1.414214 1.414214       NA 1.500000
##       10       11       14       15       16       18
##       NA 1.414214       NA       NA       NA       NA
```

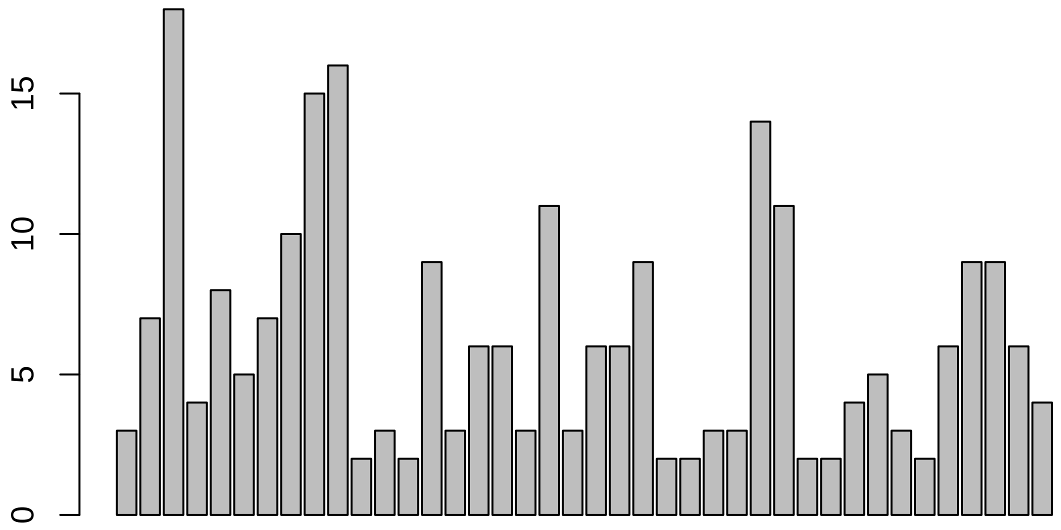
$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$$

$$S^2 = \frac{1}{N}\sum_{i=1}^{N}(X_i - \overline{X})^2 \qquad (1)$$

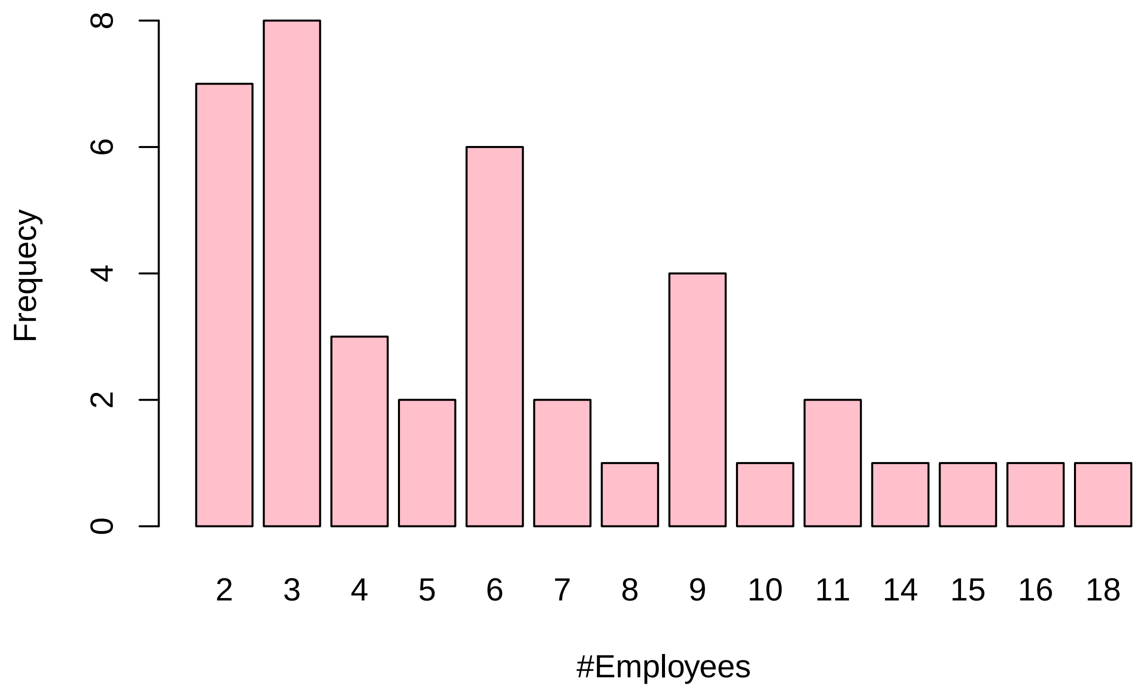
```r
table(emp$n.emp) #first line are values, second line is frequency
```

```
## 
##  2  3  4  5  6  7  8  9 10 11 14 15 16 18 
##  7  8  3  2  6  2  1  4  1  2  1  1  1  1
```

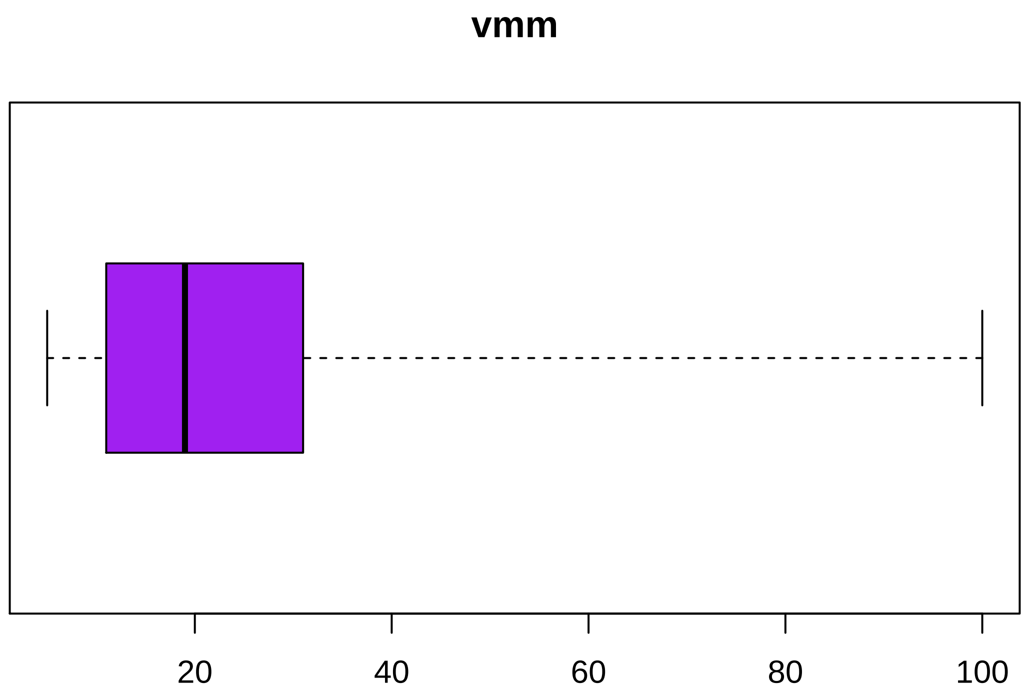
```r
barplot(emp$n.emp) # each company is a bin in x label, y is the number of employees
```



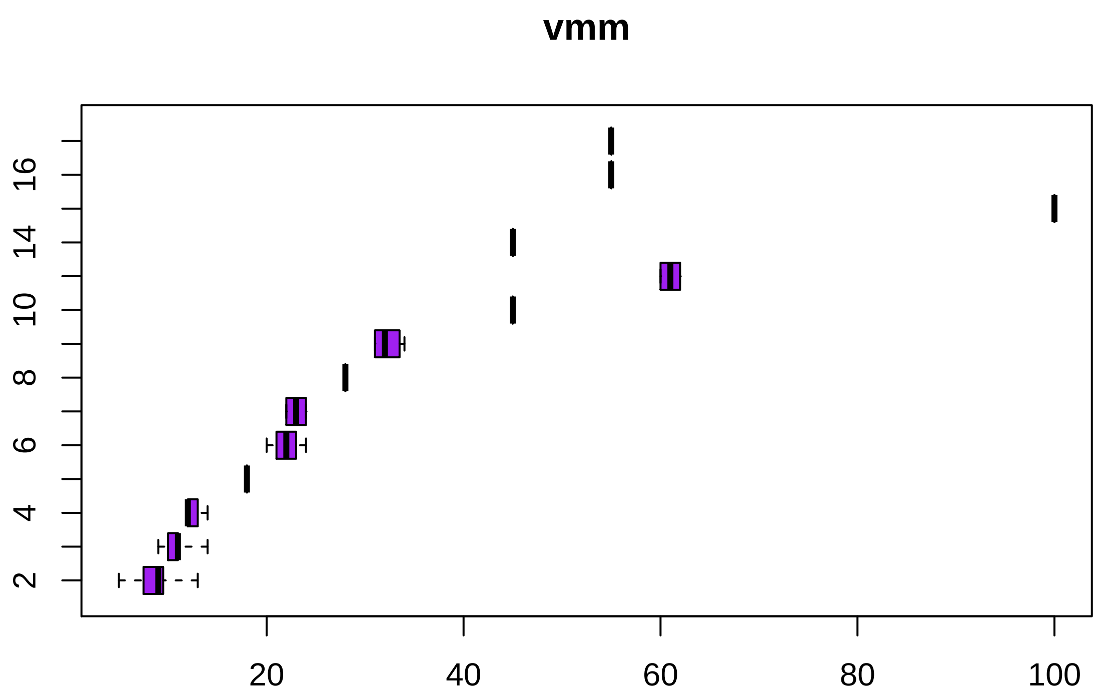
```r
barplot(table(emp$n.emp), xlab="#Employees", ylab="Frequecy", col="pink")
```



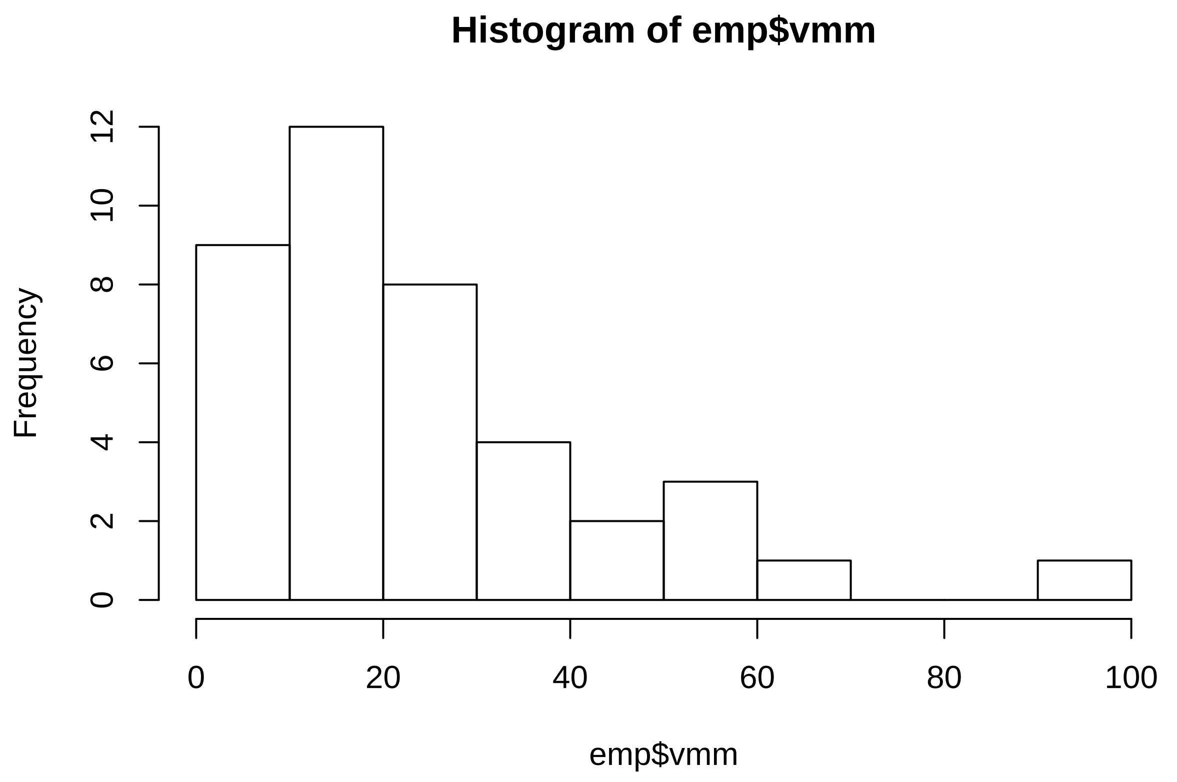
```
boxplot(emp$vmm,range=0,col="purple",horizontal=T,main="vmm")
```



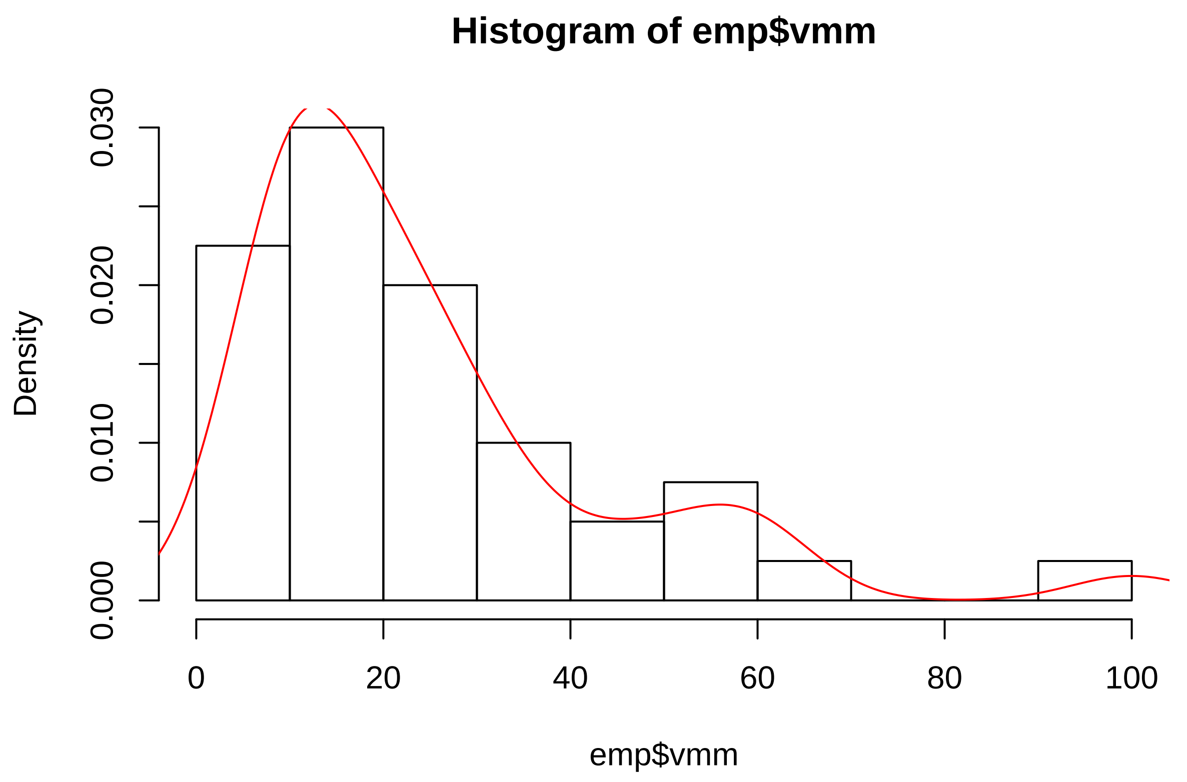
```
boxplot(emp$vmm ~ emp$n.emp,range=0,col="purple",horizontal=T,main="vmm")
```



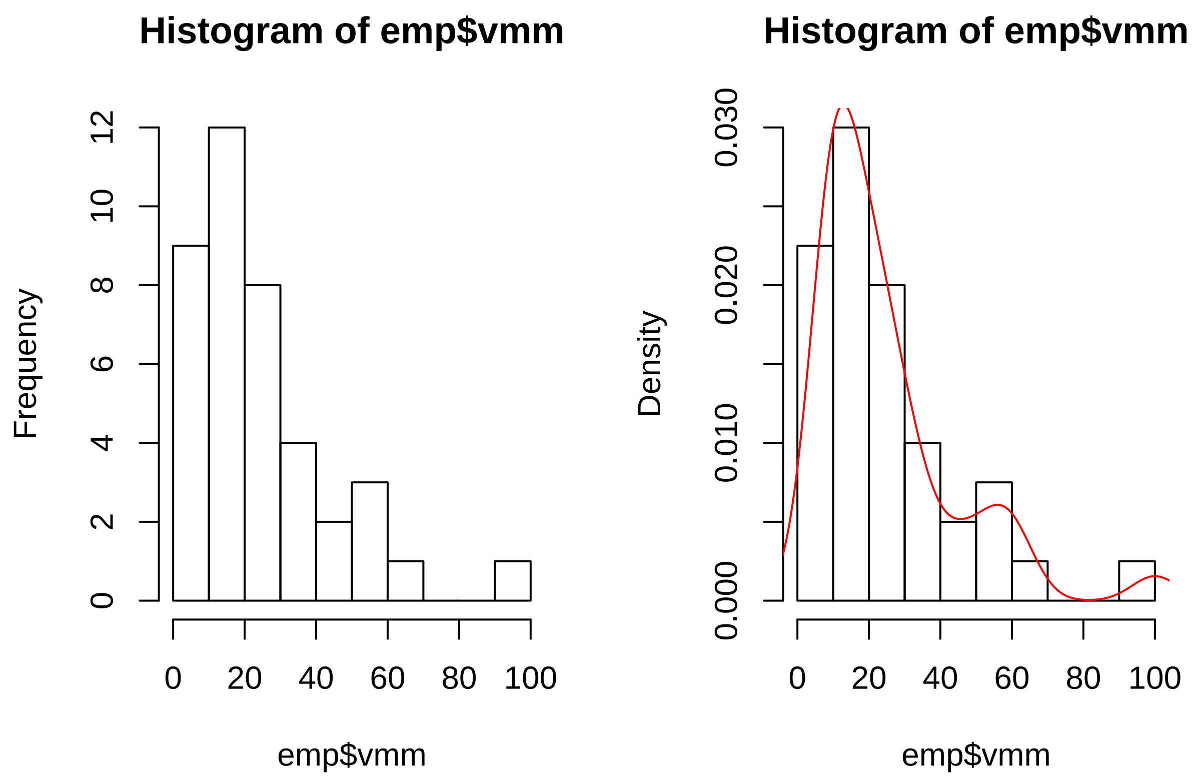
```
hist(emp$vmm)
```



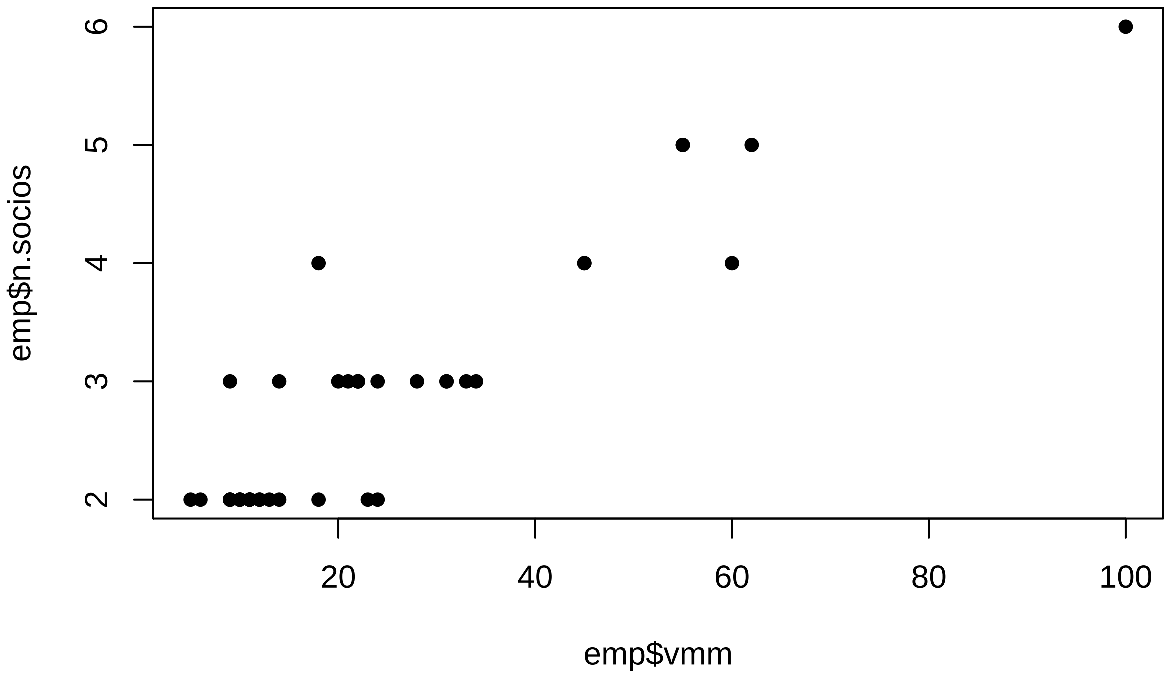
```
hist(emp$vmm, freq=F)
lines(density(emp$vmm),col=2)
```



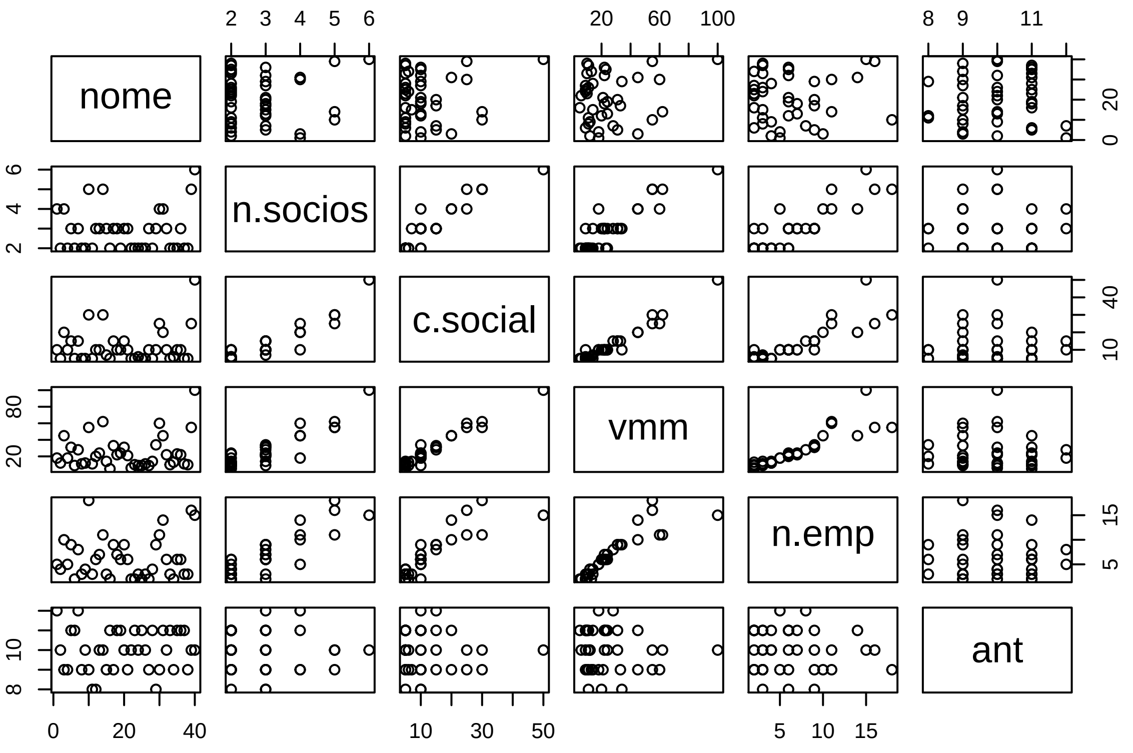
```r
par(mfrow=c(1,2))
hist(emp$vmm)

hist(emp$vmm, freq=F)
lines(density(emp$vmm),col=2)
```



```r
par(mfrow=c(1,1))

plot(emp$vmm,emp$n.socios,pch=16)
```



```r
plot(emp)
```



1.7 Lists

```
uma.lista <- list(
  um.vector=1:10,
  uma.palavra="olá",
  uma.matrix=M,
  outra.lista=list(
    a="flor",
    b=rep(3,5)
  )
)

uma.lista["um.vector"]
```

```
## $um.vector
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
uma.lista$um.vector
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
uma.lista[1]
```

```
## $um.vector
##  [1]  1  2  3  4  5  6  7  8  9 10
```

1.8 Functions

```
desconto <- function(price, discount=25){
  #Discount is a number between 0 and 100
  #calcula o desconto de um preço
  newPrice <- price*(1-discount/100)
  discount <- price - newPrice
  list(
    novo.preco=newPrice,
    desconto=discount)
}

desconto(1000,20)
```

```
## $novo.preco
## [1] 800
##
## $desconto
## [1] 200
```

```
desconto(1000,25)
```

```
## $novo.preco
## [1] 750
##
## $desconto
## [1] 250
```

This is how you function

2 Lessons 2

2.1 Random Variables and Vectors

2.1.1 Elements of probability

A random variable **X** is a function that takes an event space and return a value:

$$X : \Omega \to \mathbb{R}$$

2.1.2 Expected value

3 Lesson 3

```
g <- function(x){
  exp(x^2)
}

#create sample from uniform distribution
sample <- runif(10000)
sample.length <- length(sample)

mean(g(sample))
```

```
## [1] 1.46525
```

3.0.1 Estimating pi

```
g <- function(x){
  sqrt(1-x^2)
}

#create sample from uniform distribution
sample <- runif(100000000)

mean(g(sample))*4
```

```
## [1] 3.141693
```

```
gIndicatriz <- function(x,y){
  ifelse((x^2 + y^2) <= 1, 1, 0)
}

sampleX <- runif(1000000)
sampleY <- runif(1000000)
mean(gIndicatriz(sampleX,sampleY))*4
```

```
## [1] 3.141924
```

3.1 Hypothesis Testing

A statistical hypothesis is some conjecture about the distribution of one or more random variables. For each hypothesis designated by null hypothesis and denoted by $H0$, there is always an alternative hypothesis denoted by $H1$. We start the test by believing that $H0$ is true, and during the test we can discard that hypothesis only if the data points there.

Moreover, we can see these hypothesis testing as:

- A statistical hypothesis is some statement about the parameters of one or more populations (parametric tests) or about the distribution of the population (non-parametric tests).
- The goal of a test is to use the information of a data sample to decide (reject or no reject) about a conjecture over unknown aspects of a given population.

3.1.1 Types of error while infering through hypothesis testing

There are always some risk associated to statistical inference:

- *** Type 1 error ***: reject $H0$ when $H0$ is true (rejecting error, aka False Negative in ML nomenclature)
- *** Type 2 error ***: accept $H0$ when $H0$ is false (no rejecting error, aka False Positive in ML nomenclature)

3.1.2 Defining α to reduce a type of error

$$\alpha = P(Type1Err) = P(RejectingH0|H0istrue)$$

So, we define α as being the probability that we want for the Type 1 error - or how much are we willing to be prone to this type of error.

Therefone, α is called ***significance level*** of the test (a test that is very prone to errors is not very significant, right?)

In general, we assign a very small value to the probability of type I error (0.05 ou 0.01).

On the other end of the error spectrum, we define β as

$$\beta = P(Type2Err) = P(AcceptingH0|H0isfalse)$$

where $1 - \beta$ is called power of the test. The insight here is that the lower the β, the more "power" this test have.

3.1.3 Procedure to make a test using $p - value$

3.1.3.1 Wait, what is p-value?

(WIP)

3.1.4 Estimating test stats

The hypothesis being tested is the following:

We have a sample (the variable `popSample` below) of independent observations from a random variable that we know follows an exponencial distribution, with unknown parameter λ.

We want to test if $\lambda = 3$.

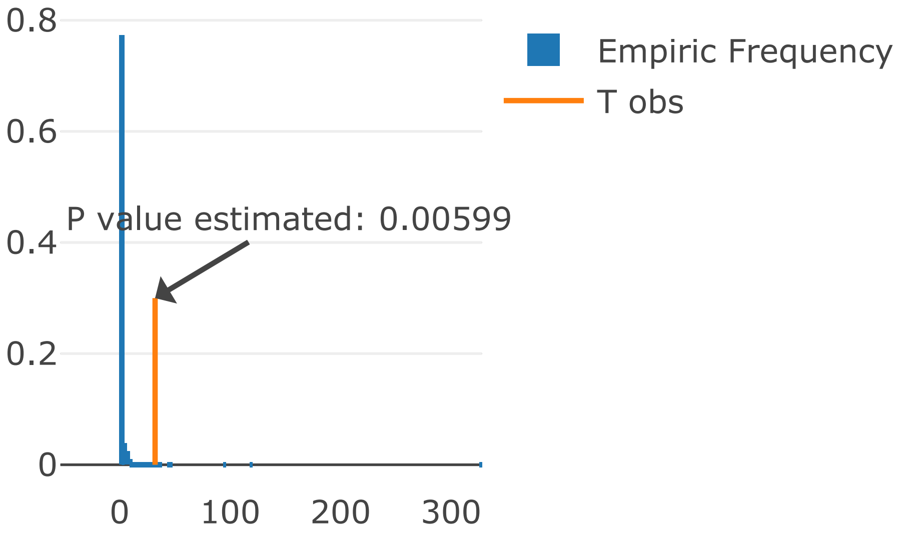
```r
popSample <- c(0.2,1.2,2.9,1.2,0.1,0.1,0.4,0.1,0.7,0.1,0.9,0.3,0.6,0.1,0.2,0.1,0.4,0.1,0.3,1.4)

lambdaEstimator <- function(sample){
  1/mean(sample)
}


parameter <- 3

testStatsEstimator <- function(sample,hypothesisLambda, estimatedLambda){
  sampleMean <- mean(sample)
  sampleLength <- length(sample)
  return(
    (
      1/((sampleMean*hypothesisLambda)^sampleLength))
        *
      (exp(
        sampleLength*
        (
          hypothesisLambda*sampleMean -1)
        )
     )
  )
}

tobs <- testStatsEstimator(popSample,parameter,lambdaEstimator(popSample))
```

Here, we will do the following 1000 times:

- we get a random sample from an exponential with $\lambda = 3$
- we obtain the estimated test statistic for this sample

By the end of this process, we will get 1000 values that represente possible values of the Test Statistic Function

```r
empiricDistTestStats <- sapply(1:1000,function(idx){
  sampleTest <- rexp(length(popSample),parameter)
  testStatsEstimator(sampleTest,parameter,lambdaEstimator(sampleTest))
})

empiricDistTestStats <- c(empiricDistTestStats,tobs)
empiricDistTestStats.df <- as.data.frame(empiricDistTestStats)
names(empiricDistTestStats.df) <- c("values")

empiricFrequency <- empiricDistTestStats.df %>% dplyr::group_by(values) %>% dplyr::summarise(n=n())

p_value_estimated <- sum(empiricFrequency[empiricFrequency$values >= tobs,]$n)/sum(empiricFrequency$n)


a <- list(
  text = paste0("P value estimated: " , round(p_value_estimated,5)),
  x = tobs,
  y = 0.3,
  xref = "x",
  yref = "y",
  ax = 50
)

plotly::plot_ly(
  x = empiricDistTestStats
  , type="histogram"
  , histnorm = "probability"
  , name = "Empiric Frequency") %>%
plotly::add_segments(
  x = tobs, xend = tobs, y = 0, yend = 0.3, name = "T obs"
) %>% plotly::layout(annotations=a)
```



4 Deliverables

5 Deliveable 1

5.1 Exercise 1

1. Consider the continuous random variable X with pdf:

$$f(x) = \begin{cases} \frac{4}{3}(x^3 + x) & 0 < x < 1 \\ 0, & \text{for all others } x \text{ values} \end{cases}$$

Now consider the random variable $Y = g(X)$, where $g(x) = log(x^2 + 4)$. Estimate $P(1.3 < Y < 1.5)$ using the Monte Carlo Method, as well as the estimator standard deviation.

$$P(1.3 < Y < 1.5) \quad = \quad P(1.3 < g(x) < 1.5) \quad = \quad P(1.3 < log(x^2 + 4) < 1.5)$$

Given that x only present values between $0 < x < 1$, that imples:

- the minimun value of $log(x^2 + 4)$ is $log(4)$
- the maximum value of $log(x^2 + 4)$ is $log(5)$

Therefore, we know that:

$$P(1.3 < log(x^2 + 4) < log(4)) = 0$$

With this taken into consideration, the probability that we want to calculate is:

$$P(log(4) < log(x^2 + 4) < 1.5)$$

Which we can know expand into:

$$P(log(4) < log(x^2+4) < 1.5) \quad = \quad P(4 < x^2+4 < e^{1.5}) \quad = \quad P(0 < x^2 < e^{1.5}-4) \quad = \quad P(0 < x < \sqrt{e^{1.5} - 4})$$

So, we now know that the probability we want to calculate can be obtain by the following integral:

$$\int_0^{\sqrt{e^{1.5}-4}} \frac{4}{3}(x^3 + x)dx \tag{2}$$

```
mc <- function(t){
  k <- sqrt(exp(1.5)-4)
  return ( (4/3) * ((k*t)^3 + k*t) *k )
}

#t follows an uniform

sample <- runif(100000)
pEst <- mean(mc(sample))

varEstimator <- (1/(length(sample)^2))*sum((mc(sample)-pEst)^2)
df <- data.frame(
  probEstimated = pEst,
  varianceMC = varEstimator
)

knitr::kable(df, format = knitr:::pandoc_to())
```

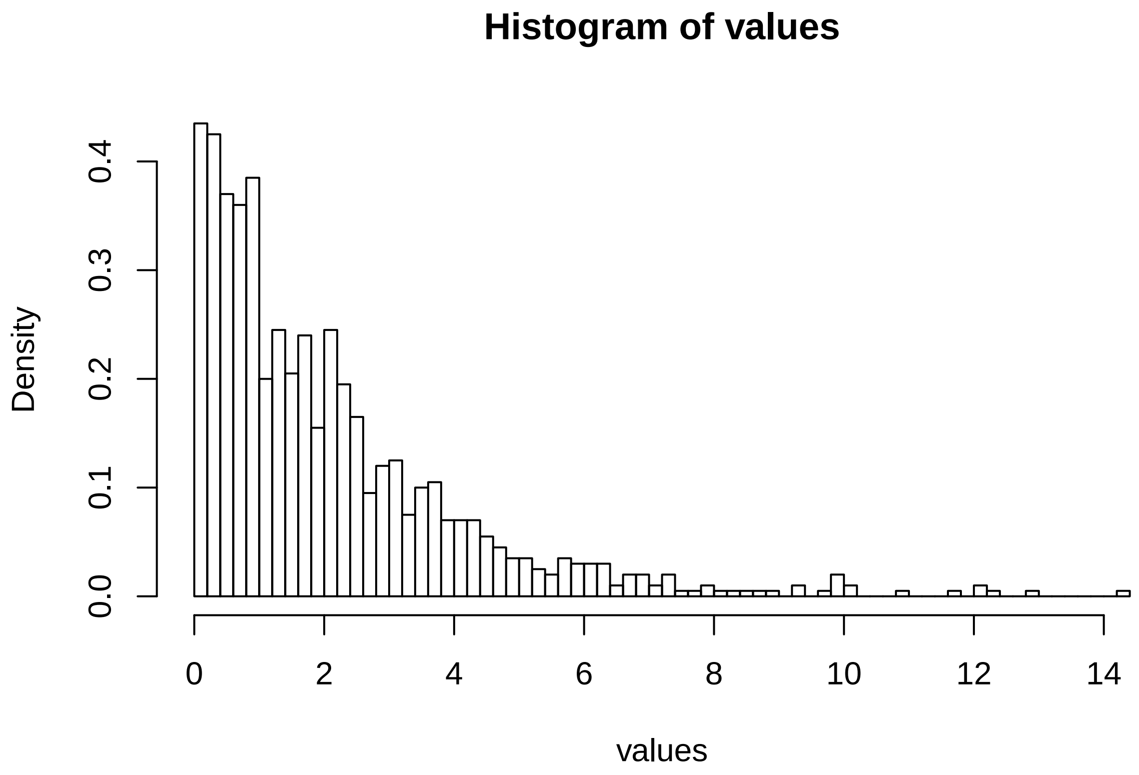
probEstimated	varianceMC
0.397763	7e-07

5.2 Exercise 2

5.2.1 2.1

5.2.2 2.2

```
lambda <- 0.5
samples <- runif(1000)


inverseExp <- function(u, lambda){
  -(1/lambda)*log(1-u)
}

values <- inverseExp(samples, lambda)

hist(values, breaks=100, freq = F)
```



```
g <- function(x){
  exp(sqrt(x))*(2/(sqrt(2*pi)))*x^(-1/2)
}

X <- runif(10000)
Y <- runif(10000)

EX <- mean(g(inverseExp(X, lambda)))
EY <- mean(g(inverseExp(Y, lambda)))
```