PEC2_SAID_BOUATRA_BELOUAFI.R

saidbouatra

Mon Dec 4 14:59:12 2017

```
getwd()
```

```
## [1] "/Users/saidbouatra/Statistical_software/statist_R_dev_git/StatS_P2"
```

```
setwd("/Users/saidbouatra/Statistical_software/statist_R_dev_git/StatS_P2")

#   Ejercicio 1: Una técnica operatoria tiene un 4% de complicaciones.
# ¿Cuál es la probabilidad de que si se realiza la técnica 96 veces haya 2 complicaciones?

dbinom(2, size = 96 , prob = 0.04)
```

```
## [1] 0.1572464
```

```
# ¿Cuál es la probabilidad de que si se realiza la técnica 101 veces haya más de 4 complicaciones?

pbinom(4 , size = 101 , prob = 0.04 , lower.tail = FALSE)
```

```
## [1] 0.3791115
```

```
# Ejercicio 2: La duración media de la estancia hospitalaria de una enfermedad es de 9 ± 3 días (media
# Suponiendo que se trata de una distribución normal, calcula la probabilidad de que una estancia dure

pnorm(15,9,3) - pnorm(8,9,3)
```

```
## [1] 0.6078085
```

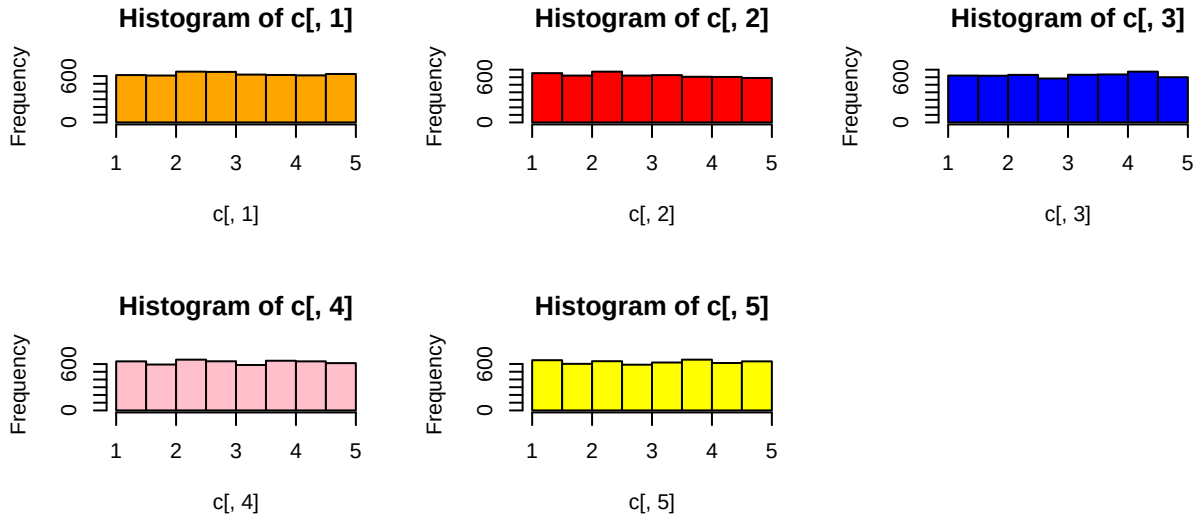
```
# Hallar el valor crítico de t para el que el área bajo la cola derecha de la f. de densidad de la vari


qt(0.05 , 16 , lower.tail = FALSE)
```

```
## [1] 1.745884
```

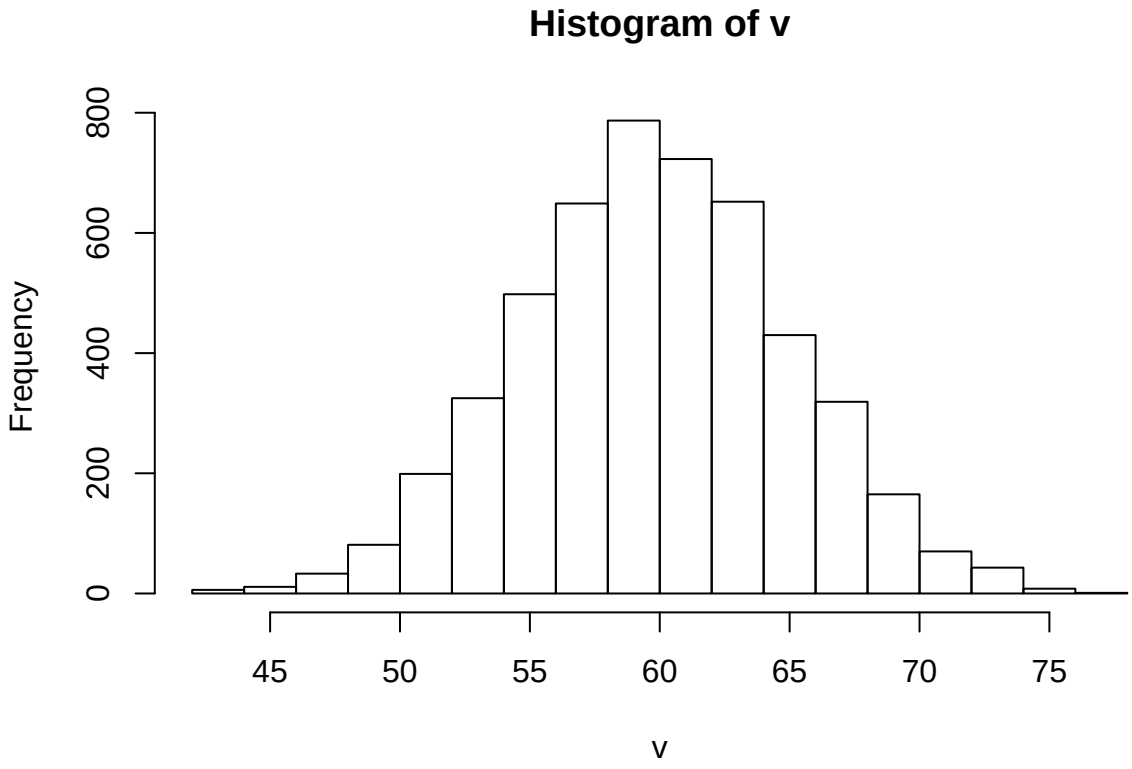
```
## Sección 2. Teorema Central del Límite.
# Ejercicio 4  (Demostración):  A. Genera 100,000 valores aleatorios de acuerdo a una distribución unif
set.seed(1234567)
a <- runif(100000 , min = 1 , max = 5)
c <- matrix(a,ncol = 20)
par(mfrow = c(3,3))
hist(c[,1],col = "orange")
hist(c[,2],col = "red")
hist(c[,3],col = "blue")
hist(c[,4],col = "pink")
hist(c[,5], col = "yellow")

# C. Crea un vector que guarde la suma de los valores por fila.

v <- apply(c,1,sum)
head(v,5)
```

```
## [1] 54.85704 54.97234 66.87867 64.64131 64.21087
```

```r
#Realiza un histograma con referencia a la distribución normal.
par(mfrow = c(1,1))
```




```r
hist(v,breaks = "Sturges")
hist(dnorm(a), mean(a),sd(a),add = TRUE ,col = "red")
```




```r
# diagrama cuantil-cuantil (qq.plot) con referencia a la distribución normal.



# Sección 3. Funciones y estructuras de control.
# Ejercicio 6
```

```r
check_natural <- function(x) {
  divisores <- c()
  if (as.integer(x) == x) {
    cat(x ," es natural y sus divisores son:","\n")
    for (i in (1:x)) {
      if (x %% i == 0) {
    divisores <- c(i)
      cat(divisores,sep = "\n")
      }}
  } else {
    warning(x," no es un numero natural")
  }
  }
# ejecutamos dos numeros para verificar la funcion check_natural.
check_natural(6)
```

```
## 6  es natural y sus divisores son:
## 1
## 2
## 3
## 6
```

```r
check_natural(5.5)
```

```
## Warning in check_natural(5.5): 5.5 no es un numero natural
```

```r
# Ejercicio 7

my_fibo <- function(n) {
  fibonacci <- numeric(n)
  fibonacci[1] <- fibonacci[2] <- 1
  for( i in 3:n ) {
    fibonacci[i] <- fibonacci[i - 2] + fibonacci[i - 1]
   print(fibonacci[i])
}}
my_fibo(12) # solo para verificar la funcion.
```

```
## [1] 2
## [1] 3
## [1] 5
## [1] 8
## [1] 13
## [1] 21
## [1] 34
## [1] 55
## [1] 89
## [1] 144
```

```r
# Ejercicio 8: Centrémonos en la familia "Apply". Explica para qué usamos estas funciones y genera un e
```