Probabilidad y simulación con R

PID_00293467

Marta Casals Fontanet
Alícia Vila Grifo

Tiempo mínimo de dedicación recomendado: 3 horas

Universitat Oberta de Catalunya

Marta Casals Fontanet

Estadística de formación y profesión. Licenciada en ITM. Máster en Educación y nuevas tecnologías. Profesora de Matemáticas en secundaria y universidad. Profesora colaboradora en el máster de Bioinformática y bioestadística de la UOC y la UB.

Alicia Vila Grifo

Licenciada en Matemáticas por la Universidad de Valencia y profesora consultora de Probabilidad, Estadística y Análisis de Datos en diferentes estudios de la UOC. Actualmente es profesora funcionaria de Informática en el Departamento de Educación de la Generalitat de Cataluña, en los ámbitos de programación y bases de datos. También participa en la coordinación de proyectos de aplicaciones web y de análisis de datos.

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por los profesores: Álvaro Leitao Rodríguez y David Cantón Fabà

Cómo citar este recurso de aprendizaje con el estilo Harvard:

Casals, M. y Vila A. (2024). *Probabilidad y simulación con R.* [Recurso de aprendizaje textual]. 1.a ed. Fundació Universitat Oberta de Catalunya (FUOC).

Primera edición: febrero 2024

Av. Tibidabo, 39-43, 08035 Barcelona
Autoría: Marta Casals Fontanet, Alícia Vila Grifo
Producción: FUOC

Índice

1. Introducción al LAB4

El **LAB4** es un recurso didáctico complementario de la asignatura **Software para el análisis de datos (SAD)** del máster interuniversitario de Bioinformática y Bioestadística de la Universitat Oberta de Catalunya (UOC) y la Universidad de Barcelona (UB).

Este **LAB4** forma parte de un conjunto de laboratorios prácticos que conjugan contenidos teóricos con ejemplos, ejercicios y casos prácticos reales del ámbito de conocimiento del máster.

El **LAB4** explica los conceptos básicos de simulación y probabilidad con el lenguaje **R** y su aplicación al ámbito biosanitario. En realidad, aunque **R** es un lenguaje de programación, no suele utilizarse para desarrollar grandes proyectos, sino más bien para realizar paquetes de funciones o bloques de programas que puedan ser reutilizados con un objetivo específico. Por ello, las aplicaciones más habituales suelen estar dirigidas a la extracción, manipulación, tratamiento e interpretación de conjuntos de datos con fines específicos.

En este **LAB4**, se introduce al estudiante en la simulación y la probabilidad con **R**.

2. Instrucciones del LAB4

El **LAB4** contiene una serie de apartados con introducciones teóricas, ejemplos y casos prácticos, además de otros ejercicios que se propondrán para que sean resueltos por el estudiante. La temporización y las pautas para trabajar este laboratorio serán indicadas en el aula de la asignatura. También incluye la propuesta de solución de los diversos ejercicios y casos prácticos, que servirá para que el estudiante pueda autoevaluar las soluciones de los ejercicios realizados por él mismo.

En ocasiones, para realizar los ejercicios, se utilizarán conjuntos de datos de paquetes propios de **RStudio**, como es el caso de *biopsy*, *anorexia* y *birthwt* del paquete *MASS*, *iris* y *airquality* del paquete **datasets,** así como también otros conjuntos de datos abiertos de repositorios externos. Todos los ejercicios se realizarán en el entorno de desarrollo integrado **RStudio** (https://cran.rstudio.com/) y se usará **R** como lenguaje de programación para la informática estadística y los gráficos.

3. Contenidos del LAB4

En este **LAB4** trabajaremos los siguientes contenidos:

- Probabilidad con R.
- Distribuciones de probabilidad.
- Tipos de distribuciones.
- Simulación con R.
- Extracción de muestras.
- Ejercicios y casos prácticos en R.

4. Probabilidad con R

La probabilidad es la parte de la estadística y la simulación que estudia algunos experimentos y fenómenos aleatorios. Se usa, en muchas ocasiones, para predecir el éxito o fracaso de un proceso. Para poder entender cómo calcular la probabilidad de ocurrencia de un experimento, pondremos un ejemplo empírico.

Ejemplo 1:

Lanzamos una moneda (cara y cruz) 30 veces y buscamos la probabilidad de que ocurra cada una y anotamos los 30 resultados en un vector (cara = 1; cruz = 0).

```
moneda<-c(0,1,0,0,1,0,1,1,0,0,1,0,0,1,1,0,0,1,0,1,0,1,0,0,1,0,1,1,0,1)
#resultados del lanzamiento de una moneda 30 veces
frec<- table(moneda)/length(moneda) #buscamos las frecuencias y generamos una tabla
print(frec) #tabla de frecuencias

## moneda
##         0         1
## 0.5333333 0.4666667
```

La probabilidad de que salga cara será 0.533 y la probabilidad de que salga cruz será 0,467.

5. Distribuciones de probabilidad

Las **distribuciones de probabilidad** son funciones que definen la probabilidad de ocurrencia de cada valor de una variable aleatoria. Podemos clasificar las distribuciones entre *discretas* y *continuas*.

En **R**, cada distribución de probabilidad se nombra mediante una palabra clave. Por defecto, al instalar **R**, se carga el paquete *stats()* que implementa muchas funciones para la realización de cálculos asociados a las distribuciones.

5.1. Distribuciones discretas

Comando en R	**Distribución estadística**
Binom((k,n,p))	Binomial
pois()	Poisson
geom()	Geométrica
hyper()	Hipergeométrica
nbinom()	Binomial Negativa

5.2. Distribuciones continuas

Comando en R	Distribución estadística
unif()	Uniforme
norm()	Normal
t()	tStudent
F()	F Fisher
chisq()	Chi-Cuadrado
exp()	Exponencial

Los comandos para estas funciones se nombran anteponiendo un prefijo al nombre del comando en **R** de la distribución. Los prefijos son los siguientes:

- **d**: función de densidad o de probabilidad.
- **p**: función de distribución.
- **q**: función para el cálculo de cuantiles.
- **r**: función para simular datos con dicha distribución.

Así, por ejemplo, si **X ~ B(n,p)** sigue una distribución binomial (discreta) y queremos calcular:

a) **P(X=k)** usaremos *dbinom(k,n,p)*
b) **P(X≤k)** usaremos *pbinom(k,n,p)*
c) **qa=min{x:P(X≤x)≥a}** usaremos *qbinom(a,n,p)*
d) **m valores aleatorios** usaremos *rbinom(m,n,p)*

Por otro lado, si nuestra variable sigue una distribución continua **X~ N(μ,σ)** y queremos calcular:

a) **f(x)** usaremos *dnorm(x,μ,σ)*
b) **P(X<k)** usaremos *pnorm(x,μ,σ)*
c) **qa=min{x:P(X≤x)≥a}** usaremos *qnorm(a,μ,σ)*
d) **m valores aleatorios** usaremos *rnorm(m,μ,σ)*

Ejemplo 2:

Vamos a probar estos comandos en una distribución discreta como la **distribución binomial**.

a) Supongamos una distribución binomial **X~B(9,0.7)** y calculemos **P(X=8)**:

```
dbinom(8, 9, 0.7) #calculamos con la función de densidad
```

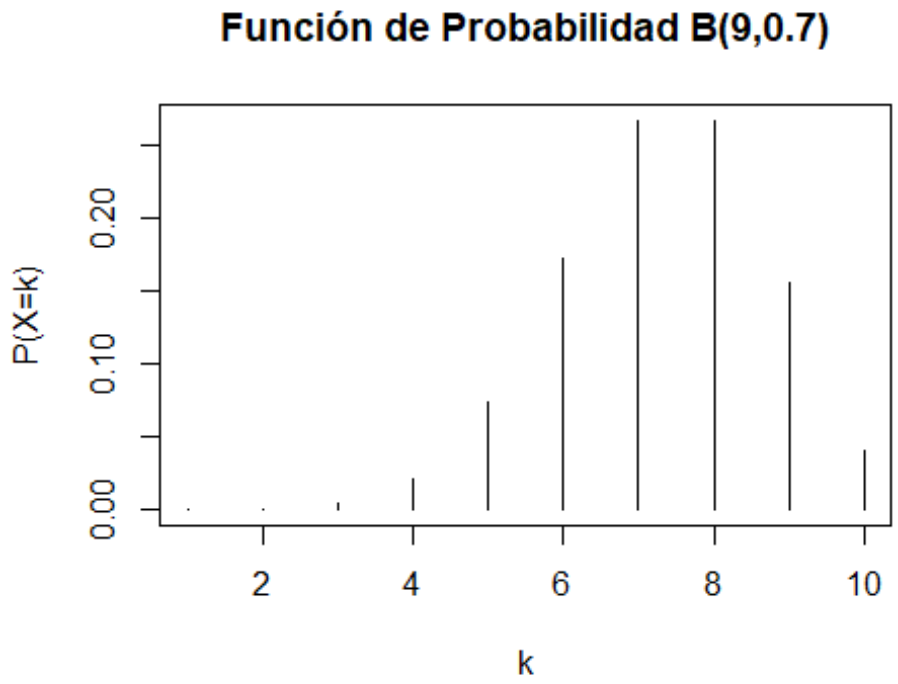
```
## [1] 0.1556496
```

b) Supongamos una distribución binomial **X~B(9,0.7)** y calculemos **P(X≤8)**:

```
set.seed(999)
pbinom(8, 9, 0.7) # función de distribución de la binomial

## [1] 0.9596464

sum(dbinom(8, 9, 0.7)) # otra manera de calcular la probabilidad

## [1] 0.1556496
```

c) si queremos simular 30 valores aleatorios de esta función:

```
rbinom(30,9, 0.7) #generación de 100 valores aleatorios

##  [1] 7 6 8 5 5 8 6 8 7 6 7 6 9 3 8 5 8 5 8 8 4 6 6 7 5 5 6 6 5 7

plot(dbinom(0:9,9,0.7),type="h",xlab="k",ylab="P(X=k)",main="Función de
Probabilidad B(9,0.7)")
```




```
x<-pbinom(0:10,9,0.7)
x

##  [1] 0.000019683 0.000433026 0.004290894 0.025294842 0.098808660
0.270340902
##  [7] 0.537168834 0.803996766 0.959646393 1.000000000 1.000000000
```

Ejemplo 3:

Vamos a probar estos comandos en una distribución continua como la **distribución normal**.

a) Supongamos una distribución normal **X~N(163,8)** y calculemos *f(178)*.

```
set.seed(999)
dnorm(178,163,8)

## [1] 0.008598284
```

b) Supongamos una *distribución normal* tal que **X~N(50,5)** y calculemos la probabilidad de que X sea mayor a 48, es decir, *P(X>48)*:

```
pnorm(48, 50, 5, lower.tail = FALSE)

## [1] 0.6554217
```

c) Supongamos que queremos calcular que X sea mayor o igual a 45 y menor que 55 en una variable normal, tal que **X~N(40,3)**:

```
pnorm(55, 40, 3) - pnorm(45, 40, 3)

## [1] 0.04779007
```

d) ¿Cuál es el valor de X que deja un 85 % por debajo de este en la distribución del apartado anterior?

```
qnorm (0.85, 40, 3)

## [1] 43.1093
```

6. Tipos de distribuciones usadas en probabilidad

6.1. Distribución uniforme ~ Generación de números aleatorios

La *distribución uniforme* se suele usar para la generación de números aleatorios. Generación de 100 números aleatorios entre 0 y 1 redondeados en una cifra decimal.

```
set.seed(999)
data_uni <- runif(100, 0, 1) #generamos los 100 números entre 0 y 1
vect_uni <- round(data_uni, 1) #redondeamos a una posición decimal
table(vect_uni) #tabla de frecuencias absolutas de cada valor

## vect_uni
##   0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9   1
##   4  15   9   6   8  12  15   8  12   8   3
```

6.2. Distribución binomial ~ Monedas y dados

La *distribución binominal* se asimila con el cálculo de probabilidad en el juego. Por ejemplo, con monedas y juegos. También en el ámbito biosanitario se puede ver cómo la proporción de individuos verifican una cierta característica dicotómica (estar enfermo o no, ser mujer o no, fumar o no fumar, etc.).

Vamos a ver ejemplos de probabilidad de una *binomial*. Tiramos una moneda perfecta y queremos:

a) Calcular la probabilidad de obtener 4 caras al lanzar 6 veces una moneda perfecta.

```
dbinom(4, 6, 0.5) #En este caso sería P[X=4] con X→B(6,0.5)

## [1] 0.234375
```

b) Calcular la probabilidad de obtener como máximo 4 caras al lanzar 6 veces una moneda perfecta.

```
pbinom(4, 6, 0.5) #En este caso sería P[X≤4] con X→B(6,0.5)

## [1] 0.890625
```

c) Generar 10 valores seudoaleatorios de una B (6, 0.5).

```
rbinom(10, 6, 0,5) #generamos 10 valores aleatorios

##  [1] 3 5 2 5 5 3 3 2 3 3
```

Ejemplo 4:

Como ejemplo usaremos un ensayo clínico en el que participan **60 pacientes**. La probabilidad de que el paciente fume es 0.45.

a) Queremos calcular la probabilidad de que exactamente 20 pacientes sean fumadores. Aquí se pide calcular P [X = 20], siendo X una variable aleatoria que representa los pacientes fumadores.

```
n=40 #número de pacientes
p=0.35 #probabilidad de que el paciente fume
val=20 #valor exacto

#podemos calcular el gráfico que representa esta probabilidad exacta
library(ggplot2)
dbinom(val,n,p)

## [1] 0.01901183

#ggplot(df, aes(x = x, y = y, fill=factor(ifelse(x==val,"Valor","Resto")))) +
#geom_bar(stat = "identity",width = 0.75) + xlab("x") + ylab("Densidad") +
#ggtitle("Distribución binomial") +
#scale_fill_manual(name = "", values=c("grey50","blue")) + theme_bw()
```

b) En el caso de que nos pidan que al menos **20 pacientes** sean fumadores y no de forma exacta. Aquí se pide calcular P [X ≥ 20] = P [X > 20] o, lo que sería lo mismo, 1 − P [X ≤ 19].

```
#miramos las dos opciones de cálculo de probabilidad
val1=19
pbinom(val1,n,p, lower.tail = FALSE) #asumimos que es la área de la derecha
con lower.tail

## [1] 0.03629264

1- pbinom(val1, n, p)

## [1] 0.03629264

# en este caso, el gráfico nos determina una zona

#ggplot(df, aes(x = x, y = y, fill=factor(ifelse(x>val1,"Valor","Resto")))) +
#geom_bar(stat = "identity",width=0.75) + xlab("x") + ylab("Densidad") +
#ggtitle("Distribución binomial") +
#scale_fill_manual(name = "", values=c("grey50","blue")) + theme_bw()
```

6.3. Distribución normal ~ MUESTREO

La *distribución normal* se usa en muchas ocasiones para generar simulaciones y generar valores de muestreo. Es una de las distribuciones más importante y relevante en estadística, la podemos encontrar en muchas de las variables que encontramos en la naturaleza y tiene una serie de propiedades que son muy interesantes.

Ejemplo 5:

Anteriormente ya hemos visto algún ejemplo de cálculo con la distribución normal, pero veamos otros con más detalle.

a) Calcular la probabilidad de que P(X≤ 35) de una distribución normal con media = 50 y varianza = 25.

```
set.seed(999)
desv <- sqrt(25) #calculamos la desviación antes de calcular nuestra
probabilidad
pnorm(42,50, desv)

## [1] 0.05479929
```

b) Calcular la probabilidad de que P(X >35) con la misma probabilidad.

```
pnorm(35, 50, sqrt(25), lower.tail = FALSE)

## [1] 0.9986501
```

c) Generar un conjunto con **100** datos que sigan una distribución normal con una **media = 25** y una **desviación típica = 4**, para poder realizar un ejercicio de estadística que nos ha marcado la consultora.

```
set.seed(999)
rnorm(100, 25, 4)
```

```
##   [1] 23.87304 19.74976 28.18074 26.08028 23.89077 22.73591 17.48537
19.93284
##   [9] 21.12900 20.51596 30.30185 25.53591 28.75500 25.69015 28.83060
19.54926
##  [17] 25.27334 25.40263 28.60538 16.70257 20.08575 27.57218 23.56095
26.17614
##  [25] 20.49893 27.56906 20.57305 21.46064 18.78362 24.49328 34.53066
27.40510
##  [33] 25.71745 29.32213 24.01275 16.54505 23.51789 27.09147 27.07122
19.38996
##  [41] 23.05745 25.03399 19.87155 20.55368 26.20266 26.10592 16.79649
25.05676
##  [49] 27.32907 24.86109 24.53334 22.42007 31.97765 26.46438 24.73276
26.13045
##  [57] 27.27078 19.88314 26.74148 22.73800 21.30675 29.65982 29.16827
29.41531
##  [65] 24.92570 20.41217 19.36728 23.87069 23.32892 28.98665 24.57486
24.72239
##  [73] 28.79880 23.33387 28.89600 25.24917 27.15369 16.74071 26.74573
24.35909
##  [81] 22.42831 28.94119 20.08571 25.34090 20.18362 23.49261 30.45459
23.98847
##  [89] 29.02596 26.74860 31.63415 25.11074 21.08512 30.13492 20.48103
29.18663
##  [97] 22.68827 23.82482 24.03289 25.87979
```

d) Se sabe que el peso en recién nacidos sigue una normal con **media = 3 y desv.típica = 0.30**. Calcular la probabilidad de que un bebé pese entre **2,5** y **3 kg** (incluidos también).

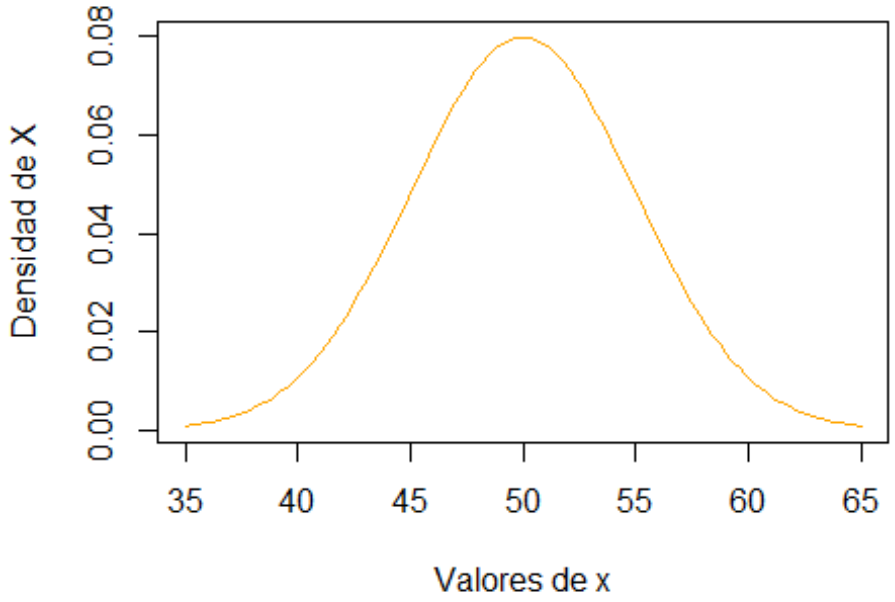
```
# se puede calcular como P [X ≤ 3] – P [X ≤ 2.5]
pnorm(3, 3, 0.3) - pnorm (2.5, 3, 0.3)
```

```
## [1] 0.4522096
```

La función *dnomr()* no tiene mucho sentido en una distribución continua, pero nos puede ser de mucha utilidad cuando necesitemos calcular el gráfico de probabilidad.

e) Construir el gráfico de la distribución de probabilidad de X, usando el comando curve. Buscar la curva de una distribución normal de **media = 50** y **varianza = 25**.

```
curve(dnorm(x, mean=50, sd= sqrt(25)), xlim = c(35,65), xlab="Valores de x",
ylab= "Densidad de X", col="orange")
```



Nota: Por defecto, en R, cuando usamos funciones de la normal, si no se especifica previamente, nuestra variable siempre será μ = 0 y σ = 1.

6.4. Distribución de Poisson ~ Procesos y colas

La distribución de Poisson se suele usar en colas y procesos industriales. El parámetro lambda (λ) representa el número de sucesos independientes que ocurren a una velocidad constante en un momento, intervalo de tiempo y en una región o espacio.

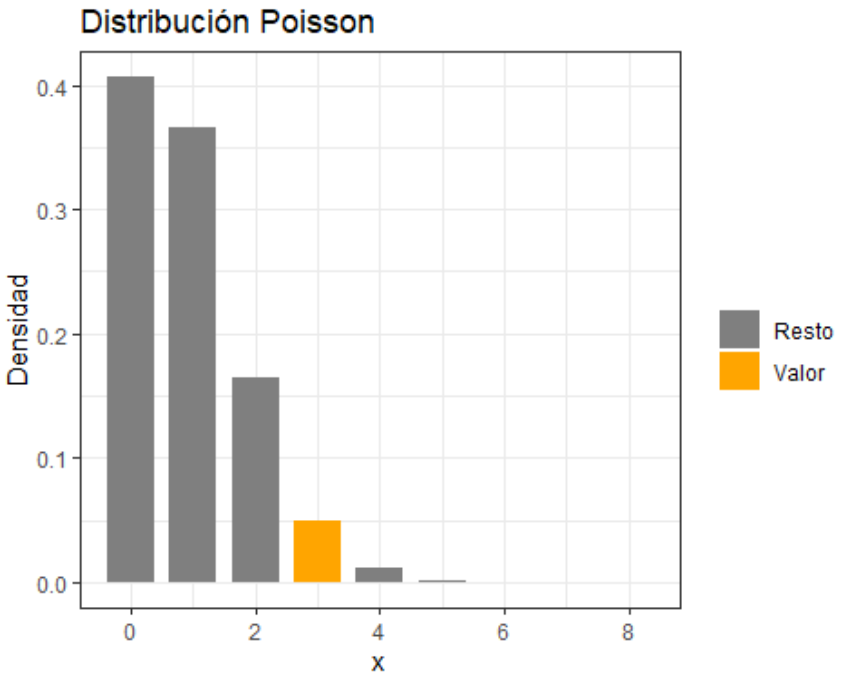
Para entender mejor la distribución de Poisson realizaremos un ejemplo práctico:

Ejemplo 6:

En un hospital se sabe que llegan en promedio 20 enfermos por hora para ser atendidos. Asumiendo que es un fenómeno de Poisson y que el intervalo de referencia es una hora, la cantidad de enfermos por hora define la tasa promedio **(lambda = 20)**:

a) ¿Qué probabilidad hay de que en la siguiente hora lleguen 3 enfermos exactamente en un hospital con pacientes con una enfermedad que sigue un lambda = 0.90?

```
#podemos graficar también

rank = 0:8
val2=3
lambda=0.90
df = data.frame(x = rank, y = dpois(rank, lambda=0.90))
ggplot(df, aes(x = x, y = y, fill=factor(ifelse(x==val2,"Valor","Resto")))) +
geom_bar(stat = "identity",width = 0.75) + xlab("x") + ylab("Densidad") +
ggtitle("Distribución Poisson") + scale_fill_manual(name = "",
values=c("grey50","orange")) + theme_bw()
```



b) ¿Qué probabilidad hay de que en la siguiente hora llegue un **máximo de 12** enfermos al hospital?

```
#Podemos sumar las probabilidades individuales
dpois(0, lambda = 20) + dpois(1, lambda = 20) + dpois(2, lambda = 20) +
dpois(3, lambda = 20) + dpois(4, lambda = 20) + dpois(5, lambda = 20) +
dpois(6, lambda = 20) + dpois(7, lambda = 20) + dpois(8, lambda = 20) +
dpois(9, lambda = 20) + dpois(10, lambda = 20) + dpois(11, lambda = 20) +
dpois(12, lambda = 20)
```

```
## [1] 0.03901199
```

```
#Calculamos directamente la probabilidad P(X <= 12)
ppois(12, 20)
```

```
## [1] 0.03901199
```

c) ¿Qué probabilidad hay de que en la siguiente hora lleguen más de 10 enfermos al hospital?

```
#P(X > 12) #probabilidad
1-ppois(10,lambda = 20) #
```

```
## [1] 0.9891883
```

Nota: Es importante que el valor de lambda sea consistente (debe coincidir el nombre de pacientes por hora de manera correcta).

6.5. Otras distribuciones de probabilidad

Hay otras distribuciones que no se usan de forma habitual en los ejercicios de cálculo de probabilidades, pero que tienen mucha importancia.

a) Distribución exponencial
b) Distribución de t de Student
c) Distribución ji al cuadrado
d) Distribución F de Fisher

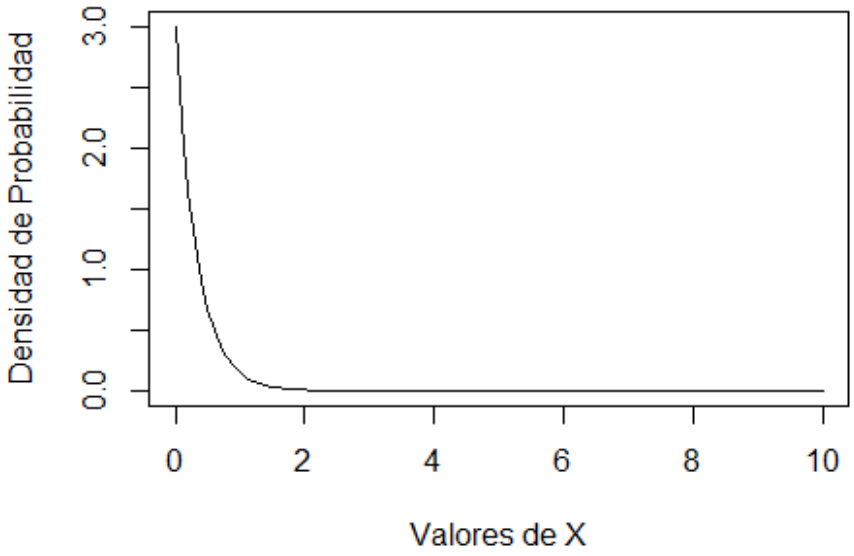
6.5.1. Distribución EXPONENCIAL

La **distribución exponencial** se puede usar para resolver problemas del ámbito de las ciencias y la ingeniería. Normalmente, se pueden usar en la teoría de colas o en problemas de falla de servicios o componentes.

Ejemplo 7:

Supongamos que el tiempo medio de atención en la primera visita a un centro sanitario es de 4 minutos. ¿Cuál es la probabilidad de que un enfermo cualquiera sea atendido en menos de 3 minutos? ¿Cuál sería la probabilidad de que se tardara entre 5 y 6 minutos?

```
pexp(3, rate = 4)

## [1] 0.9999939

# La probabilidad de demorar entre 5 y 6 minutos, inclusive
# P(X<=6)-P(X<=5)
pexp(6, rate = 3)-pexp(5, rate = 3)

## [1] 2.906723e-07

curve(dexp(x, rate = 3), xlim = c(0,10), xlab = "Valores de X", ylab =
"Densidad de Probabilidad")
```



6.5.2. Distribución t de Student

La **distribución t de Student** es una distribución de probabilidad que surge del problema de estimar la media de una población normalmente distribuida cuando el tamaño de la muestra es pequeño o muy pequeño.

Ejemplo 8:

a) Supongamos que queremos calcular el valor de la probabilidad P(T>2.20) con 10 grados de libertad:

```
pt(2.20, 10, lower.tail = F)

## [1] 0.02622053
```

b) Supongamos que queremos calcular el valor de la probabilidad P(T<2.20) con 10 grados de libertad:

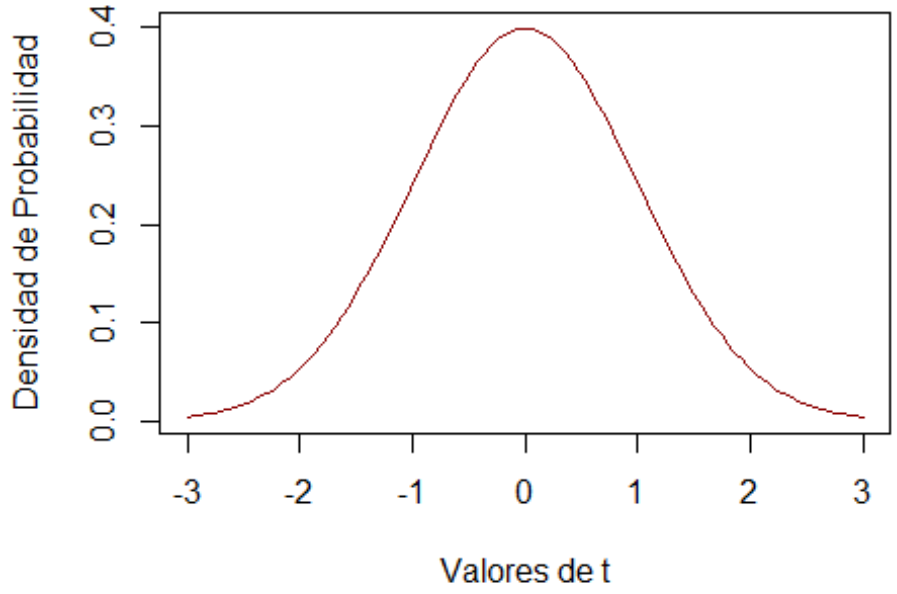
```
pt(2.20, 10, lower.tail = T) #

## [1] 0.9737795
```

c) Supongamos que queremos encontrar el percentil 2.5 y el percentil 97.5 de una distribución t de Student con 4 grados de libertad:

```
qt(c(0.025,0.975), 4)

## [1] -2.776445  2.776445
```

d) Gráfica la curva de la densidad de la distribución en general:

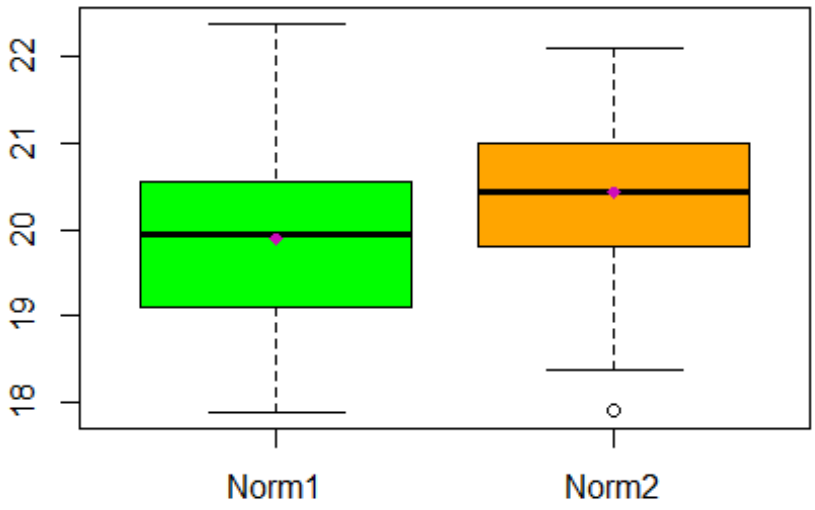
```
curve(dt(x, df = 1000), xlim = c(-3,3), xlab = "Valores de t", ylab =
"Densidad de Probabilidad", col="darkred")
```



También podemos usar esta distribución para comparar dos medias de otras distribuciones normales, por ejemplo:

Ejemplo 9:

```
#generamos 2 distribuciones normales
set.seed(999)
N1 = rnorm(100,20) # Variable aleatoria de media 10
N2 = rnorm(100,20.5) # Variable aleatoria de media 10.5
test = t.test(N1,N2) # Prueba t de Student
test

##
##  Welch Two Sample t-test
##
## data:  N1 and N2
## t = -4.1444, df = 197.47, p-value = 5.053e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.7955250 -0.2825396
## sample estimates:
## mean of x mean of y
##  19.89281  20.43184

#generamos el boxplot para visualizar más claramente
boxplot(N1,N2,names=c("Norm1","Norm2"), col=c("green", "orange"))
medias = c(mean(N1),mean(N2))
points(medias,pch=18,col=(110))
```



6.5.3. Distribución Ji cuadrado o Chi-cuadrado

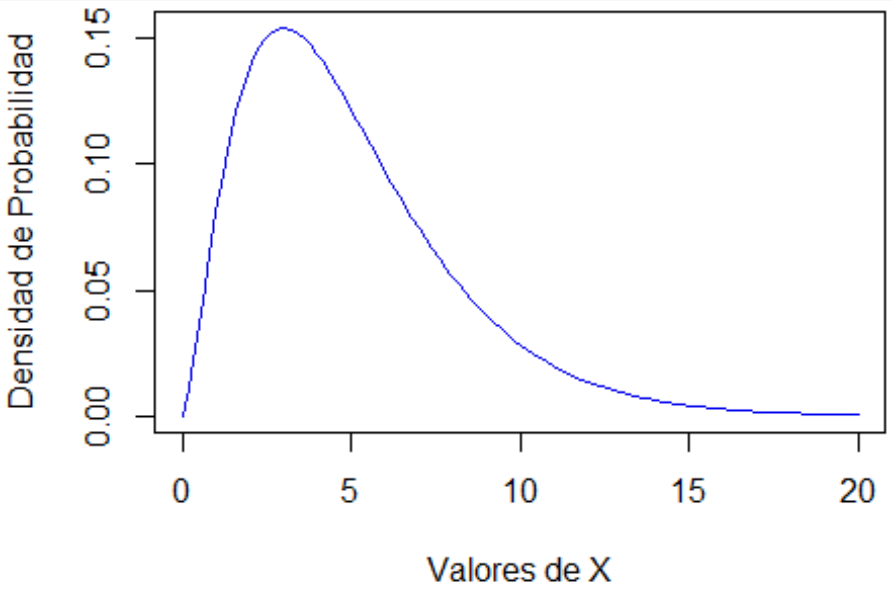
Esta distribución, también llamada *distribución de Pearson* o *distribución chi al cuadrado*, por la misma razón es una distribución continua que se especifica por los grados de libertad y el parámetro de no centralidad

El estadístico ji al cuadrado (o chi al cuadrado), o también llamado test de bondad de ajuste, que tiene distribución de probabilidad del mismo nombre, sirve para someter a prueba hipótesis referidas a distribuciones de frecuencias.

Ejemplo 10:

a) Buscar la distribución ji al cuadrado con 5 grados de libertad y percentil 90 y la gráfica correspondiente.

```
qchisq(0.90, df = 5)

## [1] 9.236357

curve(dchisq(x, df = 5), xlim = c(0,20), xlab = "Valores de X", ylab =
"Densidad de Probabilidad",col="blue2")
```



b) Tenemos los datos de un experimento almacenados en una matriz y realizamos el test para comprobar si las variables tienen asociación estadística o no a partir de los datos.

```
library(stats)
B<-matrix(c(20,47,36,25,31,19),nrow=2)
B

##      [,1] [,2] [,3]
## [1,]   20   36   31
## [2,]   47   25   19

chisq.test(B)

##
##  Pearson's Chi-squared test
##
## data:  B
## X-squared = 15.662, df = 2, p-value = 0.0003972
```

Como el valor que nos da (p-value=0.0003972) es menor al valor 0.05, podemos concluir que sí que existe una relación entre las variables.

6.5.4. Distribución F de Fisher

La distribución F es una distribución continua de muestreo de la relación de dos variables aleatorias independientes con distribuciones de ji al cuadrado, cada una dividida entre sus grados de libertad. Vamos a ver un ejemplo de cómo la podemos usar:

Ejemplo 11:

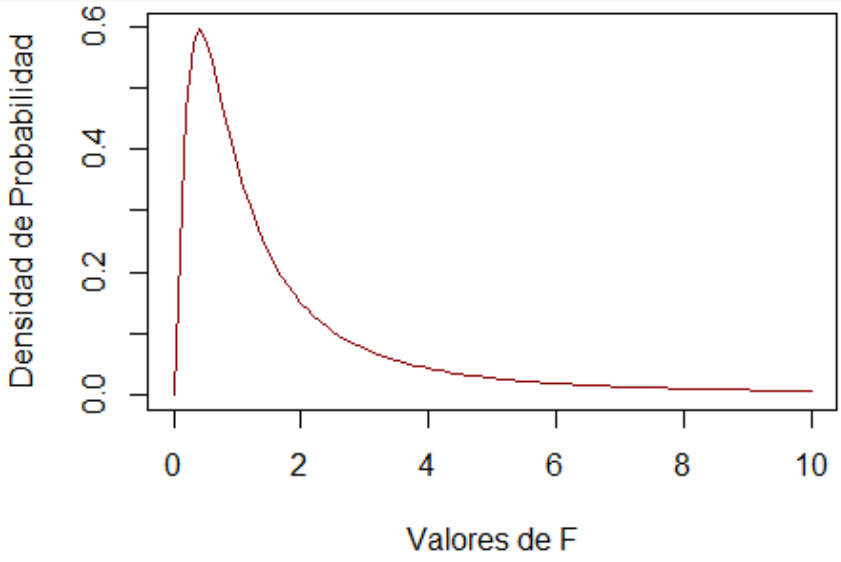
a) Buscar el percentil **9** de una **distribución F de Fisher** con **6** grados de libertad en el numerador y **6** grados de libertad en el denominador.

```
qf(0.90, df1 = 6, df2 = 3)
```

```
## [1] 5.284732
```

b) Gráfica de la densidad de la distribución del apartado anterior.

```
curve(df(x, df1 = 6, df2 = 3), xlim = c(0,10), xlab = "Valores de F", ylab =
"Densidad de Probabilidad", col="darkred")
```



7. Simulación con R

La simulación es una de las herramientas computacionales más útiles. Simular los resultados de un experimento antes de llevarlo a cabo permite a menudo detectar problemas de diseño del experimento en fases preliminares de su desarrollo.

Para hacer simulaciones podemos usar:

a) Función *sample*()
b) Función *replicate*()

c) Combinatoria
d) Funciones *rbinom*(), *runif*(),*rnorm*()
e) Las matrices en simulación

7.1. Función *sample*()

Para entender qué hacer con esta función vamos a ver un ejemplo de cómo simular experimentos con esta función:

Ejemplo 12:

Simular 150 tiradas de un dado de 6 caras balanceado de forma correcta y hay equiprobabilidad.

```
set.seed(999)
dado1 <- sample(1:6, 150, replace=TRUE)
dado1
```

```
##   [1] 3 4 5 1 6 1 2 2 6 3 5 6 6 3 3 2 1 5 3 2 4 1 5 5 1 2 2 4 1 5 3 2 1 1
3 2 2
##  [38] 4 4 4 1 2 5 1 2 5 6 1 2 5 6 4 5 4 2 3 1 4 5 3 4 1 4 1 4 3 1 5 2 3 1
3 2 6
##  [75] 5 1 5 5 4 6 2 6 4 6 3 1 5 2 5 5 5 3 4 6 4 3 1 6 5 1 4 2 3 6 6 3 3 2
3 1 1
## [112] 4 1 1 1 4 3 2 6 4 6 3 1 5 2 1 4 3 2 1 2 1 5 6 5 3 6 1 4 4 2 4 6 3 2
5 2 4
## [149] 6 3
```

Ahora supongamos, por ejemplo, que queremos simular un dado en el que un valor es más probable que cualquiera de los otros resultados. Imaginemos que el valor 6 es 5 veces más probable que los demás.

Para realizar esta modificación, al usar *sample()*, añadiremos un argumento adicional **probs** para indicar esas probabilidades, así:

```
set.seed(999)
dado2 <- sample(1:6, 150, replace = TRUE, prob = c(1, 1, 1, 1, 1, 5))
dado2
```

```
##   [1] 6 3 6 2 5 6 4 6 6 4 6 3 6 1 6 5 6 2 6 6 1 4 3 6 2 2 3 4 2 6 6 6 3 3
3 2 2
##  [38] 5 6 6 6 2 5 6 6 6 4 6 6 6 5 4 6 2 6 3 6 5 6 2 1 5 5 1 3 6 2 2 6 6 6
6 6 6
##  [75] 4 6 4 6 6 6 6 4 3 6 6 6 6 1 4 6 4 6 6 4 3 6 5 3 6 2 6 1 6 1 1 6 4 6
6 3 4
## [112] 6 5 6 6 2 4 6 6 1 6 2 2 6 2 1 2 5 6 6 6 6 6 6 6 6 6 2 6 6 6 6 1 2
4 6 6
## [149] 2 6
```

Como se puede observar en la simulación de datos del resultado anterior, el valor **6** aparece muchas más veces en el vector **dado2** que en el vector **dado1**; lo cual es totalmente lógico, ya que en nuestro muestreo hemos forzado que la probabilidad sea mayor.

Ejemplo 13:

Otro ejemplo sería con la extracción de bolas de colores. Supongamos que tenemos una urna con **20 bolas** de dos colores. De estas, 14 bolas son rojas y 6 son azules. Asignaremos el valor **1** a las bolas azules y **0** a las bolas rojas.

a) Calcular la probabilidad de extraer una bola de cada color.
b) Simular la extracción de una bola.
c) Simular la extracción de 9 bolas con remplazo.

```
#apartado a)
prob_roja<-14/20
prob_roja

## [1] 0.7

prob_azul<-6/20
prob_azul

## [1] 0.3

#apartado b) (a partir del valor de las probabilidades anteriores)

set.seed(999)
sample(c(1,0), 1, prob=c(0.3, 0.7))

## [1] 0

#apartado c)

set.seed(333)
sample (c(1,0), 9, rep=T, prob=c(0.3, 0.7))

## [1] 0 0 1 0 0 1 0 0 0
```

7.2. Función *replicate*()

Esta función nos permite generar simulaciones con **R** y facilita su réplica. Para usar *replicate* se usa la estructura

repeticiones = replicate(n = NumeroVeces,{

 # instrucciones de R necesarias

 # para fabricar el resultado. })

Ejemplo 14:

Imaginemos que queremos replicar 8 veces la simulación generada al tirar un dado 5 veces con remplazo.

```
sample(1:6, size = 5, replace=TRUE) #generamos la simulación de las tiradas
del dado

## [1] 4 1 3 2 6

#replicamos esta partida 8 veces
numRepeticiones = 8
simulaciones = replicate(n = numRepeticiones,{
  partida = sample(1:6, size = 5, replace = TRUE)
})
simulaciones

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

7.3. Combinatoria

En muchas ocasiones necesitamos usar permutaciones y combinaciones en nuestros ejercicios de probabilidad.

Las **permutaciones** son la forma en la que pueden presentarse objetos o eventos y en la que el orden de aparición es muy importante. Las **combinaciones**, en cambio, son agrupaciones en las que el contenido importa, pero el orden de los objetos no.

Recordemos que en probabilidad dos eventos son dependientes si el estado original de la situación cambia de un evento a otro, y esto altera la probabilidad del segundo evento.

Podemos ver a continuación un ejemplo en **R**:

Ejemplo 15:

Hagamos una lista de 5 elementos escogidos de 3 en 3, en la que el orden de la extracción importa y en la que el orden no importa.

```
#necesitamos la librería gtools
if(!require(gtools))
install.packages(gtools)

## Loading required package: gtools

permutations(5,3)
```

```
##       [,1] [,2] [,3]
##  [1,]    1    2    3
##  [2,]    1    2    4
##  [3,]    1    2    5
##  [4,]    1    3    2
##  [5,]    1    3    4
##  [6,]    1    3    5
##  [7,]    1    4    2
##  [8,]    1    4    3
##  [9,]    1    4    5
## [10,]    1    5    2
## [11,]    1    5    3
## [12,]    1    5    4
## [13,]    2    1    3
## [14,]    2    1    4
## [15,]    2    1    5
## [16,]    2    3    1
## [17,]    2    3    4
## [18,]    2    3    5
## [19,]    2    4    1
## [20,]    2    4    3
## [21,]    2    4    5
## [22,]    2    5    1
## [23,]    2    5    3
## [24,]    2    5    4
## [25,]    3    1    2
## [26,]    3    1    4
## [27,]    3    1    5
## [28,]    3    2    1
## [29,]    3    2    4
## [30,]    3    2    5
## [31,]    3    4    1
## [32,]    3    4    2
## [33,]    3    4    5
## [34,]    3    5    1
## [35,]    3    5    2
## [36,]    3    5    4
## [37,]    4    1    2
## [38,]    4    1    3
## [39,]    4    1    5
## [40,]    4    2    1
## [41,]    4    2    3
## [42,]    4    2    5
## [43,]    4    3    1
## [44,]    4    3    2
## [45,]    4    3    5
## [46,]    4    5    1
## [47,]    4    5    2
## [48,]    4    5    3
## [49,]    5    1    2
```

```
## [50,]    5    1    3
## [51,]    5    1    4
## [52,]    5    2    1
## [53,]    5    2    3
## [54,]    5    2    4
## [55,]    5    3    1
## [56,]    5    3    2
## [57,]    5    3    4
## [58,]    5    4    1
## [59,]    5    4    2
## [60,]    5    4    3
```

```
combinations(5,3)
```

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

Podemos observar que si el orden no importa siempre tendremos listas menos largas.

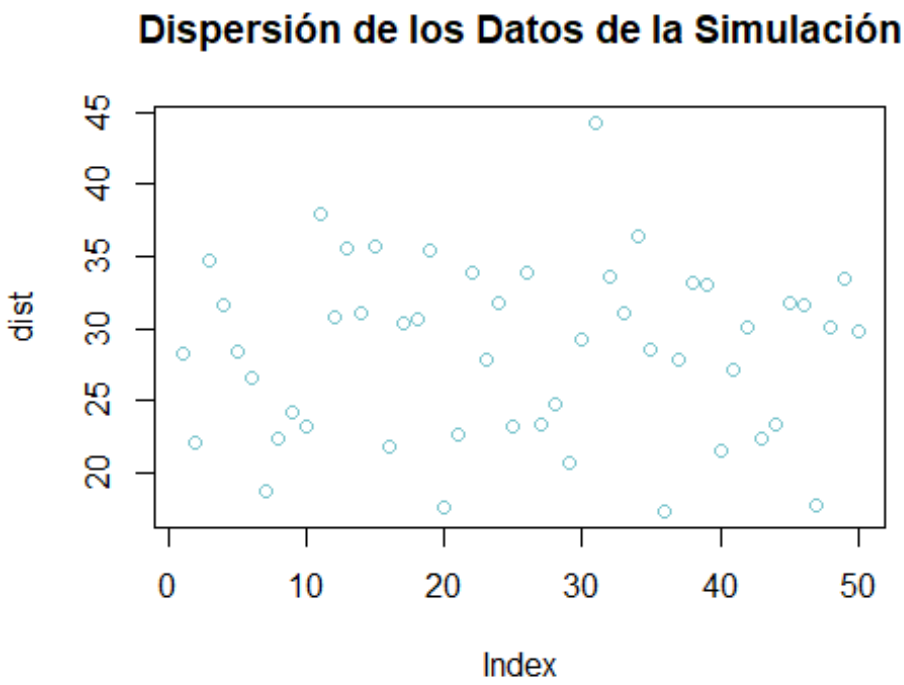
7.4. Funciones *rbinom*(), *runif*() y *rnorm*()

Ya hemos usado estas funciones para generar números y pequeñas simulaciones. Vamos a ver un típico ejemplo de estadística para validar una afirmación teórica.

Ejemplo 16:

Generar un conjunto de 50 datos aleatorios con distribución normal, media = 30, varianza = 36 y crear un gráfico.

```
set.seed(999)
dist <-rnorm(50, mean=30, sd= sqrt(36) )
plot(dist, col="cadetblue3", main="Dispersión de los Datos de la Simulación")
```



Ejemplo 17:

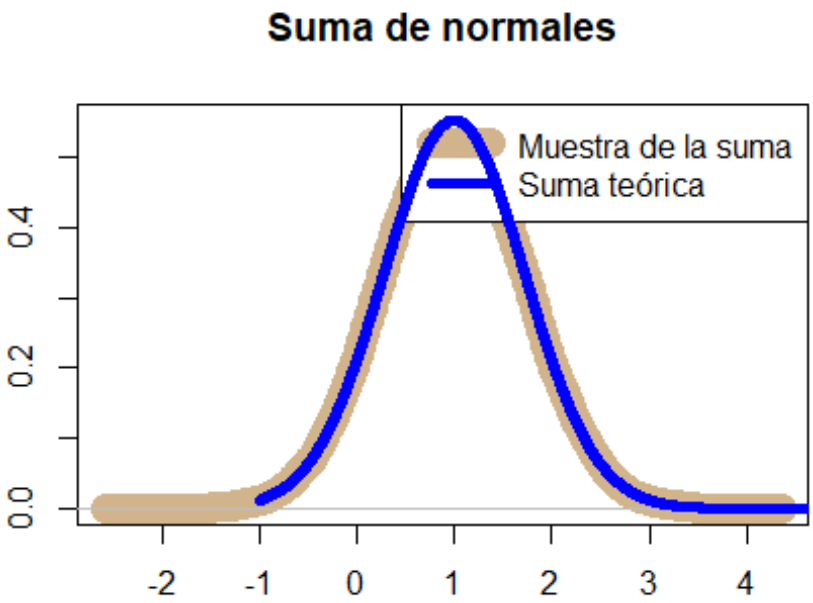
Se ha visto en teoría que si **X1~N(μ1,σ1)** y **X2~N(μ2,σ2)** son variables normales independientes, su suma es de nuevo una variable con distribución normal; en la que la media es la suma de las medias y la desviación es la raíz de la suma de las dos desviaciones al cuadrado.

$ N(μ1+μ2,sqrt(σ2^{2}+σ1^{2})

Vamos a ver si es verdad usando la función *rnorm()*:

```
n = 100000 #tamaño

mu1 = 4
sigma1 = 0.4
muestraX1 = rnorm(n, mean = mu1, sd = sigma1) #simulación de la primera
muestra

mu2 = -3
sigma2 = 0.6
muestraX2 = rnorm(n, mean = mu2, sd = sigma2) #simulación de la segunda
muestra

suma = muestraX1 + muestraX2 #suma de las dos muestras
head(suma) #primeros valores de la muestra suma

## [1] 0.5471656 1.5693697 2.1822596 0.1131247 0.2274404 1.1126092

plot(density(suma), col="tan", main="Suma de normales", xlab="", ylab="",
lwd= 15)
```

```
legend("topright", c("Muestra de la suma","Suma teórica"), col=c("tan",
"blue"), lwd=c(15, 5))
curve(dnorm(x, mean = mu1 + mu2, sd = sqrt(sigma1^2 + sigma2^2)),
      from = -1, to = 5, add = TRUE, n, col="blue", lwd= 5)
```



Si nos fijamos en el gráfico, las líneas son casi calcadas.

7.5. Las matrices en la simulación

Como ya hemos podido ver en los **LAB** anteriores, las matrices son un tipo especial de tablas de **R** que tienen la propiedad de que todos los elementos de la tabla son del mismo tipo: todos números, todos factores, etc.

En una simulación, lo habitual es repetir un mismo experimento un cierto número de veces (en general, muchas veces). Y por eso surge la necesidad de almacenar los resultados de cada una de esas repeticiones (iteraciones). Si queremos crear una matriz se puede usar la función *matrix()* y la forma más normal es partiendo de un vector.

Ejemplo 18:

Por ejemplo, si fabricamos un vector con 64 números al azar:

```
set.seed(999)
(valores = sample(-30:30, size = 64, replace = TRUE))
```

```
##  [1]  -4 -27  30 -24  10 -17 -30  27 -21  -9   4  -8  25 -16 -26 -24   7 -
12  12
## [20] -13 -30   6 -12  27 -19 -22   0  22  24   6  26  27  27 -27  25  26 -
16   0
## [39]  24 -15  -2  12 -29 -22   2  -4  19  -7   9  -5 -27  -3 -27  10 -24
3 -18
## [58]  -6 -29 -18  23   8  -8  26
```

```
(A = matrix(valores, nrow = 8, byrow = TRUE)) #generamos la matriz 8x8
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]   -4  -27   30  -24   10  -17  -30   27
## [2,]  -21   -9    4   -8   25  -16  -26  -24
## [3,]    7  -12   12  -13  -30    6  -12   27
## [4,]  -19  -22    0   22   24    6   26   27
## [5,]   27  -27   25   26  -16    0   24  -15
## [6,]   -2   12  -29  -22    2   -4   19   -7
## [7,]    9   -5  -27   -3  -27   10  -24    3
## [8,]  -18   -6  -29  -18   23    8   -8   26
```

7.6. Funciones para manejar matrices

a) La función *t()* intercambia las filas y columnas de una matriz, es decir, nos genera la matriz traspuesta.

b) La función *dim()* permite ver, pero también cambiar las dimensiones de la matriz.

c) Las funciones *colSums()* y *rowSums()* nos suman los valores por filas y columnas.

d) Las funciones *colMeans()* y *rowMeans()* nos buscan el promedio por filas y columnas.

Ejemplo 19:

```
set.seed(999)
A = matrix(1:16, nrow = 4)
A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```
t(A) #Transponemos la matriz A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
## [4,]   13   14   15   16
```

```
dim(A) #Dimensión de la matriz
```

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

```
colSums(A) #Buscamos la suma por columnas
```

```
## [1] 10 26 42 58
```

```
rowSums(A) #Buscamos la suma por filas
```

```
## [1] 28 32 36 40
```

```
colMeans(A) #Buscamos la media por columnas

## [1]  2.5  6.5 10.5 14.5

rowMeans(A) #Buscamos la media por filas

## [1]  7  8  9 10

A+5 #Sumamos 5 a los valores de A

##      [,1] [,2] [,3] [,4]
## [1,]    6   10   14   18
## [2,]    7   11   15   19
## [3,]    8   12   16   20
## [4,]    9   13   17   21

A>3 #Aplicamos una condición lógica

##       [,1] [,2] [,3] [,4]
## [1,] FALSE TRUE TRUE TRUE
## [2,] FALSE TRUE TRUE TRUE
## [3,] FALSE TRUE TRUE TRUE
## [4,]  TRUE TRUE TRUE TRUE
```

Nota: Recuerda que podrás usar cualquier función que se aplique a un vector y se aplicará elemento a elemento de tu matriz.

8. Muestreo

En estadística es muy importante el muestreo. Normalmente, nuestra matriz siempre estará constituida per una muestra de datos, pero si tuviéramos toda la población, quizás nos interesaría extraer una muestra aleatoria. En **R** es un procedimiento sencillo y se usan los números aleatorios o seudoaleatorios.

8.1 Extracción de muestras aleatorias simples

Para hacer extracción de muestras aleatorias simples en **R** podemos usar diferentes métodos:

a) Función *sample*():

En muchas ocasiones podemos seleccionar casos de una muestra con la función *sample()*.

Ejemplo 20:

Imaginemos que tenemos un dataset de crímenes y queremos una muestra de 40 casos. Para no extraer el número de un mismo individuo más de una vez, deberemos indicar *replace* = FALSE.

```
crime<-data.frame(crimtab)
dim(crime)
```

```
## [1] 924   3

n=40 #tamaño de la muestra
set.seed(999) #incluimos la semilla para que nos dé el mismo valor siempre
mimuestra <- sample(1:nrow (crime),n, replace=FALSE)
head(mimuestra)

## [1] 923 324  61 583 361 654
```

Vemos que nos da como resultado los registros de aquellos casos que queremos seleccionar para nuestra muestra.

b) Extracción con el paquete **dplyr**:

Ejemplo 21:

Con los mismos datos del ejemplo anterior, generamos otra muestra usando este paquete y el siguiente código:

```
library(dplyr)

##
## 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

mimuestra2<- crime %>%
  sample_n(size=n,replace=FALSE)
head(mimuestra2)

##   Var1   Var2 Freq
## 1   12 162.56    8
## 2  9.9 157.48    1
## 3 10.2 142.24    0
## 4   11 162.56   37
## 5 12.6  165.1    1
## 6 13.3 185.42    0
```

9. Ejercicios y casos prácticos con R

Ejercicio 1:

En un ensayo clínico se ha tomado como muestra un total de 100 pacientes. La probabilidad de que el paciente padezca una enfermedad cardiovascular es de **0.45**. Calculad:

a) La probabilidad de que exactamente 30 pacientes padezcan esta enfermedad.
b) La probabilidad de que al menos 15 pacientes tengan esta enfermedad. ¿Os parece curioso este resultado?
c) Obtened la probabilidad de que entre 5 y 10 pacientes (ambos inclusive) tengan la enfermedad.

Ejercicio 2:

a) Generad **500** observaciones de una distribución binomial, B(1, 0.5) y almacenad los datos en un vector llamado **«Bino»**. Calculad las frecuencias absolutas y relativas de los valores generados en el vector anterior. ¿Qué observáis con el resultado que habéis obtenido?
b) Si en **promedio hay 12 individuos** por minuto cruzando una calle, calculad la probabilidad de que **18 o más** individuos crucen esta calle en un minuto cualquiera.

Ejercicio 3:

Supongamos que estamos en una fábrica de fármacos para el dolor de cabeza que realiza un número específico de fármacos por minuto y que sigue una **distribución de Poisson** con parámetro λ **= 0.5.** Determinad:

a) La probabilidad de que, en un minuto al azar, se genere un único fármaco.
b) La probabilidad de que, en un minuto al azar, se generen un máximo de 3 fármacos.
c) La probabilidad de que se generen 4 fármacos en 8 minutos.

Ejercicio 4:

Disponemos de una urna con **40 bolas**. **25 son amarillas y las demás son verdes**. Asignaremos el valor 1 a las bolas amarillas y el 0 a las bolas de color verde:

a) Calculad la probabilidad de extracción de cada tipo de bola.
b) Simulación de la extracción de una bola sin remplazo.
c) Simulación de 9 extracciones con remplazo.

Ejercicio 5:

Resolved los apartados siguientes:

a) Calculad la probabilidad de obtener cuatro caras al lanzar seis veces una moneda perfecta.
b) Calculad la probabilidad de obtener como mucho cuatro caras al lanzar seis veces una moneda perfecta.

c) Generad 10 valores seudoaleatorios de una distribución X~B(6,0.5).

d) Generad una muestra de **tamaño 10000** de una distribución normal **N(3,2)** y cread un histograma con las barras de color con la muestra que se os ha dado.

e) Tal como hemos visto en teoría, vamos a ver si la suma de estas dos variables normales **x1~N(6, 0.5)** y **x2~N(3, 0.6)** también es una variable normal.

Ejercicio 6:

Imaginemos un juego en el que tiramos un dado 5 veces en cada partida y queremos simular 8 partidas del juego y ver partidas ganadoras. Las partidas ganadoras serán aquellas que tengan un 6 en alguna de las tiradas del dado.

Ejercicio 7:

Tenemos un bol con 3 tipos de manzanas **(golden, fuji, gala)**.

a) ¿Cuántas permutaciones tendremos? La teoría dice que serán 3^2 permutaciones, es decir, 9 parejas.

b) ¿Cuántas combinaciones podemos tener?

Por otro lado, calculad cuántos codones hay que tengan todos sus nucleótidos distintos. Recordad que los nucleótidos son **A, T, G, C**.

Ejercicio 8:

Dada una distribución **N(μ,σ2) ~ N(20,9)**,

a) calculad los cuartiles (0.25,0.5,0.75);

b) calculad la probabilidad P[X >= 8], y c) representad la función de densidad.

Ejercicio 9:

A partir del conjunto de datos **ChickWeight**, buscad un resumen estadístico de la variable **weight**. Buscad una muestra de **30 polluelos** con el comando *sample()* de forma aleatoria llamada *Mimuestra3* y generad un nuevo conjunto de datos con solo los registros seleccionados en tu muestra.

Cread un *dataframe* nuevo, con este subconjunto de datos llamado *Mimuestra3*, y comprobad si los **30 registros** siguen una distribución de la variable **weight** parecida.

Ejercicio 10:

Resolved los siguientes y comentad los resultados obtenidos: a) Se ha realizado un estudio a partir del cual se sabe que un determinado fármaco reduce los efectos de la alergia al polen. Con los datos obtenidos hasta el momento, solo en el 4 % de los casos no ha surgido efecto.

a) ¿Qué tipo de distribución representa este caso? ¿Cuál es la probabilidad de que si se realiza la técnica 100 veces haya más de 5 casos en los que el fármaco no sea efectivo?

b) La duración media de recuperación de una gripe común es de **7 ± 2 días (media y desviación estándar)**. Suponiendo que se trata de una distribución normal, calculad la probabilidad de que la recuperación dure entre 6 y 10 días.

Caso práctico (acumulativo):

En este ejercicio, estudiaremos la distribución de unos datos unidimensionales. Los datos que usaremos son el *dataset* **Faithful** incorporado en la librería Stepfun de R. Este *dataset* nos da información sobre el tiempo de espera entre erupciones y la duración de la erupción del Géiser Old Faithful en el Parque Nacional de Yellowstone, Wyoming, Estados Unidos (https://stat.ethz.ch/R-manual/Rdevel/library/datasets/html/faithful.html).

a) Realizad un estudio estadístico descriptivo de la variable **eruptions** y generad el gráfico de tallo y hojas.

b) Generar un gráfico de tallo y hojas con estos datos.

c) Generad un histograma. Definid intervalos menores en el histograma y añadid al gráfico la función de densidad superpuesta.

d) Usando la función *ecdf* de la biblioteca estándar Stepfun, representad la función de distribución empírica de la muestra, es decir, una distribución a partir de estos datos que proporcione un cierto parecido a una distribución verdadera asociada con la población.

e) ¿Creéis que la distribución del apartado anterior corresponde a alguna distribución estándar conocida?

f) Seleccionad las erupciones que duran más de 3 minutos. ¿Qué podéis decir de estos datos?

g) Generad un gráfico con los datos del apartado f). ¿Creéis que se parecen a alguna distribución?

Solución a los ejercicios propuestos y casos prácticos con R

Solución del ejercicio 1:

```
#Usaremos una distribución binomial
#apartado a)
n=100 #tamaño de la muestra
p=0.45 #probabilidad
val1=30
dbinom(val1,n,p)
```

```
## [1] 0.0007757151
```

```
#apartado b)
val2=14
pbinom(val2,n,p,lower.tail=FALSE)
```

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

```
#apartado c)
val3=5
val2=9
pbinom(val3,n,p)-pbinom(val2,n,p)
```

```
## [1] -3.850508e-15
```

Solución del ejercicio 2:

```
#apartado a)
set.seed(999) # semilla
bino <-rbinom(500, 1, 0.5) # genera 200 observaciones con distribución
table(bino) #frecuencias absolutas
```

```
## bino
##   0   1
## 253 247
```

```
prop.table(table(bino)) #frecuencias relativas
```

```
## bino
##     0     1
## 0.506 0.494
```

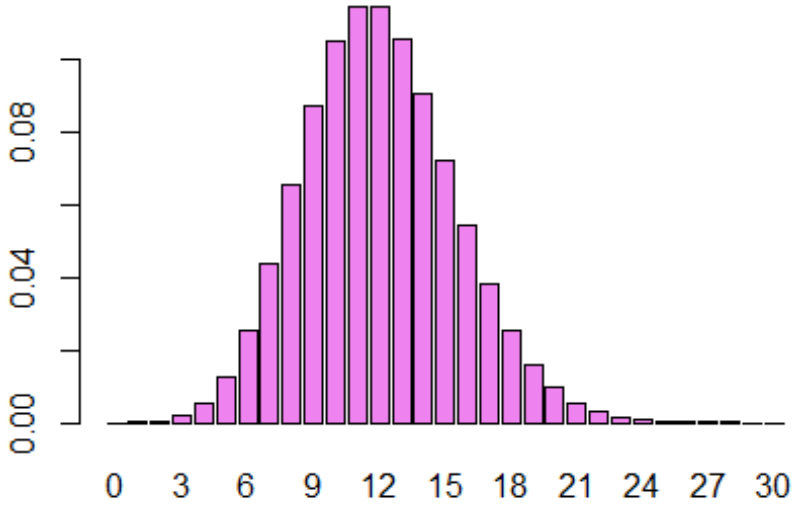
```
#apartado b)
lambda=12
ppois(17, lambda, lower.tail = FALSE) #usamos la cola derecha
```

```
## [1] 0.0629663
```

```
1-ppois(17,lambda) #usamos la cola izquierda
```

```
## [1] 0.0629663
```

```
barplot(dpois(x = 0:30, 12), names.arg = 0:30, col="violet")
```



Solución del ejercicio 3:

```
#apartado a)
dpois(c(1), 0.5)

## [1] 0.3032653

#apartado b)
ppois(c(2), 0.5) #queremos P( X ≤ 3),

## [1] 0.9856123

#apartado c)
lambda = 8/2
dpois(c(8), lambda)

## [1] 0.02977018
```

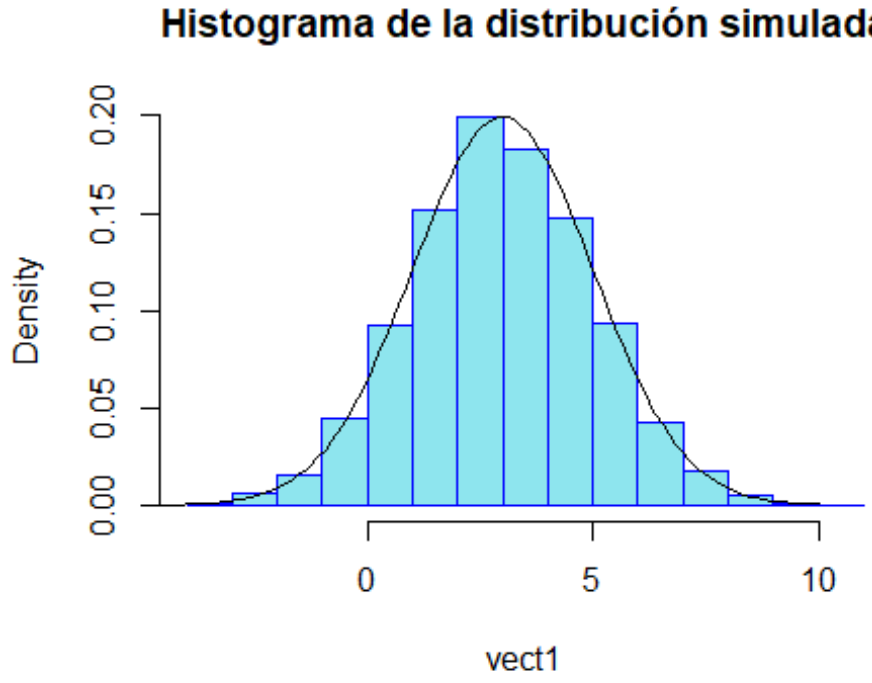
Solución del ejercicio 4:

```
#apartado a)
b_amarilla <-25/40 #probabilidad bolas azules
b_amarilla

## [1] 0.625

b_verde <-15/40 #probabilidad bolas rojas
b_verde

## [1] 0.375

b_amarilla + b_verde #comprovación

## [1] 1
```

```
#apartado b)
sample(c(1,0), 1, prob=c(0.625,0.375)) #simulamos 1 extracción de una bola sin remplazo

## [1] 1

#apartado c)
sample(c(1,0), 9, rep=T, prob=c(0.625,0.375)) #simulamos 9 extracciones con remplazo

## [1] 1 0 1 1 0 1 1 1 0
```

Solución del ejercicio 5:

```
vect1 <- rnorm(10000, 3, sd=2) #generamos la distribución normal y la almacenamos en un vector
head(vect1) #observamos las primeras observaciones

## [1] -0.9836115  4.6088225  1.1521722  1.6554406  1.2166075  0.9599479

hist(vect1, freq =FALSE, main="Histograma de la distribución simulada",
col="cadetblue2", border="blue")
curve(dnorm(x, mean = 3, sd = 2), from = -5, to = 10, add = TRUE)
```



Solución del ejercicio 6:

```
set.seed(999) #semilla de aleatorización
sample(1:6, size = 5, replace = TRUE) #generamos la partida

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

```
#usamos la función replicate para simular las 8 partidas
numRep=8
repeticiones = replicate(n = numRep,{
partida = sample(1:6, size = 5, replace = TRUE) })
repeticiones #miramos el resultado de la replicación de las partidas.
```

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

```
class(repeticiones) #vemos qué tipo de objeto es repeticiones
```

```
## [1] "matrix" "array"
```

```
dim(repeticiones)
```

```
## [1] 5 8
```

```
repeticiones==6 #con esta instrucción podemos ver qué partidas son ganadoras
```

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

Solución del ejercicio 7:

```
library(gtools)
manzanas <-c("golden" , "fuji", "gala")
#apartado a)
permutations(3,2,v=manzanas,repeats.allowed = T)
```

```
##       [,1]     [,2]
##  [1,] "fuji"   "fuji"
##  [2,] "fuji"   "gala"
##  [3,] "fuji"   "golden"
##  [4,] "gala"   "fuji"
##  [5,] "gala"   "gala"
##  [6,] "gala"   "golden"
##  [7,] "golden" "fuji"
##  [8,] "golden" "gala"
##  [9,] "golden" "golden"
```

```
#apartado b)
permutations( n = 3 , r = 2 , v=manzanas) #sin repetición
```

```
##      [,1]     [,2]
## [1,] "fuji"   "gala"
## [2,] "fuji"   "golden"
## [3,] "gala"   "fuji"
## [4,] "gala"   "golden"
## [5,] "golden" "fuji"
## [6,] "golden" "gala"
```

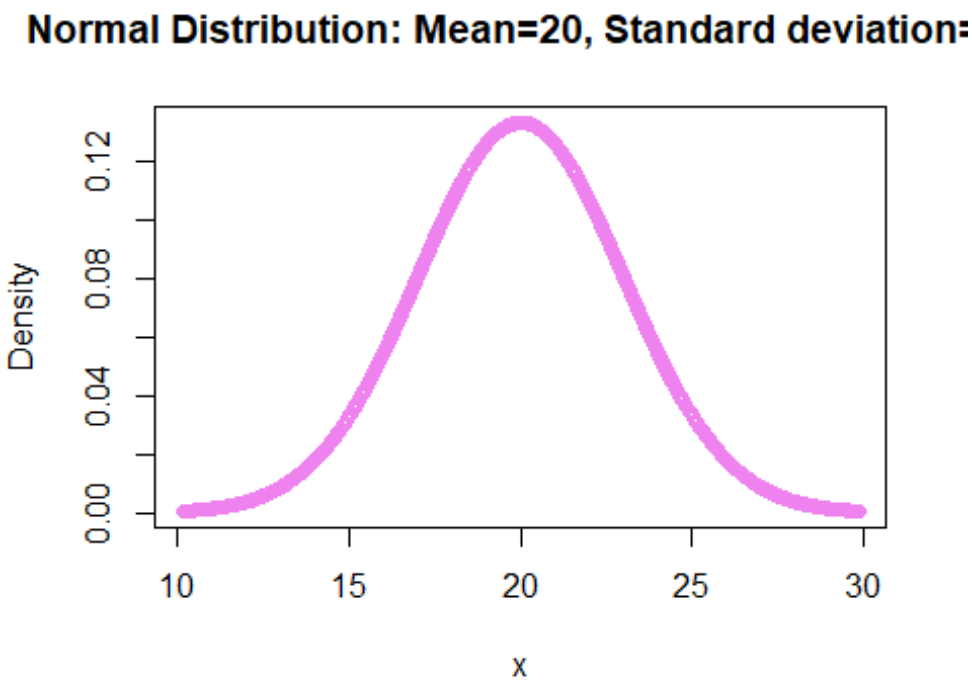
```
#apartado c)
library(gtools)
permutations(n = 4, r = 3, v = c("A", "T", "G", "C"), repeats.allowed = TRUE)
```

```
##       [,1] [,2] [,3]
##  [1,] "A"  "A"  "A"
##  [2,] "A"  "A"  "C"
##  [3,] "A"  "A"  "G"
##  [4,] "A"  "A"  "T"
##  [5,] "A"  "C"  "A"
##  [6,] "A"  "C"  "C"
##  [7,] "A"  "C"  "G"
##  [8,] "A"  "C"  "T"
##  [9,] "A"  "G"  "A"
## [10,] "A"  "G"  "C"
## [11,] "A"  "G"  "G"
## [12,] "A"  "G"  "T"
## [13,] "A"  "T"  "A"
## [14,] "A"  "T"  "C"
## [15,] "A"  "T"  "G"
## [16,] "A"  "T"  "T"
## [17,] "C"  "A"  "A"
## [18,] "C"  "A"  "C"
## [19,] "C"  "A"  "G"
## [20,] "C"  "A"  "T"
## [21,] "C"  "C"  "A"
## [22,] "C"  "C"  "C"
## [23,] "C"  "C"  "G"
## [24,] "C"  "C"  "T"
## [25,] "C"  "G"  "A"
## [26,] "C"  "G"  "C"
## [27,] "C"  "G"  "G"
## [28,] "C"  "G"  "T"
## [29,] "C"  "T"  "A"
## [30,] "C"  "T"  "C"
## [31,] "C"  "T"  "G"
## [32,] "C"  "T"  "T"
## [33,] "G"  "A"  "A"
## [34,] "G"  "A"  "C"
## [35,] "G"  "A"  "G"
## [36,] "G"  "A"  "T"
## [37,] "G"  "C"  "A"
```

```
## [38,] "G"  "C"  "C"
## [39,] "G"  "C"  "G"
## [40,] "G"  "C"  "T"
## [41,] "G"  "G"  "A"
## [42,] "G"  "G"  "C"
## [43,] "G"  "G"  "G"
## [44,] "G"  "G"  "T"
## [45,] "G"  "T"  "A"
## [46,] "G"  "T"  "C"
## [47,] "G"  "T"  "G"
## [48,] "G"  "T"  "T"
## [49,] "T"  "A"  "A"
## [50,] "T"  "A"  "C"
## [51,] "T"  "A"  "G"
## [52,] "T"  "A"  "T"
## [53,] "T"  "C"  "A"
## [54,] "T"  "C"  "C"
## [55,] "T"  "C"  "G"
## [56,] "T"  "C"  "T"
## [57,] "T"  "G"  "A"
## [58,] "T"  "G"  "C"
## [59,] "T"  "G"  "G"
## [60,] "T"  "G"  "T"
## [61,] "T"  "T"  "A"
## [62,] "T"  "T"  "C"
## [63,] "T"  "T"  "G"
## [64,] "T"  "T"  "T"
```

Solución del ejercicio 8:

```
#apartado a)
qnorm(c(0.25,0.5,0.75), mean=20, sd=3, lower.tail=FALSE) #para encontrar los
cuartiles usamos qnorm

## [1] 22.02347 20.00000 17.97653

#apartado b)

pnorm(c(8), mean=20, sd=3, lower.tail=FALSE) #para calcular la probabilidad
usaremos

## [1] 0.9999683

#apartado c)

x <- seq(10.128, 29.872, length.out=1000)
plot(x, dnorm(x, mean=20, sd=3), xlab="x", ylab="Density", main=paste("Normal
Distribution: Mean=20, Standard deviation=3 "), col="violet")
```



Solución del ejercicio 9:

```
require(datasets)
data("ChickWeight")
View(ChickWeight)
summary(ChickWeight$weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    35.0    63.0   103.0   121.8   163.8   373.0
```

```
n=30 #tamaño de la muestra
set.seed(969) #incluimos la semilla para que nos de el mismo valor siempre
mimuestra3 <- sample(1:nrow (ChickWeight),n, replace=FALSE)
mimuestra3
```

```
##  [1] 106 572 103 551  57  46 507 285 485 486 363 318 334  29 417 569  10
63 196
## [20] 533 320 194 505  17  11 133 421 300 336 180
```

```
datamuestra<-
ChickWeight[c(324,61,186,10,291,279,207,391,403,491,82,421,460,393,181,375,54
9,441,378,249,223,16,541,363,258, 9,353,411,344),1] #escogemos la primera
columna, es decir la variable weight, de los registros de nuestra muestra
summary(datamuestra) #generamos el summary del valor de nuestros registros
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    41.0    64.0   135.0   137.1   184.0   321.0
```

Solución del ejercicio 10:

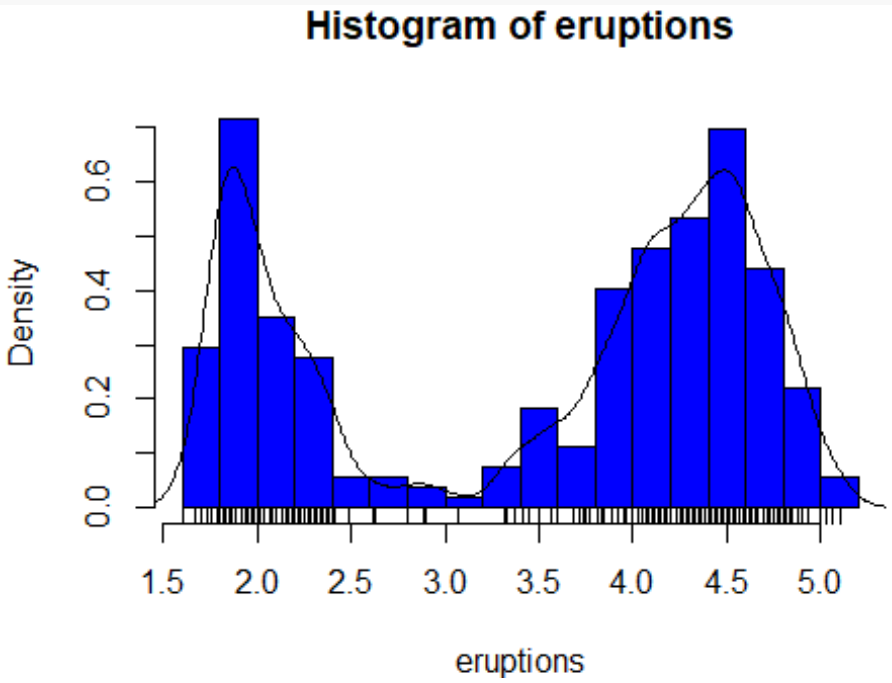
```
# apartado a)

#Se trata de una distribución binomial, ya que contamos el número de éxitos
(presentar rechazo) en una secuencia #de ensayos (número de veces que se
aplica la técnica) de Bernoulli (2 posibles estados: sin/con rechazo)
#independientes entre sí.

1-pbinom(c(5),100,0.04)

## [1] 0.2116251

#apartado b)

#restamos las 2 distribuciones normales
pnorm(10,7,2)-pnorm(6,7,2)

## [1] 0.6246553
```

Solución caso práctico 10:

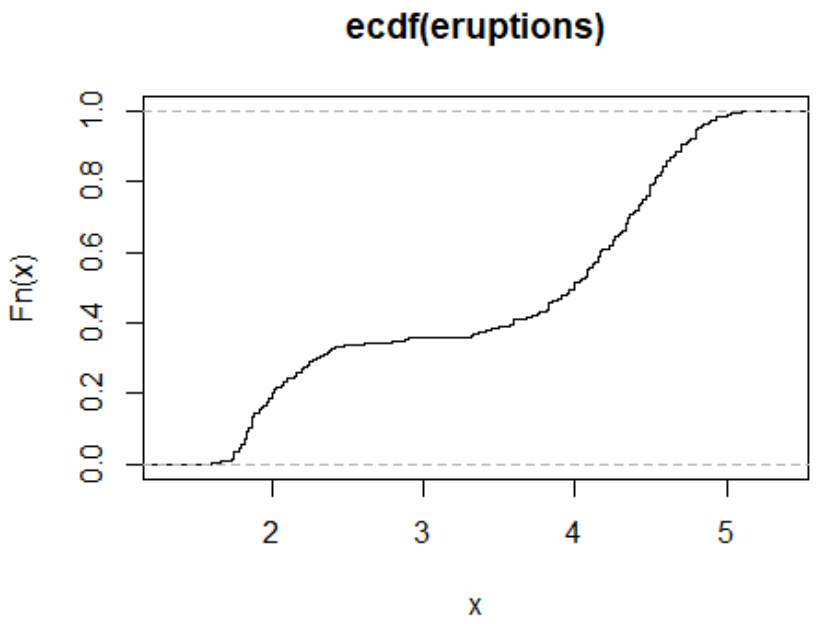
```
#apartado a)

data("faithful")
attach(faithful)
summary(eruptions)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.600   2.163   4.000   3.488   4.454   5.100

fivenum(eruptions)

## [1] 1.6000 2.1585 4.0000 4.4585 5.1000

#apartado b)
stem(eruptions, scale=1,width=80) #gráfico de tallo y hojas

##
##   The decimal point is 1 digit(s) to the left of the |
##
##   16 | 070355555588
##   18 | 000022233333335577777777888822335777888
##   20 | 00002223378800035778
##   22 | 0002335578023578
##   24 | 00228
##   26 | 23
##   28 | 080
##   30 | 7
##   32 | 2337
##   34 | 250077
##   36 | 0000823577
```

```
##    38 | 2333335582225577
##    40 | 000000335778888800223355557778
##    42 | 03335555778800233333555577778
##    44 | 022223355577800000000233333577788888
##    46 | 00002333577000000023578
##    48 | 00000022335800333
##    50 | 0370
```

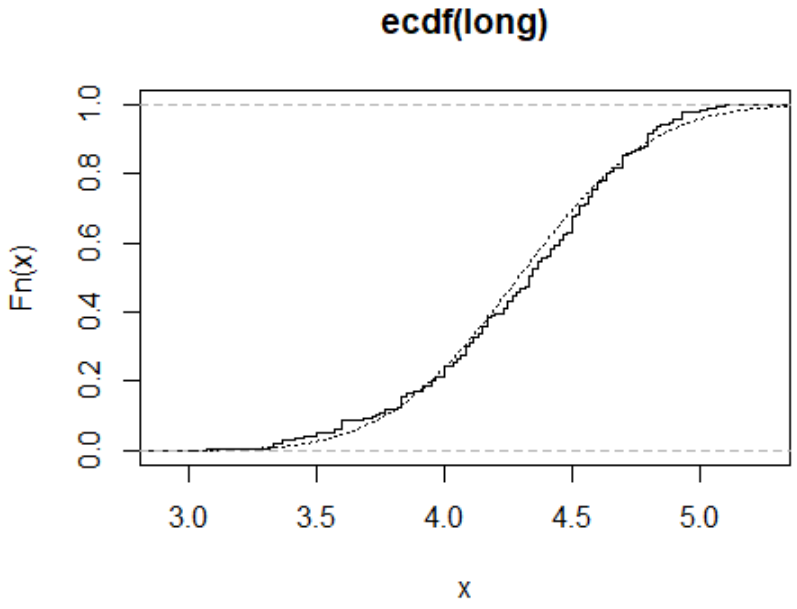
```
#apartado c)
hist(eruptions, seq (1.6, 5.2, 0.2), prob= TRUE, col="blue")
lines(density(eruptions, bw=0.1))
rug(eruptions)
```




```
#apartado d)
plot(ecdf(eruptions), do.points=FALSE, verticals=TRUE)
```




```
#apartado e) No se parece a una distribución estándar conocida. Podemos decir
que es una distribución acumulada y quizás una distribución bimodal.

#apartado f)
long<-eruptions [eruptions >3] ## Seleccionamos los datos con la condición
requerida:

# apartado g)
plot(ecdf(long), do.points=FALSE, verticals=TRUE)
x<-seq(3, 5.4, 0.01)
lines(x, pnorm(x, mean=mean(long), sd=sqrt(var(long))), lty=3)
```




```
#Ahora sí que podemos pensar que se parece a una distribución exponencial.
```

Información adicional

Algunos de los recursos materiales utilizados y recomendados para trabajar este **LAB4** son los siguientes:

Página principal de R Markdown

Guía de R Markdown

R Markdown: The Definitive Guide

Trucos de R Markdown en RStudio

R Data for Science

The R Book