```
####### CLASE 1: PRINCIPIOS BÁSICOS #########


#### PARTE A: TIPOS SIMPLES.

# Ejemplos de asignación de variables.
num <- 6
nombre <- "José"

# Consultar todos los objetos cargados en el entorno.
objects()

# Otras maneras para efectuar asignaciones.
nombre2 = "Agustín" # OJO! No está soportada en todos los entornos.

# Operaciones.
num + 5           # Salida directa en pantalla.



#### PARTE B: VECTORES.

x <- c(2,8,10,12)
x[2]             # Los arreglos se indexan desde el 1.

nombres <- c("José", "Pedro", "Agustín")
nombres

y <- 1:4         # Otra forma de generar vectores.w
y

z <- x + y
z                # Podemos operar directamente sobre vectores.

## Comando seq

w <- seq(1,4)     # Equivalente a 1:4

r <- seq(1,4,0.5) # Especificamos un 'step'.
r

r <- seq(1,-3,-0.5) # Podemor también ir para atrás.
r                  # Con un paso de 0.5 positivo obtendríamos un
error.

seq(-5,5, length=10) # Nos armamos una tira de 10 elementos.
# Notar la manera especial con la cual podemos pasar los parámetros a
las funciones.
# Además, si no se asigna el resultado a una variable sale lo obtenido
por pantalla.

## Comando rep.
```

```
x <- rep(0,10)
x

rep(1:3,4)      # Repetimos varias veces una lista.
rep(c("José", "Agustín"), 4)

## Comandos sobre vectores.
sum(c(2,5,6))

## Vectores lógicos.
x <- c(2,4,7,20)
y <- c(3,2,8,19)
z <- x>y
z     # z pasa a tener un vector con valores lógicos.

sum(z)  # En R, el TRUE se puede pensar como 1 y el FALSE como 0.


#### PARTE C: COMANDO SAMPLE.


# ¿Qué hace?
sample(1:5, size=5, replace=T)
# Notar la otra forma de pasar parámetros a una función.

sample(c("José", "Agustín", "Pedro"), size=2, replace=T)

# Esto fue el muestreo con reposición.

sample(1:5, size=3, replace=F)

# ¿Cómo hago para mezclar("shuffle") un vector?
sample(1:5, size=5, replace=F)

# ¿Y para un vector genérico?
x <- c("Buenos Aires", "Mar del Plata", "Rosario", "Córdoba")
x

sample(x, size=length(x), replace=F)



#### PARTE D: FUNCIONES.

# El valor de retorno de la última expresión será lo que devuelva la
función.
sumar = function(x1,x2)
{
  suma = x1 + x2
  suma
}

sumar(10,3)
sumar(c(2,4), c(10,2))
```

```
# Ahora puedo sumar cualquier cosa, los tipos se resuelven en tiempo
de ejecución.



# Vamos a querer armar una función que nos permita mezclar un vector.
# (Nota: esto conviene tenerlo definido en un script).

shuffle <- function(v)
{
  sample(v, size=length(v), replace=F)
}

shuffle(1:20)

# Ejemplo en donde se use un ciclo.
sumarNumeros = function(n)
{
  suma = 0
  for (i in 1:n)   # Se puede iterar sobre una colección.
  {
    suma = suma + i
  }
  suma
}

sumarNumeros(10)

# Vamos a implementar la función "sum" de R nuevamente.
sumarTodo = function(v)  # Esperamos un vector de números.
{
  suma = 0
  for (elem in v)  # Fijarse como iteramos sobre toda la colección.
  {
    suma = suma + elem
  }
  suma
}


sumarTodo(c(2,4,10))

# Metamos ahora el if. Sumaremos únicamente aquellos valores que sean
positivos.
sumarPositivos = function(v)
{
  suma = 0
  for (elem in v)
  {
    if (elem > 0)
    {
      suma = suma + elem
    }
  }
```

```
  suma
}

sumarPositivos(c(2,-4,10,-50))



#### PARTE E: PLOTS.

## Los plots serán en esencia "scatter-plots" de y vs. x.


x <- seq(-3, 3, length=10)
y <- exp(x)
plot(x,y) # Forma básica para plotear.

# Distintos estilos.
plot(x,y, type='l') # Con líneas.
plot(x,y, type='b') # Con líneas y puntos.

title("Gráfixo exp(x)")

# Título más sofisticado.
plot(x,y, type='b') # Con líneas y puntos.
title(main="Gráfico", sub="Exponencial", xlab="x", ylab="exp(x)")

# Estos comandos pueden ir embebidos en la llamada al plot.
plot(x,y, type="l", main="Gráfico", sub="Exponencial", xlab="x",
ylab="exp(x)")


#### PARTE F: DISTRIBUCIONES DISCRETAS.

# Supongamos X de distribución Bi(n,p).
# Calcularemos P(X=k) con el comando dbinom(k,n,p)
# Ejemplo:
dbinom(3,8,0.2)
# También podemos hacer que actúe sobre un vector.
dbinom(1:3,8,0.2)


# Para calcular la distribución acumulada, tenemos dos maneras.
# Forma 1: sumando probabilidades puntuales hasta el k.
# Ejemplo, calcularemos P(X<= 3).
sum(dbinom(0:3,8, 0.2))

# R también proveé una función más directa.
pbinom(3,8,0.2)

# dnbinom y pnbinom sirven para la exponencial negativa.
# dpois y ppois sirven para la distribución de Poisson.
# ¿Y para la geométrica? RTA: es un caso particular de la binomial
negativa.
```

```
# Experimento: Urna con 8 bolitas rojas y 15 azules.
# Se extraen 6 bolitas con reposición, queremos calcular la
probabilidad
# de que hayan al menos 3 bolitas rojas.

# Forma 1, "la fácil".
# P(X >= 3) = 1 - P(X<=2)
1 - pbinom(2,6, 8/23)

# Forma 2, "la estimativa".
# Repetiremos el experimento una buena cantidad de veces.
# Atención a toda la sintaxis especial que se despliega...

n <- 10000
exitos <- 0
for (i in 1:n)
{
  urna <- rep(c("roja","azul"), c(8,15))  # Prueben ejecutar esta
línea por separado.
  muestra <- sample(urna, 6, replace=T)
  enRojo <- sum(muestra == "roja")  # Recordar lo visto antes de
vectores lógicos
                                          # Se podía haber
implementado con un ciclo esto (más costoso).
  if (enRojo >= 3)
  {
    exitos <- exitos + 1
  }
}

exitos/n  # El estimado. Ir jugando con el n, ¿qué pasa cuando es
grande?


#### PARTE G: MÚLTIPLES PLOTEOS.

# Haremos cuatro gráficas en una misma página.
# Será una binomial n=10 y con un p que varía entre 0.20, 0.40, 0.60,
0.80.

# IMPORTANTE: Estos cambios son PERMANENTES durante toda la sesión. Se
recomienda guardar
# los parámetros en una variable y después, cuando se finalizó volver
a los valores
# originales.

oldpar <- par(mfrow=c(2,2))
x <- 0:10
y <- dbinom(x, 10, 0.20)
plot(x,y, type="b", main="p=0.20")
y <- dbinom(x, 10, 0.40)
plot(x,y, type="b", main="p=0.40")
y <- dbinom(x, 10, 0.60)
plot(x,y, type="b", main="p=0.60")
```

```
y <- dbinom(x, 10, 0.80)
plot(x,y, type="b", main="p=0.80")

par(oldpar)

plot(x,y)



# PENDIENTE: cuatro gráficas en una misma tabla.
```