Inferencia Estadística

Profesor(es): Jarnishs Beltran
Ayudante: Pablo Rivera

Pauta Ayudantía N°3
Otoño 2020

Ejercicios

i) Haga lo siguiente en R:

a) Crea un intervalo de datos que esté comprendido entre 1 y 30 (ambos números incluidos), con diez números aleatorios que salgan del intervalo anterior y que además, ni uno de éstos se repita.

b) Obtener 4 números aleatorios con repetición en el intervalo 1 - 10.

c) Obtener 10 números aleatorios sin repetición en el intervalo 50 - 100.

d) Ahora, obtener con 5 números decimales generados aleatoriamente, con un mínimo de valor 3 y máximo de 4.

Respuesta:

```
sample(1:30,10,replace=F)
[1] 5 3 19 10 28 4 11 23 16 22
```

```
sample(1:10,4,replace=T)
[1] 5 5 4 1
```

```
sample(50:100,10,replace=F)
[1] 76 88 67 78 95 66 94 62 64 58
```

```
runif(5, min=3, max=4)
[1] 3.537344 3.629892 3.362016 3.860888 3.930647
```

```
runif(5, 3, 4)
```

ii) Sea (X,Y) un vector aleatorio con distribución uniforme en el cuadrado [-1,1] x [-1,1] de área 4.

a) Aproximar mediante simulación $P(X+Y \leq 0)$ y compararla con la probabilidad teórica (obtenida aplicando la regla de Laplace $\frac{área favorable}{área posible}$.

Probabilidad teórica 1/2 (area favorable / área total),

Nota: R maneja internamente los valores lógicos como 1 (TRUE) y 0 (FALSE).

```r
set.seed(1)
n <- 10000
x <- runif(n, -1, 1)
y <- runif(n, -1, 1)
indice <- (x+y < 0)
# Aproximación por simulación
sum(indice)/n
```

```
## [1] 0.4996
```

```r
mean(indice) # Alternativa
```

```
## [1] 0.4996
```

b) Aproximar el valor de π mediante simulación a partir de $P(X^2 + Y^2 \leq 1)$.

```r
set.seed(1)
n <- 10000
x <- runif(n, -1, 1)
y <- runif(n, -1, 1)
indice <- (x^2+y^2 < 1)
mean(indice)
```

```
## [1] 0.7806
```

```r
pi/4
```

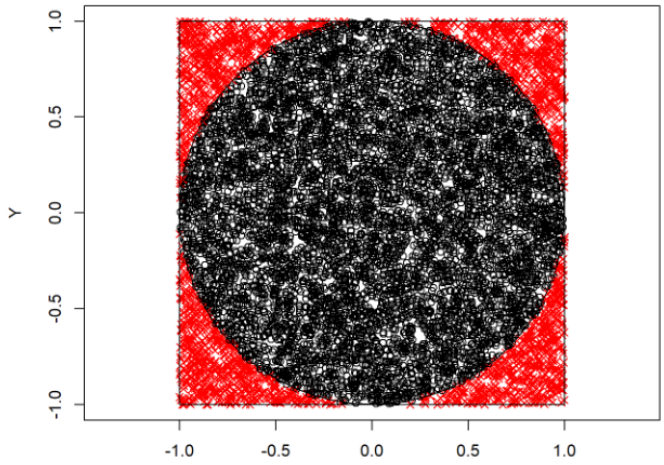
```
## [1] 0.7853982
```

```r
pi_aprox <- 4*mean(indice)
pi_aprox
```

```
## [1] 3.1224
```

c) gráfique.

```r
# Colores y símbolos depediendo de si el índice correspondiente es verdadero:
color <- ifelse(indice, "black", "red")
simbolo <- ifelse(indice, 1, 4)
plot(x, y, pch = simbolo, col = color,
     xlim = c(-1, 1), ylim = c(-1, 1), xlab="X", ylab="Y", asp = 1)
     # asp = 1 para dibujar circulo
symbols(0, 0, circles = 1, inches = FALSE, add = TRUE)
symbols(0, 0, squares = 2, inches = FALSE, add = TRUE)
```



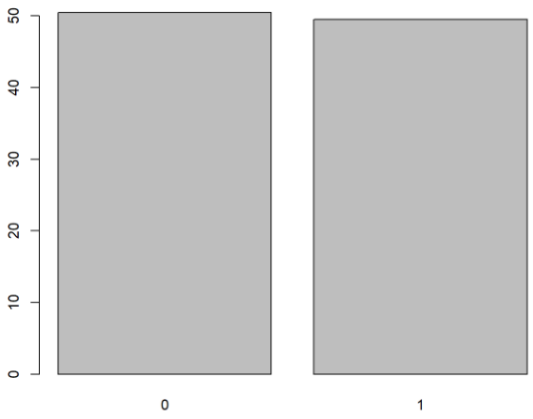
iii) Consideramos el experimento de Bernoulli consistente en el lanzamiento de una moneda.

a) Empleando la función **sample**, obtener 1000 simulaciones del lanzamiento de una moneda (0 = cruz, 1 = cara), suponiendo que no está trucada. Aproximar la probabilidad de cara a partir de las simulaciones. Representar porcentajes en gráfico.

```
set.seed(1)
nsim <- 10000
x <- sample(c(cara = 1, cruz = 0), nsim, replace = TRUE, prob = c(0.5,0.5))
mean(x)
```

```
## [1] 0.4953
```

```
barplot(100*table(x)/nsim) # Representar porcentajes
```

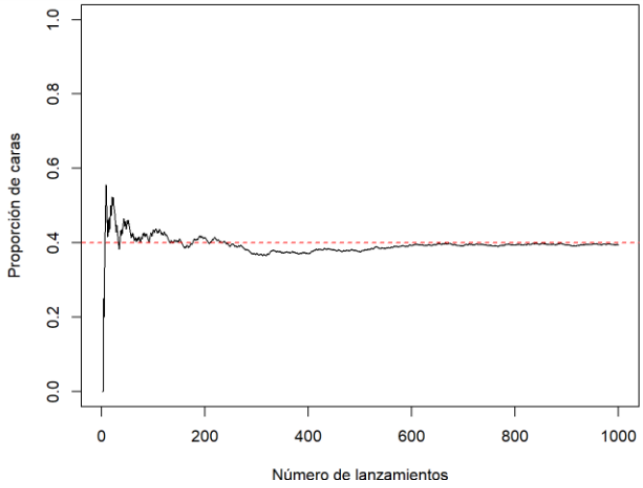


b) En R pueden generarse valores de la distribución de Bernoulli mediante la función **rbinom(nsim, size=1, prob)**. Generar un gráfico de lineas considerando en el eje X el número de lanzamientos (de 1 a 10000) y en el eje Y la frecuencia relativa del suceso cara (puede ser recomendable emplear la función **cumsum**).

```r
set.seed(1)
nsim <- 1000
p <- 0.4
x <- rbinom(nsim, size = 1, prob = p) # Simulamos una Bernouilli
n <- 1:nsim
# Alternativa programación: x <- runif(nsim) < p
mean(x)
```

```
## [1] 0.394
```

```r
plot(n, cumsum(x)/n, type="l", ylab="Proporción de caras",
     xlab="Número de lanzamientos", ylim=c(0,1))
abline(h=p, lty=2, col="red")
```



Herramientas en el paquete base de R

- **set.seed(entero)** : permite establecer la semilla (y el generador).
- **RNGkind()**: selecciona el generador.
- **rdistribución(n,...)**: genera valores aleatorios de la correspondiente distribución. Por ejemplo: runif(n, min = 0, max = 1), generaría n valores de una uniforme.
- **sample()**: genera muestras aleatorias de variables discretas y permutaciones
- **RUNIF()**: ¿Qué pasa si queremos obtener también números con decimales?
- **simulate()**: genera realizaciones de la respuesta de un modelo ajustado.

La semilla se almacena (en **globalenv**) en **.Random.seed**; es un vector de enteros cuya dimensión depende del tipo de generador:

- No debe ser modificado manualmente; se guarda con el entorno de trabajo.
- Si no se especifica con **set.seed** (o no existe) se genera a partir del reloj del sistema.

  Nota: Puede ser recomendable (para depurar) almacenarla antes de generar simulaciones, por ejemplo: **semilla** $\leftarrow$ **.Random.seed**.