1. Introducción

El siguiente documento fue compilado usando **knitr** un paquete diseñado para la generación *dinámica* de reportes para R y *Python*. Esto se hace con el propósito de tener un documento consistente y replicable. Normalmente las entregas tienen un pantallazo del código y de la salida, sin embargo, esto además de verse muy *feo* no tiene alguna manera sencilla de hacer la replicación de la simulación. Por eso mismo, en esta entrega se utiliza este paquete y facilitar manipulación, lectura del código, además de inmediatamente mostrar la salida de los códigos.

En el archivo comprimido `.7z` se encuentran varios archivos; `notas.tex`, `notas.Rnw`, `notas.pdf`, `referencias.bib`, `run.R`, `body_fat.csv`, `loans_income.csv`, y un directorio `img/`. Aquí interesan dos archivos; `notas.Rnw` se encuentra todo el código fuente del documento (LaTeXy R), en `run.R` se encuentra el *script* para poder compilar el documento. Si se desea hacer uso del *script* se necesitan unas librerías, paquetes y programas ya instalados en su máquina local, estos son; LaTeX, se recomienda hacer la instalación local dependiendo de su sistema operativo; R, lenguaje de programación para estadística; **knitr**, para poder compilar el `.Rnw` y que genere un PDF. Con estas dependencias ya instaladas, tiene que ejecutar el archivo `run.R` (en sistemas UNIX como GNU/Linux, MacOSX, FreeBSD, etc. Se puede correr como cualquier *script* de *bash*, si cuenta con otro sistema operativo tiene que ejecutar el *script* dentro de un entorno para correr código en R). Esto va a generar un PDF y de ahí puede ejecutar cuantas veces desee el *script* y ver cómo cambian los valores del PDF dinámicamente. Si no desea hacer todo ese proceso de compilación del `.Rnw` puede ejecutar directamente el PDF que viene en el comprimido, siendo este la última salida de la compilación de los autores.

2. Stopping generating new simulation data

Write a program to generate standard normal random variables until you have generated n of them, where $n \geq 100$ is such that $S/\sqrt{n} < 0.01$, where S is the sample standard deviation of the n data values. Note that this is the "Method for Determining When to Stop Generating New Data". Also, answer the following questions:

Debido a que se va a generar variables aleatorias **normales estándar** X_i, es decir, $X_i \sim \mathcal{N}(0,1)$ y se debe parar de generar cuando $\frac{S}{\sqrt{n}} < 0.01$.

```r
x <- numeric()
s <- 1
n <- 0

while(n < 100 || s / sqrt(n) >= 0.01) {
  x <- c(x, rnorm(1))
  n <- length(x)
  s <- sd(x)
}
```

```r
cat("Cantidad de Xi generados", n, "\n")

## Cantidad de Xi generados 9987

cat("Desviación estándar", round(s, 5), "\n")

## Desviación estándar 0.99928

cat("Error estándar S/sqrt(n)", round(s/sqrt(n), 5), "\n")

## Error estándar S/sqrt(n) 0.01
```

En el código se declara una variable s inicializada en 1 para que luego sea nuevamente computada a la desviación estándar de x que es un vector que va a almacenar todas las variables normales aleatorias generadas. Según el enunciado tenemos una condición de que al menos deben haber 100 variables aleatorias generadas ($n \geq 100$) y que el error estándar sea menor a 0.01 ($S/\sqrt{n} < 0.01$). En el ciclo `while`, por lo tanto, tiene sentido que el ciclo continue si alguna de las afirmaciones anteriores son falsas, por eso queda la condición de esa manera. Se hace uso de la función `rnorm(1)` para que genere una variable aleatoria normal con los valores por defecto (promedio 0 y desviación 1).

2.1. How many normals do you think will be generated? Give an analytic estimate.

2.1.1. Respuesta

Aunque ya se tengan los resultados de la simulación, se puede hacer un estimado analítico con la condición $S/\sqrt{n} < 0.01$ donde se puede despejar n para saber cuántas variables se necesitan para parar el criterio.

Se tiene en primer lugar la inecuación:

$$\frac{S}{\sqrt{n}} < 0.01$$

Se eleva ambas partes con menos 1:

$$\left(\frac{S}{\sqrt{n}}\right)^{-1} > 0.01^{-1}$$
$$\frac{\sqrt{n}}{S} > 100$$

Multiplicando ambas partes por S:

$$\not{S} \times \frac{\sqrt{n}}{\not{S}} > 100 \times S$$
$$\sqrt{n} > 100 \times S$$

Ahora se cancela la raíz elevando ambas partes al cuadrado:

$$(\cancel{\sqrt{n}})^{\cancel{2}} > (100 \times S)^2$$

Quedando entonces:

$$n > 10000 \times S^2$$

Para hacer el ejercicio se utilizó en R la función `rnorm(1, mean = 0, sd = 1)`, de esta manera genera únicamente 1 valor con media 0 y desviación 1, por lo tanto, podemos hacer $S = 1$ para estimar cuántos n necesitamos para que se cumpla condición y deje de generar variables aleatorias, por lo tanto:

$$n > 10000 \times (1)^2$$
$$n > 10000$$

Este valor se acerca bastante al que se imprime en la simulación.

2.2. How many normals did you generate?

2.2.1. Respuesta

Se han generado `n=` 9987 normales.

2.3. What is the sample mean of all the normals generated?

2.3.1. Respuesta

Utilizando el comando `mean()` al vector `x` se obtiene -0.0090067.

2.4. What is the sample variance?

2.4.1. Respuesta

La varianza muestral se calcula como:

$$S^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$

Esto se puede sacar con la función `var()` de R, por lo tanto, aplicando eso al vector `x` se obtiene 0.9985517.

2.5. Comment on the results of (2.3) and (2.4). Were they surprising?

2.5.1. Respuesta

El resultado de 2.3 es la media de `x` que es -0.0090067 y el de 2.4 es la varianza muestral de `x` dando 0.9985517. Estos valores muy sorprendentes no fueron... Se espera que al simular muchas veces una distribución normal, esta nos dé los valores de la media y la desviación estándar al cuadrado. Pero sí es interesante que dado este algoritmo para detenerse después de cruzar un $S/\sqrt{n}$ nos dé valores muy cercanos a una distribución normal con $\mu = 0$ y $\sigma = 1$.

3. Gaining confidence with confidence intervals

We know that the $\mathcal{U}(-1, 1)$ r.v. has mean 0. Use a sample of size 1000 to estimate the mean and give a 95 % confidence interval (CI). Does the CI contain 0? Repeat the above a large number of times (≥ 100). What percentage of time does the CI contain 0? Write your code so that it produces output similar to the following:

```
Number of trials: 10

Sample mean  lower bound  upper bound  contains mean?

    -0.0733      -0.1888       0.0422               1
    -0.0267      -0.1335       0.0801               1
    -0.0063      -0.1143       0.1017               1
    -0.0820      -0.1869       0.0230               1
    -0.0354      -0.1478       0.0771               1
    -0.0751      -0.1863       0.0362               1
    -0.0742      -0.1923       0.0440               1
     0.0071      -0.1011       0.1153               1
     0.0772      -0.0322       0.1867               1
    -0.0243      -0.1370       0.0885               1

100 percent of CI's contained the mean
```

3.1. Respuesta

Se sabe que para tener un intervalo de confianza de 95 %, el límite inferior y superior serán, respectivamente:

$$\left(\bar{X} - 1.96\frac{S}{\sqrt{n}}, \bar{X} + 1.96\frac{S}{\sqrt{n}}\right)$$

El siguiente código permite sacar el intervalo de confianza del 95 % y también el estimado de la media.

```r
n <- 1000
x <- runif(n, -1, 1)
xmean <- mean(x)
S <- sd(x)
L <- xmean - 1.96 * S/sqrt(n)
U <- xmean + 1.96 * S/sqrt(n)
cat("El estimado es", xmean, "\n")

## El estimado es 0.006255274

cat("95% está entre (", L, ", ", U, ") \n", sep="")

## 95% está entre (-0.02947446, 0.041985)
```

Aquí se estimó la media de la distribución uniforme, dando un resultado de 0.0062553, muy cercano a 0. El intervalo de confianza está entre -0.0294745 y 0.041985, en este caso sí contiene a 0.[1] Ahora, se va a realizar el proceso unas 200 veces para conocer el porcentaje de intervalos de confianza que contienen 0.

```r
trials <- 200
n <- 1000
true_mean <- 0

results <- data.frame(
  sample_mean = numeric(trials),
  lower_bound = numeric(trials),
  upper_bound = numeric(trials),
  contains_mean = integer(trials)
)

for (i in 1:trials){
  x <- runif(n, -1, 1)
  xmean <- mean(x)
  S <- sd(x)
  L <- xmean - 1.96 * S/sqrt(n)
  U <- xmean + 1.96 * S/sqrt(n)
  contains <- as.integer(L <= true_mean & true_mean <= U)

  results[i, ] <- c(xmean, L, U, contains)
}

cat("Número de intentos:", trials, "\n")
```

[1] Aquí hay que hacer una aclaración, los autores utilizaron knitr, una herramienta para la generación de reportes dinámicos en R, entonces cada vez que se compile este archivo `.Rnw` va a salir una salida diferente y puede ser que ya el intervalo no contenga 0.

```
## Número de intentos: 200

print(results, digits= 4)

##     sample_mean lower_bound upper_bound contains_mean
## 1     0.0254950  -0.0100232   0.0610131             1
## 2    -0.0085545  -0.0443401   0.0272310             1
## 3     0.0251374  -0.0110409   0.0613157             1
## 4    -0.0098395  -0.0456662   0.0259872             1
## 5    -0.0275464  -0.0637938   0.0087010             1
## 6    -0.0158372  -0.0525266   0.0208522             1
## 7     0.0092906  -0.0260364   0.0446176             1
## 8     0.0187823  -0.0164983   0.0540629             1
## 9    -0.0332268  -0.0690083   0.0025547             1
## 10    0.0016282  -0.0350078   0.0382641             1
## 11    0.0140130  -0.0206145   0.0486405             1
## 12   -0.0368833  -0.0724265  -0.0013401             0
## 13    0.0327855  -0.0030232   0.0685943             1
## 14    0.0077752  -0.0274891   0.0430395             1
## 15   -0.0058749  -0.0414951   0.0297454             1
## 16   -0.0436421  -0.0788764  -0.0084078             0
## 17    0.0081328  -0.0278039   0.0440695             1
## 18    0.0137140  -0.0223525   0.0497804             1
## 19    0.0068106  -0.0290390   0.0426603             1
## 20    0.0250513  -0.0109206   0.0610232             1
## 21   -0.0106300  -0.0465779   0.0253178             1
## 22   -0.0187729  -0.0548239   0.0172781             1
## 23    0.0222221  -0.0138181   0.0582623             1
## 24    0.0323193  -0.0042008   0.0688393             1
## 25    0.0007964  -0.0346673   0.0362600             1
## 26   -0.0349184  -0.0714849   0.0016481             1
## 27   -0.0105215  -0.0460226   0.0249796             1
## 28   -0.0239368  -0.0596105   0.0117370             1
## 29   -0.0347011  -0.0692925  -0.0001096             0
## 30    0.0041244  -0.0315729   0.0398216             1
## 31   -0.0036135  -0.0388212   0.0315941             1
## 32    0.0278733  -0.0080383   0.0637849             1
## 33   -0.0356497  -0.0716456   0.0003463             1
## 34    0.0217460  -0.0137713   0.0572633             1
## 35    0.0317556  -0.0042010   0.0677123             1
## 36    0.0058310  -0.0299271   0.0415892             1
## 37   -0.0060516  -0.0423462   0.0302430             1
## 38   -0.0092422  -0.0449933   0.0265088             1
## 39   -0.0047129  -0.0392223   0.0297965             1
## 40    0.0102205  -0.0260182   0.0464592             1
```

```
## 41   -0.0155934  -0.0520299   0.0208431             1
## 42    0.0013527  -0.0348691   0.0375745             1
## 43    0.0492690   0.0132061   0.0853320             0
## 44   -0.0048877  -0.0405361   0.0307608             1
## 45    0.0214129  -0.0138898   0.0567155             1
## 46    0.0019436  -0.0347594   0.0386465             1
## 47   -0.0102737  -0.0463731   0.0258257             1
## 48    0.0077403  -0.0275825   0.0430630             1
## 49    0.0311965  -0.0049409   0.0673338             1
## 50    0.0340839  -0.0008779   0.0690456             1
## 51    0.0154519  -0.0200984   0.0510023             1
## 52    0.0031811  -0.0331878   0.0395501             1
## 53    0.0053748  -0.0297564   0.0405060             1
## 54    0.0025822  -0.0332426   0.0384070             1
## 55   -0.0132447  -0.0492687   0.0227793             1
## 56   -0.0084548  -0.0442845   0.0273749             1
## 57   -0.0195016  -0.0558214   0.0168183             1
## 58   -0.0324558  -0.0687982   0.0038867             1
## 59    0.0169603  -0.0180123   0.0519329             1
## 60    0.0126508  -0.0227248   0.0480265             1
## 61    0.0073433  -0.0294259   0.0441125             1
## 62   -0.0072303  -0.0429478   0.0284872             1
## 63   -0.0233878  -0.0584422   0.0116666             1
## 64    0.0127293  -0.0217453   0.0472039             1
## 65    0.0116224  -0.0239805   0.0472253             1
## 66   -0.0073965  -0.0433926   0.0285996             1
## 67    0.0233222  -0.0124854   0.0591298             1
## 68   -0.0074283  -0.0436730   0.0288165             1
## 69   -0.0009751  -0.0369045   0.0349543             1
## 70    0.0376848   0.0010809   0.0742888             0
## 71    0.0088466  -0.0267701   0.0444634             1
## 72    0.0006483  -0.0344811   0.0357778             1
## 73    0.0391946   0.0026050   0.0757841             0
## 74    0.0389297   0.0035947   0.0742647             0
## 75    0.0075703  -0.0288384   0.0439790             1
## 76   -0.0255409  -0.0622470   0.0111653             1
## 77   -0.0043501  -0.0404561   0.0317560             1
## 78    0.0104081  -0.0255287   0.0463450             1
## 79    0.0192937  -0.0168601   0.0554474             1
## 80    0.0045044  -0.0314571   0.0404659             1
## 81   -0.0261570  -0.0618136   0.0094995             1
## 82   -0.0274987  -0.0633523   0.0083548             1
## 83    0.0111382  -0.0248151   0.0470914             1
## 84   -0.0105435  -0.0468051   0.0257181             1
```

```
## 85   -0.0433863  -0.0790022  -0.0077704             0
## 86   -0.0209618  -0.0574145   0.0154909             1
## 87    0.0112753  -0.0246450   0.0471957             1
## 88    0.0064147  -0.0291562   0.0419855             1
## 89   -0.0129176  -0.0481816   0.0223463             1
## 90   -0.0102663  -0.0459456   0.0254130             1
## 91   -0.0122325  -0.0478888   0.0234238             1
## 92   -0.0045260  -0.0401793   0.0311272             1
## 93   -0.0051586  -0.0413091   0.0309919             1
## 94    0.0087217  -0.0274986   0.0449420             1
## 95   -0.0154927  -0.0517511   0.0207656             1
## 96    0.0125142  -0.0233766   0.0484050             1
## 97    0.0101088  -0.0253520   0.0455696             1
## 98   -0.0054958  -0.0409913   0.0299997             1
## 99   -0.0205793  -0.0565753   0.0154168             1
## 100   0.0035394  -0.0319980   0.0390768             1
## 101  -0.0233856  -0.0586502   0.0118791             1
## 102   0.0073179  -0.0288438   0.0434795             1
## 103  -0.0086537  -0.0443555   0.0270481             1
## 104   0.0122331  -0.0232013   0.0476674             1
## 105   0.0127516  -0.0227928   0.0482960             1
## 106  -0.0199937  -0.0557514   0.0157639             1
## 107   0.0185273  -0.0170278   0.0540825             1
## 108  -0.0185376  -0.0542334   0.0171582             1
## 109  -0.0414948  -0.0766554  -0.0063342             0
## 110  -0.0176417  -0.0522924   0.0170090             1
## 111  -0.0275557  -0.0624288   0.0073174             1
## 112   0.0003012  -0.0348286   0.0354310             1
## 113  -0.0513428  -0.0874710  -0.0152146             0
## 114  -0.0130487  -0.0493153   0.0232179             1
## 115  -0.0038245  -0.0395667   0.0319177             1
## 116   0.0006508  -0.0359880   0.0372895             1
## 117   0.0064366  -0.0293382   0.0422114             1
## 118   0.0050912  -0.0304286   0.0406110             1
## 119   0.0141831  -0.0211602   0.0495265             1
## 120  -0.0255897  -0.0612252   0.0100458             1
## 121   0.0296222  -0.0053549   0.0645993             1
## 122   0.0052114  -0.0307209   0.0411437             1
## 123  -0.0243995  -0.0609898   0.0121908             1
## 124   0.0381040   0.0016036   0.0746044             0
## 125   0.0039015  -0.0316666   0.0394695             1
## 126   0.0324262  -0.0039653   0.0688177             1
## 127  -0.0021379  -0.0381598   0.0338840             1
## 128  -0.0212123  -0.0566400   0.0142154             1
```

```
## 129   0.0060904  -0.0298804   0.0420611             1
## 130   0.0142518  -0.0219702   0.0504737             1
## 131   0.0012816  -0.0356453   0.0382086             1
## 132  -0.0125322  -0.0490203   0.0239559             1
## 133   0.0049564  -0.0300079   0.0399206             1
## 134  -0.0190919  -0.0546320   0.0164482             1
## 135   0.0027143  -0.0331567   0.0385853             1
## 136   0.0009666  -0.0351077   0.0370409             1
## 137  -0.0107947  -0.0472321   0.0256427             1
## 138  -0.0036940  -0.0387358   0.0313478             1
## 139  -0.0016589  -0.0377822   0.0344643             1
## 140   0.0050871  -0.0306992   0.0408734             1
## 141  -0.0080339  -0.0423915   0.0263238             1
## 142   0.0144445  -0.0209353   0.0498243             1
## 143  -0.0140612  -0.0502821   0.0221596             1
## 144  -0.0057160  -0.0418512   0.0304193             1
## 145  -0.0505733  -0.0866764  -0.0144703             0
## 146   0.0031317  -0.0322092   0.0384726             1
## 147   0.0112386  -0.0248881   0.0473653             1
## 148   0.0307962  -0.0045678   0.0661602             1
## 149   0.0486622   0.0128055   0.0845190             0
## 150  -0.0021208  -0.0386715   0.0344299             1
## 151  -0.0051740  -0.0406151   0.0302670             1
## 152   0.0124564  -0.0235176   0.0484303             1
## 153  -0.0180247  -0.0543202   0.0182708             1
## 154   0.0037214  -0.0327193   0.0401620             1
## 155   0.0158358  -0.0204935   0.0521652             1
## 156   0.0047154  -0.0308656   0.0402964             1
## 157   0.0186252  -0.0160640   0.0533143             1
## 158   0.0181221  -0.0181391   0.0543834             1
## 159  -0.0130069  -0.0490858   0.0230721             1
## 160  -0.0240735  -0.0591725   0.0110256             1
## 161  -0.0141651  -0.0500198   0.0216897             1
## 162   0.0013348  -0.0337508   0.0364205             1
## 163   0.0099742  -0.0258028   0.0457511             1
## 164  -0.0136318  -0.0489832   0.0217196             1
## 165  -0.0483612  -0.0835555  -0.0131668             0
## 166   0.0032477  -0.0323637   0.0388591             1
## 167   0.0036615  -0.0321798   0.0395028             1
## 168  -0.0239945  -0.0596045   0.0116156             1
## 169  -0.0170821  -0.0532935   0.0191294             1
## 170  -0.0008841  -0.0367548   0.0349866             1
## 171   0.0164516  -0.0194232   0.0523265             1
## 172  -0.0337204  -0.0691566   0.0017158             1
```

```
## 173   0.0039740  -0.0319877   0.0399357             1
## 174  -0.0003175  -0.0359544   0.0353193             1
## 175  -0.0136335  -0.0478048   0.0205378             1
## 176   0.0119160  -0.0244215   0.0482534             1
## 177  -0.0248979  -0.0600821   0.0102863             1
## 178   0.0114422  -0.0243498   0.0472341             1
## 179   0.0254674  -0.0099476   0.0608824             1
## 180  -0.0063897  -0.0419781   0.0291987             1
## 181   0.0022030  -0.0333867   0.0377926             1
## 182   0.0062179  -0.0295304   0.0419661             1
## 183  -0.0030928  -0.0398819   0.0336964             1
## 184  -0.0116764  -0.0476108   0.0242579             1
## 185  -0.0176273  -0.0543799   0.0191253             1
## 186   0.0061038  -0.0308903   0.0430979             1
## 187   0.0300347  -0.0062431   0.0663125             1
## 188  -0.0028723  -0.0380911   0.0323466             1
## 189   0.0336142  -0.0023791   0.0696075             1
## 190  -0.0006728  -0.0369962   0.0356505             1
## 191  -0.0157786  -0.0511136   0.0195564             1
## 192  -0.0071275  -0.0423067   0.0280517             1
## 193   0.0079981  -0.0290632   0.0450595             1
## 194   0.0170803  -0.0197589   0.0539195             1
## 195  -0.0211645  -0.0566879   0.0143590             1
## 196  -0.0186346  -0.0546066   0.0173374             1
## 197  -0.0045474  -0.0407236   0.0316289             1
## 198  -0.0227765  -0.0585251   0.0129721             1
## 199   0.0119097  -0.0237606   0.0475799             1
## 200  -0.0083345  -0.0446391   0.0279701             1

porcentaje <- mean(results$contains_mean) * 100

cat("\n", porcentaje, "% de los intervalos de",
"confianza contienen la media real\n")

##
##  93 % de los intervalos de confianza contienen la media real
```

En conclusión, el 93 % de los intervalos generados incluyen al 0.

4. Standard deviation of a proportion

Assume a manager is using the sample proportion $\hat{p}$ to estimate the proportion p of a new shipment of computer chips that are defective. He doesn't know p for this shipment, but in previous shipments it has been close to 0.01, that is 1 % of chips have been defective.

4.1. If the manager wants the standard deviation of $\hat{p}$ to be about 0.02, how large a sample should she take based on the assumption that the rate of defectives has not changed dramatically?

En clase se vieron los estimadores de probabilidad, donde se quiere estimar

$$p = (PX \in A)$$

Donde A es el subconjunto del espacio de estados de Ω de X. Es decir, para nuestro problema, este subconjunto de espacios en la muestra que tomó la administradora del cargamento de chips defectuosos. Se puede definir la variable indicadora Z como:

$$Z = \begin{cases} 1, & X \in A \\ 0, & X \notin A \end{cases}$$

1 significa que sí está defectuoso y 0 que no lo está. Se puede escribir el estimador como:

$$p = E[Z]$$

En clase se realizó una demostración de como el valor esperado de Z se le puede asignar a p. Se define la varianza de Z como:

$$\begin{aligned} \text{Var}(Z) &= E[Z^2] - E[Z]^2 \\ &= (1^2 \times P(X \in A) + 0^2 \times P(X \notin A)) - (P(X \in A)) \\ &= p - p^2 \end{aligned}$$

Aplicando factorización:

$$\text{Var}(Z) = p(1-p)$$

Estimar p se puede realizar mediante el promedio muestral de Z, es decir, se puede estimar la proporción $\hat{p}$ con el promedio de los chips que son defectuosos:

$$\hat{p} = \frac{1}{n} \sum_{i=1}^{n} Z_i$$

Se puede calcular entonces, la varianza del estimador:

$$\text{Var}(\hat{p}) = \frac{1}{n} \text{Var} \left(\sum_{i=1}^{n} Z_i \right)$$

Por independencia de los valores del cargamento de chips Z_i se puede meter la varianza de estos en la suma:

$$\begin{aligned}\text{Var}(\hat{p}) &= \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}(Z_i) \\ &= \frac{1}{n^2}\sum_{i=1}^{n}p(1-p) = \frac{1}{n}p(1-p)\end{aligned}$$

Por lo tanto, una aproximación es:

$$\text{Var}(\hat{p}) \approx \frac{1}{n}\hat{p}(1-\hat{p})$$

Teniendo en cuenta que la desviación estándar es el cuadrado de la varianza, se puede despejar:

$$\sigma_{\hat{p}} \approx \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

El problema nos dice que la desviación estándar de $\hat{p}$ ($\sigma_{\hat{p}}$) debe ser sobre 0.02 y nos piden hallar cuántas muestras debe tomar la administradora, por lo tanto, en la parte izquierda de la ecuación reemplazamos por 0.02, se despeja para n y la proporción de defectuosos se toma como 0.01 debido a que en el problema se nos dice que no ha cambiado tanto.

$$\begin{aligned}\sigma_{\hat{p}} &\approx \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \\ 0.02 &\approx \sqrt{\frac{0.1(1-0.1)}{n}} \\ 0.02 &\approx \sqrt{\frac{0.0099}{n}} \\ (0.02)^2 &\approx \left(\sqrt{\frac{0.0099}{n}}\right)^{\not 2} \\ 0.0004 &\approx \frac{0.0099}{n} \\ n &\approx \frac{0.0099}{0.0004} = 24.75\end{aligned}$$

Por lo tanto, la administradora debe tomar un muestra de por lo menos 25 chips para tener una desviación estándar de al menos 0.02.

4.2. Now suppose something went wrong with the production run and the actual proportion of defectives in the shipment is 0.3, that is 30 % are defective. Now what would be the actual standard deviation of $\hat{p}$ for the sample size you choose in a)?

Teniendo 25 chips y cambiando el valor de la proporción de defectuosos, se obtiene:

$$\begin{aligned}\sigma_{\hat{p}} &\approx \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \\ &\approx \sqrt{\frac{0.3(1-0.3)}{25}} \\ &\approx 0.091\end{aligned}$$

Es decir, si la proporción de detección es del 30 %, la desviación estándar de $\hat{p}$ con una cantidad de chips de 25, será aproximadamente 0.091.

5. Bets

You pay 10.000 pesos to participate in a bet game, which consists in tossing two coins together. If two heads fall, you earn 15.000 pesos. If one head and one tail fall you earn 5.000 pesos. In any other case you earn nothing. Let X the random variable of your profit.

5.1. Analytically find the probability mass function p_X, the mean $E[X]$, and variance $\mathrm{Var}(X)$ of X.

Dado que X es la variable aleatoria de la ganancia, podemos listar los posibles valores que puede tomar de acuerdo a los lanzamientos:

Salida moneda	**Ganancia**	**Ganancia X**	**Probabilidad**
HH	15.000	+5.000	0.25
HT-TH	5.000	−5.000	0.5
TT	0	−10.000	0.25

Cuadro 1: Posibles salidas y ganancias al tirar dos monedas, (H) cara y (T) sello.

Del cuadro 1 se toma que los posibles valores para las ganancias son:

$$X \in \{-5000, -10000, 5000\}$$

Teniendo en cuenta los valores que puede tomar X, la función probabilidad de masa debe retornar la probabilidad (valga la redundancia) de obtener alguno de los valores de X, en el cuadro 1 se pueden ver dichas probabilidades, estas fueron calculadas de acuerdo a contar cuántas veces pueden salir los diferentes valores para la ganancia de X.

$$p_X(x) = \begin{cases} 0.25, & x = 5000 \\ 0.5, & x = -5000 \\ 0.25, & x = -10000 \\ 0, & \text{En otro caso} \end{cases}$$

Debido a que estamos con un conjunto continuo para X, el valor esperado se multiplica la probabilidad por cada valor de este:

$$E[X] = \sum_{I=1}^{3} x\dot{p}_X(x) = 5000(0.25) + (-5000)(0.5) + (-10000)(0.25)$$
$$E[X] = 1250 - 2500 - 2500 = -3750$$

Es decir, en promedio se pierde 3.750 pesos por jugada.
Para calcular la varianza se puede utilizar la fórmula:

$$\text{Var}(X) = E[X^2] - (E[X])^2$$

Se computa $E[X^2]$:

$$\begin{aligned} E[X^2] &= 5000^2(0.25) + (-5000)^2(0.5) + (-10000)^2(0.25) \\ &= 25000000(0.25) + 25000000(0.5) + 100000000(0.25) \\ &= 6250000 + 12500000 + 25000000 = 43750000 \end{aligned}$$

Ahora se le resta el cuadrado de la media:

$$\text{Var}(X) = 43750000 - (-3750)^2 = 43750000 - 14062500 = 29687500$$

5.2. Write a code that simulates the r.v. X using the command sample. Generate an *iid* `sample` $\{X_i\}$ of size $n = 10^5$.

```
outcomes <- c(-10000, -5000, 5000)
probabilities <- c(0.25, 0.5, 0.25)

n <- 10^5

X <- sample(outcomes, size = n , replace = TRUE, prob = probabilities)

length(X)

## [1] 100000

summary(X)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -10000  -10000   -5000   -3750    5000    5000
```

En este fragmento de código se realiza la creación del vector `X` de tamaño 10^5 creado con la función `sample` que toma las salidas que se explicaron anteriormente, el tamaño de 10^5, que se reemplace y la probabilidad.

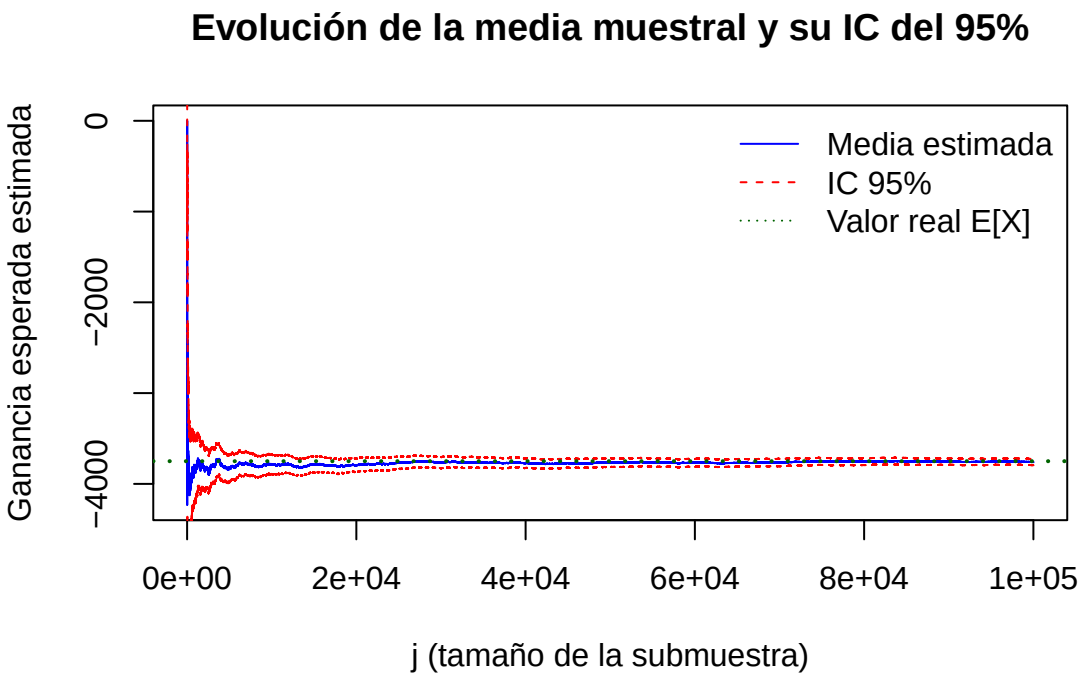
5.3. Modify your code to calculate: (i) the estimated mean profit $\bar{X}_j$ for each sample subsequence: $\{X_1, X_2, ..., X_j\}$, $j = 2, 3, \ldots, n$, (ii) the 95 % CI's of each estimated mean profit $\bar{X}_j$.

```
mean_estimates <- numeric(n)
lower_CI <- numeric(n)
upper_CI <- numeric(n)
ie <- 1.96

for (j in 2:n) {
  x_sub <- X[1:j]
  mean_j <- mean(x_sub)
  sd_j <- sd(x_sub)
  se_j <- sd_j / sqrt(j)

  mean_estimates[j] <- mean_j
  lower_CI[j] <- mean_j - ie * se_j
  upper_CI[j] <- mean_j + ie * se_j
}
```

En esta parte del código se crean tres vectores `mean_estimates`, `lower_CI`, `upper_CI` de tamaño 10^5 y también una variable `ie`. El bucle en cada iteración calcula la media muestral de los valores $\bar{X}_j$, la desviación estándar `sd_j` y el error estándar, estos son necesarios para luego calcular el intervalo de confianza junto con la media.

5.4. Plot $\bar{X}_j$ and their 95 % CI's in terms of $j = 2, \ldots, 10^5$. Add an horizontal line corresponding to the actual value $E[X]$.

```
# Your plotting code remains the same
plot(2:n, mean_estimates[2:n], type = "l", col = "blue", lwd = 1,
     xlab = "j (tamaño de la submuestra)",
     ylab = "Ganancia esperada estimada",
     main = "Evolución de la media muestral y su IC del 95%")
lines(2:n, lower_CI[2:n], col = "red", lty = 2)
lines(2:n, upper_CI[2:n], col = "red", lty = 2)
abline(h = -3750, col = "darkgreen", lty = 3, lwd = 2)
legend("topright", legend = c("Media estimada", "IC 95%", "Valor real E[X]"),
       col = c("blue", "red", "darkgreen"), lty = c(1, 2, 3), bty = "n")
```



Se puede ver que a medida que se aumenta la submuestra j, la media estimada $\bar{X}$ se acerca más al valor esperado $E[X]$ y también, el intervalo de confianza se va acortando más.

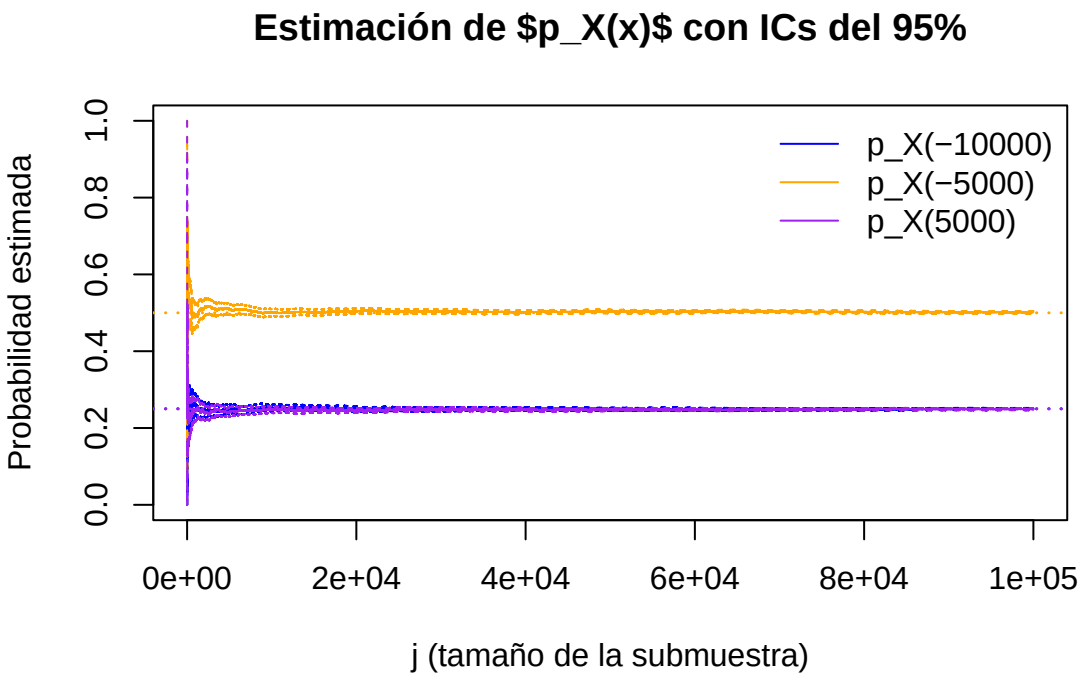
5.5. Repeat *c)* and *d)* to estimate the probabilities $p_X(x)$, their 95 % CI's, and their plots for each $j = 2, \ldots, 10^5$, adding the actual values.

```r
# Posibles valores de X
outcomes <- c(-10000, -5000, 5000)
probs_true <- c(0.25, 0.5, 0.25)

# Inicializar matrices para almacenar las estimaciones y CIs
p_estimates <- matrix(0, nrow = n, ncol = 3)
p_lower <- matrix(0, nrow = n, ncol = 3)
p_upper <- matrix(0, nrow = n, ncol = 3)

z <- 1.96  # Nivel de confianza del 95%

for (j in 2:n) {
  x_sub <- X[1:j]
  for (i in 1:3) {
    val <- outcomes[i]
    p_hat <- mean(x_sub == val)
```

```
    se <- sqrt(p_hat * (1 - p_hat) / j)

    p_estimates[j, i] <- p_hat
    p_lower[j, i] <- max(0, p_hat - z * se)
    p_upper[j, i] <- min(1, p_hat + z * se)
  }
}
```

Gráfico de las probabilidades estimadas con sus ICs para cada posible valor de X:

```
colors <- c("blue", "orange", "purple")
labels <- c("p_X(-10000)", "p_X(-5000)", "p_X(5000)")
true_values <- c(0.25, 0.5, 0.25)

plot(2:n, p_estimates[2:n,1], type = "l", col = colors[1], ylim = c(0,1),
     xlab = "j (tamaño de la submuestra)", ylab = "Probabilidad estimada",
     main = "Estimación de $p_X(x)$ con ICs del 95%")

for (i in 1:3) {
  lines(2:n, p_estimates[2:n,i], col = colors[i], lwd = 1)
  lines(2:n, p_lower[2:n,i], col = colors[i], lty = 2)
  lines(2:n, p_upper[2:n,i], col = colors[i], lty = 2)
  abline(h = true_values[i], col = colors[i], lty = 3, lwd = 1.5)
}

legend("topright", legend = labels, col = colors, lty = 1:1, bty = "n")
```



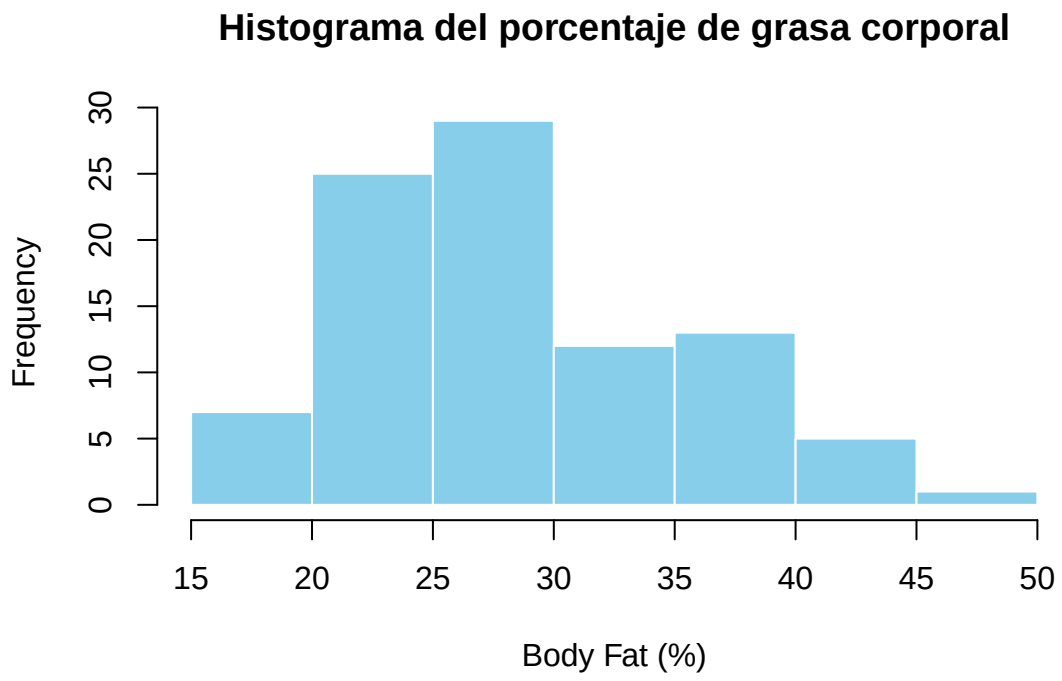
Se observa cómo las probabilidades estimadas convergen a los valores reales a medida que aumenta el tamaño de muestra j, y cómo los intervalos de confianza se vuelven más estrechos.

6. Bootstrap

This exercise is based on the article Introduction to Bootstrapping in Statistics with an Example and the dataset body fat.csv that contains the body fat percentages of 92 adolescent girls. Generate a program that gives:

6.1. An histogram of the sample data.

```r
# Leer el archivo
fat_data <- read.csv("body_fat.csv" )

# Histograma
hist(fat_data[[colnames(fat_data)[1]]],
     main = "Histograma del porcentaje de grasa corporal",
     xlab = "Body Fat (%)",
     col = "skyblue",
     border = "white")
```



Realizar el histograma en R es bastante sencillo, únicamente tener el archivo `.csv` en la misma ruta local que donde se está trabajando, utilizar la función `read.csv()` para leer este archivo y luego con la función `hist()` se colocan los datos de la primera columna y otras opciones para el estilo del gráfico.

Se muestra que la mayor frecuencia de grasa corporal se encuentra entre 20-30 %.

6.2. The 95 % confidence interval of the mean of the data from the traditional method (i.e., via the Central Limit Theorem).

```
# Estadísticas necesarias
media <- mean(fat_data[[colnames(fat_data)[1]]])
std <- sd(fat_data[[colnames(fat_data)[1]]])
n <- length(fat_data[[colnames(fat_data)[1]]])

# Cálculo del error estándar
se <- std / sqrt(n)

# Intervalo de confianza del 95%
c <- 1.96
li <- media - c*se
ls <- media + c*se

# Mostrar resultado
```

```r
cat(sprintf("The 95%% confidence interval for the mean is: [%.2f, %.2f]",
                                                              li, ls))

## The 95% confidence interval for the mean is: [27.14, 29.99]
```

Como se puede apreciar en el programa, hay un 95 % de confianza para que la media esté en el intervalo $[27.1370094, 29.9934253]$. Las funciones de R ayudan mucho a hacer los cálculos del promedio, la desviación estándar y el valor de n se sacaron con la ayuda de estas funciones. Los intervalos de confianza se sacaron como ya se viene haciendo, utilizando un $c = 1.96$.

6.3. A number of 500 bootstrapped samples from the original dataset, with 92 observations each.

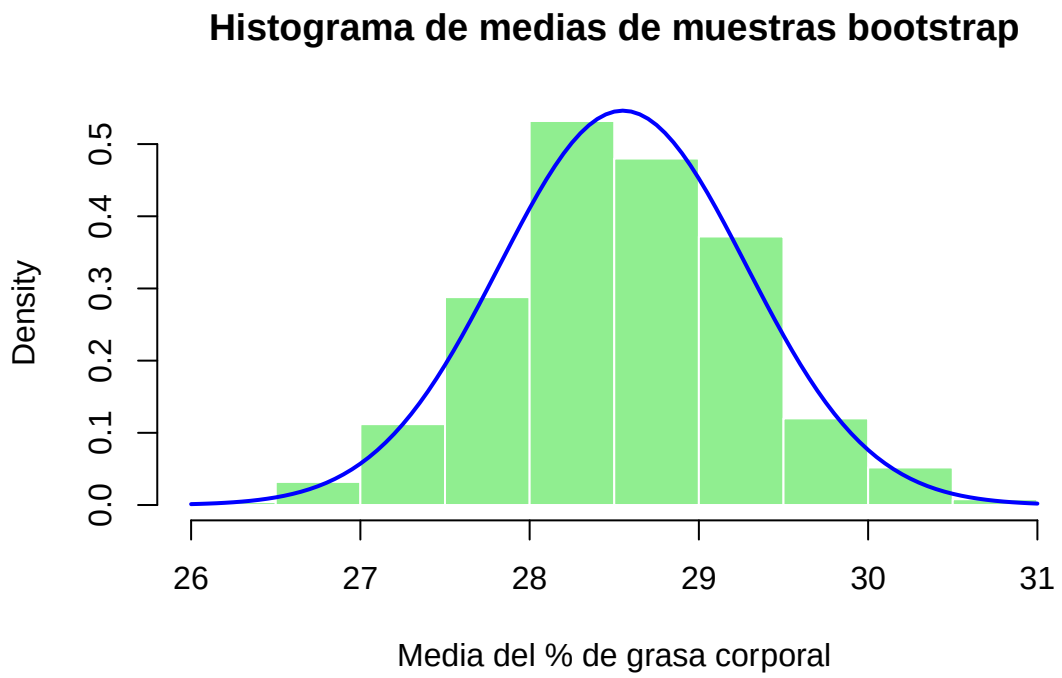
```r
original_data <- fat_data[[colnames(fat_data)[1]]]
n <- length(original_data)

# Generar 500 muestras bootstrap
bootstrap_samples <- replicate(500, sample(original_data, size = n,
                     replace = TRUE), simplify = FALSE)
```

Este código utiliza la función `replicate()` para repetir la operación 500 veces, la función `sample()` toma una muestra con reemplazo del tamaño de 92. Esto termina generando que `bootstrap_samples` tenga 500 vectores, cada uno con 92 observaciones.

6.4. An histogram of the means of each bootstrapped sample.

```r
# Calcular la media de cada muestra bootstrap
bootstrap_means <- sapply(bootstrap_samples, mean)

# Histograma de las medias
hist(bootstrap_means,
     main = "Histograma de medias de muestras bootstrap",
     xlab = "Media del % de grasa corporal",
     col = "lightgreen",
     border = "white",
     probability = TRUE)  # Importante: escala el eje y para densidades

# Parámetros de la normal
mean_boot <- mean(bootstrap_means)
sd_boot <- sd(bootstrap_means)
```

```r
# Añadir curva de distribución normal
curve(dnorm(x, mean = mean_boot, sd = sd_boot),
      col = "blue", lwd = 2, add = TRUE)
```



Con la función `sapply()` le aplica a los elementos en `bootstrap_samples` la función `mean()`. Después se crea el histograma y se dibuja una curva de tipo campana para una ditribución normal, reflejando la similitud de los promedios de los *bootstraps*.

6.5. A 95 % bootstrapped confidence interval of the mean.

```r
# Calcular intervalo de confianza bootstrap del 95%
bootstrap_ci <- quantile(bootstrap_means, probs = c(0.025, 0.975))


bootstrap_ci

##     2.5%    97.5%
## 27.14875 30.03261
```

Se utiliza la función `quantile()` para obtener los intervalos exactos, para posteriormente poder hacer la comparación de ambos resultados.

Entonces, el 95 % del intervalo de confianza para la media es:

$$[27.15,\ 30.03]$$

6.6. A comparison of both confidence intervals

Con el método tradicional del teórema central del límite:

$$[27.14,\ 29.99]$$

Con el método *bootstrap*

$$[27.15,\ 30.03]$$

7. Reliability of a system

Suppose a 3-out-of-4 system where each component is functioning with probabilities $p = \{pi\}$:

7.1. Write the structure function of the system $\phi(x)$.

La estructura funciona si al menos 3 de los 4 componentes están estructurados. Dado que hay dos estados; funciona ($\phi = 1$) o lo contrario ($\phi = 0$). Se puede tener un estado de los componentes $x = (x_1, x_2, x_3, x_4)$, donde cada $x_i \in \{0, 1\}$ representa si el componente i está funcionando.

La estructura entonces es:

$$\phi(x_1, x_2, x_3, x_4) = \begin{cases} 1 & \text{si } x_1 + x_2 + x_3 + x_4 \geq 3 \\ 0 & \text{si } x_1 + x_2 + x_3 + x_4 < 3 \end{cases}$$

7.2. Deduce analytically the reliability function $R(\mathbf{p})$. Evaluate it when $\{p_i\} = \{0.9, 0.5, 0.2, 0.1\}$

Se tomó gran parte de la información del libro *Introduction to Probability Models* de Sheldon Ross [1].

Suponiendo que X_i, el estado del i-ésimo componente, es una variable aleatoria tal que:

$$P\{X_i = 1\} = p_i = 1 - P\{X_i = 0\}$$

El valor p_i es la probabilidad de que el i-ésimo componente está funcionando, esto se llama la confiabilidad del i-ésimo componente. Si definimos r como:

$$r = P\{\phi(X) = 1\}, \quad \text{donde } X = (X_1, \ldots, X_n)$$

r es llamada la confiabilidad del sistema. Cuando los componentes, es decir, las variables aleatorias $X_i, i = 1, \ldots, n$, son independientes, se expresa r como una función de componentes de confiabilidad. Es decir:

$$r = r(\mathbf{p}), \quad \text{donde } \mathbf{p} = (p_1, \ldots, p_n)$$

Esta función $r(\mathbf{p})$ es llamada la función de confiabilidad, que es la que se busca encontrar paes este sistema 3-out-of-4.

Dado que para este sistema se necesita que por lo menos 3 de los 4 componentes sirvan, uno puede pasar a la función ϕ unos parámetros de $(1, 1, 1, 0)$, simbolizando que los componentes x_1, x_2 y x_3 están funcionando. La función de confiabilidad está dada entonces por todas las posibles combinaciones de los elementos que sumadas den mayor o igual a 3, es decir:

$$r(\mathbf{p}) = P\left\{\sum_{i=1}^{n} X_i \geq 3\right\}$$

Esto se puede expandir como:

$$r(\mathbf{p}) = P\{X = (1,1,1,1)\} + P\{X = (1,1,1,0)\} + P\{X = (1,1,0,1)\} + P\{X = (1,0,1,1)\} + P\{X = (0,1,1,1)\}$$

Teniendo que p_i simboliza la probabilidad de que el i-ésimo componente está funcionando y 1 - p_i es que no está funcionando el componente, lo anterior se puede escribir como:

$$r(\mathbf{p}) = p_1p_2p_3p_4 + p_1p_2p_3(1-p_4) + p_1p_2(1-p_3)p_4 + p_1(1-p_2)p_3p_4 + (1-p_1)p_2p_3p_4$$

Los casos para los cuales el sistema cuenta con un componente no funcionando, es decir, $1 - p_i$, se pueden agrupar en uno mismo, quedando:

$$r(\mathbf{p}) = p_1p_2p_3p_4 + p_1p_2p_3 + p_1p_2p_4 + p_1p_3p_4 + p_2p_3p_4 - 4p_1p_2p_3p_4$$

Sustrayendo con el término positivo $p_1p_2p_3p_4$:

$$r(\mathbf{p}) = p_1p_2p_3 + p_1p_2p_4 + p_1p_3p_4 + p_2p_3p_4 - 3p_1p_2p_3p_4$$

Ahora, podemos evaluar cuando $\mathbf{p} = \{0.9, 0.5, 0.2, 0.1\}$:

$$r(\mathbf{p}) = 0.9 \times 0.5 \times 0.2 + 0.9 \times 0.5 \times 0.1 + 0.9 \times 0.2 \times 0.1 + 0.5 \times 0.2 \times 0.1 - 3(0.9 \times 0.5 \times 0.2 \times 0.1)$$

Esto se puede colocar en una calculadora y va a dar el resultado de **0.136**, se puede construir una pequeña función en R para ahorrar calculos en el caso de que se den más conjuntos para $\mathbf{p}$:

```r
reli_fun <- function(p1, p2, p3, p4){
  return (p1*p2*p3 + p1*p2*p4 + p1*p3*p4 + p2*p3*p4 - 3 * (p1 * p2 * p3 * p4))
}

reli_fun(0.9, 0.5, 0.2, 0.1)
```

```
## [1] 0.136
```

7.3. Estimate via simulation, for each $n = 10^2, 10^3, 10^4, 10^5$ realizations, the reliability of the system $R(p)$. Take $\{pi\} = \{0.9, 0.5, 0.2, 0.1\}$.

```
# Given probabilities
p <- c(0.9, 0.5, 0.2, 0.1)

# Function to calculate system reliability for one trial
calculate_reliability <- function(p) {
  # Simulate the components as working (1) or failing (0) based on probabilities
  x <- rbinom(4, 1, p)

  # Apply the reliability function r(p) as given
  r <- sum(x[1]*x[2]*x[3]) + sum(x[1]*x[2]*x[4]) + sum(x[1]*x[3]*x[4]) +
              sum(x[2]*x[3]*x[4]) - 3 * prod(x)

  return(r)
}

# Function to run simulation for different number of realizations n
simulate_system_reliability <- function(n, p) {
  reliabilities <- numeric(n)

  for (i in 1:n) {
    reliabilities[i] <- calculate_reliability(p)
  }
  return(mean(reliabilities))
}

# Run simulation for each n: 10^2, 10^3, 10^4, 10^5 realizations
results <- sapply(c(10^2, 10^3, 10^4, 10^5),
            function(n) simulate_system_reliability(n, p))

# View results
names(results) <- c("n=10^2", "n=10^3", "n=10^4", "n=10^5")
print(results)

##  n=10^2  n=10^3  n=10^4  n=10^5
## 0.12000 0.12600 0.14050 0.13702
```

En primer lugar se tiene un vector `x` que simula cada componente del sistema retornando 1 o 0 con una probabilidad que es el conjunto para p_i que se define en el enunciado del problema.

Después se pasa a la función de confiabilidad. Seguido, se pasa a la función `simulate_system_reliability()` donde se simula la confiabilidad n veces y se guarda en un vector `reliabilities`. Por último, se corre la simulación para diferentes valores de n, guardando todo en `results`. Los resultados se presentan en el cuadro 2 y se ve que a medida que se hacen más simulaciones, la probabilidad de que el sistema esté funcionando se va acercando al valor obtenido analíticamente de 0.136.

n	Confiabilidad de que el sistema está funcionando.
10^2	0.12
10^3	0.126
10^4	0.1405
10^5	0.13702

Cuadro 2: Resultados de la función confiabilidad para diferentes valores de n.

7.4. Find the standard deviation of the reliability estimation for each n.

Se hace una modificación a la función `simulate_system_reliability()` para que también entregue la desviación estándar de cada estimación de la confiabilidad.

```
simulate_system_reliability <- function(n, p) {
 reliabilities <- numeric(n)

 for (i in 1:n) {
   reliabilities[i] <- calculate_reliability(p)
 }

 sd_val <- sd(reliabilities)

 return(c(SD = sd_val))
}
```

Se deja tal cual estaba la función, pero ahora retorna la desviación estándar de esa estimación para ese n en específico. Por lo tanto, ahora queda volver a correr la simulación para los diferentes n.

```
results <- sapply(c(10^2, 10^3, 10^4, 10^5), function(n) simulate_system_reliability(n,
names(results) <- c("n=10^2", "n=10^3", "n=10^4", "n=10^5")

# Print results
print(results)

##    n=10^2    n=10^3    n=10^4    n=10^5
## 0.3015113 0.3471607 0.3409914 0.3428752
```

Los resultados se muestran en el cuadro 3 reflejando cómo los sistemas varían entre ellos.

n	Desviación estándar
10^2	0.3015113
10^3	0.3471607
10^4	0.3409914
10^5	0.3428752

Cuadro 3: Desviación estándar para cada n.

7.5. Find the 95 % CI's for each n. Do the confidence intervals contain the actual reliability value from item 7.2?

Teniendo en cuenta:

$$CI = \bar{X} \pm c \times \frac{s}{\sqrt{n}}$$

Se puede volver a modificar nuevamente la función `simulate_system_reliability()` para que también retorne el intervalo superior e inferior, esto teniendo en cuenta la formula que se viene manejando durante todo el taller, por lo tanto, el código sería:

```
simulate_system_reliability <- function(n, p) {
  reliabilities <- numeric(n)
  for (i in 1:n) {
    reliabilities[i] <- calculate_reliability(p)
  }

  mean_val <- mean(reliabilities)
  sd_val <- sd(reliabilities)
  error_margin <- 1.96 * (sd_val / sqrt(n))  # 95% CI

  return(c(mean = mean_val,
           lower = mean_val - error_margin,
           upper = mean_val + error_margin))
}
```

Se ha agregado una variable de `mean_val` y el `error_margin` para luego entre ellas restar y sumar, respectivamente, para obtener el intervalo de confianza completo. Ahora, se puede volver a correr la simulación e imprimir la salida.

```
# Run simulation for each n
results <- t(sapply(c(10^2, 10^3, 10^4, 10^5),
                    function(n) simulate_system_reliability(n, p)))
```

```r
rownames(results) <- c("n=10^2", "n=10^3", "n=10^4", "n=10^5")

# Print result as a table
print(round(results, 5))

##           mean   lower   upper
## n=10^2 0.15000 0.07966 0.22034
## n=10^3 0.12200 0.10170 0.14230
## n=10^4 0.13630 0.12957 0.14303
## n=10^5 0.13569 0.13357 0.13781
```

Recordando que el valor de la sección 7.2 es 0.136, y el intervalo quedó como:

$$[0.1335674, 0.1378126]$$

Por lo tanto, el valor que se encontró anteriormente sí cae en el intervalo de confianza.

8. Reading assignment

Se optó por buscar una opción gratuita del libro en PDF, esto debido a que la versión que se encuentra en la Tadeo es muy incómoda por leer, el PDF se puede encontrar libre de descargar **aquí**. Se han tomado las siguientes ideas mientras se hacía la lectura del libro:

- Las muestras que se deben elegir de un población deben ser representativas y aleatorias, porque pueden pasar casos como el de las elecciones de Landon y Roosevelt, creando un sesgo para el *dataset*.
- Lo que se entiende por un muestreo estratificado es seleccionar nuestros datos por grupos. Por ejemplo, si quiero saber los ingresos de los habitantes de Bogotá, no me puedo tomar solo muestras de la gente estrato 6, porque me quedará sesgado el análisis y voy a concluir que todos los habitantes de Bogotá ganan mucho dinero. En cambio, tengo que reconocer cómo están segmentados los estratos sociales en la ciudad; 10 % para estrato 6, 20 % para el 5, etc.[2] De esos grupo ahora se toman mustras aleatorias y así hemos reducido el sesgo del experimento.
- Las diferencias entre $\bar{x}$ y μ están en que la información sobre muestras es observada, y la información sobre poblaciones grandes es a menudo inferida de muestras más pequeñas.

Referencias

[1] Sheldon M Ross. *Introduction to probability models*. Academic press, 2014.

[2] Datos inventados por los autores