

Taller de Programación en R - Taller 1

Andrés Díaz, Yilmer Palacios

2024-02-01

1 Primer Punto

1.1) Definan una semilla para trabajar durante el script. Respondan: ¿Por qué es importante definir una semilla?

Respuesta: Según R-Coder.com, una semilla es el iniciador de un generador de números pseudoaleatorios, que son utilizados para la simulación de distribuciones de probabilidad que sean requeridas por el algoritmo. Al ser pseudoaleatorios, los resultados obtenidos a partir de estos números generados son replicables siempre y cuando se utilice el mismo generador y la misma semilla.

Es importante definir una semilla en un algoritmo que vaya a utilizar un generador de números con distribuciones de probabilidad ya que permite que terceros puedan llegar al mismo resultado con el algoritmo y los datos utilizados. Si no se definiera semilla, los resultados cambiarían cada vez que se ejecutara el código, dada la naturaleza de los datos que se está utilizando.

Para este ejercicio, se definirá la semilla usando la fecha en la que se comenzó a trabajar en este taller.

```
set.seed(31012024)
```

1.2) Primero, creen una lista con números secuenciales de 1 en 1 desde el 1 hasta el 50. Luego, creen tres (3) listas diferentes que contengan respectivamente: una variable numérica de clase int que se distribuya de forma uniforme entre el intervalo 5 a 50, una lista que repita el carácter "Años" y una lista con nombres propios aleatorios de personas, todas las cuatro (4) listas deben tener el mismo tamaño.

```
set.seed(31012024) #La ejecutamos de nuevo para que tenga efecto sobre el runif
rm(list=ls())      # borramos environment
l_num = list(numeros = seq(1,50,1)) # Creamos la lista de 1 en 1 hasta 50
l_edad = list(edad= round(runif(50,5,50),0)) # Creamos la lista de edades con una distribución uniforme entre 5 y 50
l_texto = list(variable=rep("años",50)) # Creamos una lista con la palabra "años" repetida 50 veces
```

```
# Para la lista de nombres vamos a crear dos vectores, uno con los nombres y otro con los apellidos de los integrantes del curso; luego usaremos la función sample para tomar de los vectores elementos aleatorios; finalmente, uniremos los nombres y apellidos aleatorios para
```

formar la lista. Fuente: <https://r-coder.com/funcion-sample-r/>

```
rapellidos <- c("Aguirre", "Arteaga", "Borda", "Caballero", "Carbonell",  
"Carvajal", "Criollo", "Diaz", "Duquino", "Escobar", "Fernandez",  
"Fonseca", "Galindo", "Gonzalez", "Hernandez", "Huertas", "Karaman",  
"Lasso", "Naranjo", "Navarrete", "Navarro", "Neusa", "Osorno",  
"Palacios", "Patiño", "Perdomo", "Prada", "Rueda", "Villamizar",  
"Viveros", "Zuluaga", "Valencia", "Gomez", "Bejarano", "Silva", "Ortega",  
"Huerfano", "Cardenas", "Barreto", "Carreño", "Rocha", "Correa",  
"Jimenez", "Barragan", "Rincon", "Ramirez", "Charry", "Munoz",  
"Pedreros", "Lian", "Jaramillo", "DeZulategi", "Amaya", "Corredor",  
"Ortiz", "Rodriguez", "Martinez", "Olmos", "Echeverri", "Arias",  
"Moreno", "Rojas", "Aguilar", "Cuellar", "Guacheta", "Vargas", "Muaoz",  
"DelCastillo")
```

```
rnombres <- c("Dennis", "Diana", "Luis", "Alejandra", "Henry", "Pharad",  
"Andres", "Edinson", "Yuri", "Jose", "Laura", "German", "Valentina",  
"Daniel", "Samuel", "Manuela", "Manuel", "Angella", "Yilmer", "Raul",  
"Gabriel", "Angela", "Camilo", "Santiago", "Julian", "Juanita", "Juan",  
"Silvia", "Francisco", "Estefania", "Johana", "Carolina", "Olegario",  
"Valentina", "Nicolas", "Fernando", "Alejandro", "Romina", "Camila",  
"Salomon", "Sofia", "Felipe", "Enrique", "Katherine", "Sebastian",  
"Alexis", "Maria", "Alexander", "David", "Paula", "Felipe", "Juliana")
```

```
nom_apellido <- c(paste(sample(rnombres,50, replace=TRUE),  
sample(rapellidos,50, replace=TRUE)))
```

#Ahora si creamos la lista

```
l_nombres = list(nom_apellido)
```

1.3) Creen una lista en la que cada elemento j sea la concatenación de los elementos j de las tres listas creadas en el punto anterior. Ordenen y/o agreguen caracteres a cada elemento de la lista para que se consolide una oración con orden semántico que refleje la edad del individuo.

```
l_oracion <- c()
```

#creamos un bucle que va desde 1 hasta el número de elementos (50), es cada paso del bucle se concatenarán con "paste" las listas y se guardarán en una posición de la lista "l_oracion"

```
for (i in 1:length(l_nombres[[1]])){  
  l_oracion[[1]][i] <- paste(l_nombres[[1]][i],"tiene",l_edad[[1]][i],  
                             l_texto[[1]][i])  
}
```

```
l_oracion
```

```

## [[1]]
## [1] "Camila Rincon tiene 6 años"      "Laura Aguilar tiene 49 años"
## [3] "Alejandro Viveros tiene 29 años" "Daniel Muaoz tiene 40 años"
## [5] "Dennis Patiño tiene 38 años"     "Valentina Patiño tiene 11
años"
## [7] "Andres Guacheta tiene 5 años"     "Maria Arias tiene 22 años"
## [9] "Dennis Barreto tiene 29 años"     "Romina Navarrete tiene 16
años"
## [11] "Diana Silva tiene 31 años"        "Valentina Rodriguez tiene 36
años"
## [13] "Andres Palacios tiene 35 años"     "Camilo Martinez tiene 24
años"
## [15] "Juliana Arteaga tiene 38 años"     "Olegario Cardenas tiene 32
años"
## [17] "Manuela Charry tiene 49 años"     "Juliana DeZulategi tiene 36
años"
## [19] "Olegario Cuellar tiene 9 años"     "Juanita Arteaga tiene 38
años"
## [21] "Alejandro Neusa tiene 30 años"     "David Diaz tiene 27 años"
## [23] "Camila Jimenez tiene 10 años"     "Juanita Huertas tiene 35
años"
## [25] "Juanita Diaz tiene 21 años"        "Diana Arias tiene 24 años"
## [27] "Angela Galindo tiene 15 años"     "Raul Valencia tiene 31 años"
## [29] "Carolina Valencia tiene 46 años"  "Maria Villamizar tiene 46
años"
## [31] "Alejandro Charry tiene 47 años"     "Camila Guacheta tiene 21
años"
## [33] "Pharad Echeverri tiene 40 años"    "Nicolas Echeverri tiene 27
años"
## [35] "Yilmer Diaz tiene 26 años"         "Edinson Munoz tiene 38 años"
## [37] "Jose Rojas tiene 8 años"           "Salomon Jaramillo tiene 14
años"
## [39] "Paula Echeverri tiene 21 años"     "Alexander Rojas tiene 19
años"
## [41] "Laura Neusa tiene 22 años"         "Juliana Galindo tiene 24
años"
## [43] "Manuela Corredor tiene 21 años"    "Felipe Aguilar tiene 33
años"
## [45] "Paula Martinez tiene 10 años"      "Raul Caballero tiene 30
años"
## [47] "German Munoz tiene 39 años"        "Valentina Carbonell tiene 48
años"
## [49] "Silvia Galindo tiene 8 años"       "David Martinez tiene 29
años"

```

1.4) Usando un loop realicen un código que presente (print) la edad de cada uno de los individuos dentro de las listas, pero únicamente si el nombre del individuo empieza por una letra distinta de J y la edad sea distinta de un número par.

Creamos un ciclo for que va desde 1 hasta el número de elementos de La Lista oración (50), luego pasamos a dos codicionales, el primero evalúa

si el nombre inicia por J y el segundo si el residuo de dividir la edad en 2 es mayor que cero, si lo es, el número es impar, a los elementos que pasen ambas condiciones se les hará "print".

```
for (j in 1:length(l_oracion[[1]])){  
  
  if(substring(l_oracion[[1]][j],1,1)!="J"){  
  
    if(l_edad[[1]][j]%%2>0){  
  
      print(l_oracion[[1]][j])  
  
    }  
  
  }  
  
}
```

```
## [1] "Laura Aguilar tiene 49 años"  
## [1] "Alejandro Viveros tiene 29 años"  
## [1] "Valentina Patiño tiene 11 años"  
## [1] "Andres Guacheta tiene 5 años"  
## [1] "Dennis Barreto tiene 29 años"  
## [1] "Diana Silva tiene 31 años"  
## [1] "Andres Palacios tiene 35 años"  
## [1] "Manuela Charry tiene 49 años"  
## [1] "Olegario Cuellar tiene 9 años"  
## [1] "David Diaz tiene 27 años"  
## [1] "Angela Galindo tiene 15 años"  
## [1] "Raul Valencia tiene 31 años"  
## [1] "Alejandro Charry tiene 47 años"  
## [1] "Camila Guacheta tiene 21 años"  
## [1] "Nicolas Echeverri tiene 27 años"  
## [1] "Paula Echeverri tiene 21 años"  
## [1] "Alexander Rojas tiene 19 años"  
## [1] "Manuela Corredor tiene 21 años"  
## [1] "Felipe Aguilar tiene 33 años"  
## [1] "German Munoz tiene 39 años"  
## [1] "David Martinez tiene 29 años"
```

1.5) Programen una función que tome como entrada una lista con valores numéricos y que su output sea el promedio de los valores de la lista y la desviación estándar asociada a la misma muestra. Usando esta función respondan: ¿Cuál es la edad promedio de su lista? ¿Cuál es la desviación estándar?

```
funcion_param_stat <- function(lista_edad){  
  
  media <- mean(lista_edad)  
  des_est <- sd(lista_edad)
```

```

    print(paste("La edad promedio es",media))
    print(paste("La desviación estándar es",signif(des_est,4)))

}

funcion_param_stat(l_edad[[1]])

## [1] "La edad promedio es 27.66"
## [1] "La desviación estándar es 12.3"

```

1.6) Programen una función que tome como entrada una lista con valores numéricos y estandarice los valores. Es decir, que los transforme a una normal estándar los datos. Apliquen las funciones que desarrollaron en el literal 1.5) dentro de la función que propongan en este literal

```

funcion_norm_estan <- function(listaval){

  media <- mean(listaval)
  des_est <- sd(listaval)

  dist_estandar <- c()

  for (i in 1:length(listaval)){

    dist_estandar[i] <- (listaval[i]-media)/des_est

  }

  print(dist_estandar)

}

dist_estandar <- funcion_norm_estan(l_edad[[1]])

## [1] -1.7609744  1.7349582  0.1089430  1.0032514  0.8406498 -1.3544706
## [7] -1.8422752 -0.4601623  0.1089430 -0.9479668  0.2715445  0.6780483
## [13]  0.5967476 -0.2975608  0.8406498  0.3528453  1.7349582  0.6780483
## [19] -1.5170721  0.8406498  0.1902438 -0.0536585 -1.4357714  0.5967476
## [25] -0.5414630 -0.2975608 -1.0292676  0.2715445  1.4910559  1.4910559
## [31]  1.5723567 -0.5414630  1.0032514 -0.0536585 -0.1349593  0.8406498
## [37] -1.5983729 -1.1105684 -0.5414630 -0.7040646 -0.4601623 -0.2975608
## [43] -0.5414630  0.4341460 -1.4357714  0.1902438  0.9219506  1.6536574
## [49] -1.5983729  0.1089430

```

1.7) Por otra parte, generen una lista de listas llamada `outcomes_nominales`. Esta lista contendrá 3 vectores de 50 observaciones cada uno con los outcomes de interés: salario, índice de salud, experiencia laboral. Para esto, generen para cada vector valores numéricos de clase float basado en una distribución normal estándar con media 0 y varianza 1.

```
set.seed(31012024)
salario <- rnorm(50,0,1)
ind_salud <- rnorm(50,0,1)
exp_laboral <- rnorm(50,0,1)

outcomes_nominales <- list(Salario = salario,IndiceSalud = ind_salud,
ExpLaboral = exp_laboral)
print(outcomes_nominales)

## $Salario
## [1] -1.91416079  0.08283508  0.60387299 -2.41213551  0.06383401
0.19480237
## [7]  0.46030633  0.59475467  2.19532204 -1.40370623  0.14190947 -
1.20862880
## [13] -0.39271258 -0.76049622  1.34540467  1.46010798  0.78195863 -
0.05794777
## [19] -1.55772948 -0.38427497 -0.30211594 -0.38658020 -1.23173010
0.66510870
## [25] -1.46091704 -0.59775726  1.79699371  1.80880576 -0.85514235
1.23510008
## [31]  0.13529639  1.63720832 -0.18216793  0.11242070 -0.09909349 -
0.30273958
## [37] -0.34165724  3.54663044  0.34843065  0.66299703  0.32769197
0.53503017
## [43]  0.82915328  0.47189576  1.10595962 -0.51280806  0.49014971 -
1.86663771
## [49]  0.60413448 -0.39408111
##
## $IndiceSalud
## [1] -1.388088237  0.113564205 -0.391351305  0.168781987  0.004735466
## [6]  0.489238913  0.341522750  1.468029567 -0.368587272  0.129880746
## [11]  2.333766030 -0.880164560  0.523374668  1.818628231 -0.566250997
## [16]  1.959629655  1.187044268 -1.452526484 -0.642244988 -0.322394590
## [21] -0.165078338 -0.565036365  0.740957706 -0.195629731  0.805880574
## [26] -0.127574338  0.509536329  0.621140869  0.916252096 -0.762888720
## [31] -0.532742753  0.400304175  1.132465071  0.529173913 -0.087158133
## [36]  0.883525585 -1.492477526  0.296693762 -0.478331572 -0.695289255
## [41] -0.311569947 -1.594083429  0.166684231 -0.857302800 -0.245187888
## [46] -1.903302208  0.485943384 -0.519674229  0.854043886 -1.457759585
##
## $ExpLaboral
## [1]  0.297300659  0.372693193  0.103886105 -0.293698669 -1.468215527
## [6] -0.570178265  1.096950099 -1.144961795 -0.746812669  1.327276595
## [11] -0.919121509 -0.964707460 -0.384276547  0.125925023 -0.950705622
```

```
## [16] 1.659851435 0.005855219 0.046903205 -0.291512620 -0.036997326
## [21] 0.990550383 0.822904884 -2.711220988 -1.237013300 0.902699590
## [26] 1.620900029 0.831470677 0.469134755 1.602089729 -0.532298390
## [31] -1.072937527 -0.272019289 -0.477099217 0.781683806 0.113061748
## [36] -0.800484756 0.074146031 -0.020136954 0.612494007 0.510199231
## [41] -0.559683996 -2.250110313 1.005131196 -0.891300623 0.605136784
## [46] -0.999962139 -0.280055914 0.880141053 -1.507191621 -0.255060619
```

1.8) Creen una función que transforme una lista en una matriz. Para esto, la función debe tomar como input una lista y debe tener como output una matriz X que concatene los datos de esta lista y un vector de 1's.

```
funcion_lista_matriz <- function(lista){
  #Input
  #Vector de unos con la longitud de la edad
  #Vector de la edad con la edad
  vector_unos <-
matrix(rep(1,length(lista[[1]]),nrow=length(l_edad[[1]]))
  matriz <- cbind(lista,vector_unos)

  colnames(matriz)[1] <- "Edad"
  colnames(matriz)[2] <- "Constante"
  matriz

}

matriz = funcion_lista_matriz(l_edad[[1]])
print(matriz)
```

```
##      Edad Constante
## [1,]    6         1
## [2,]   49         1
## [3,]   29         1
## [4,]   40         1
## [5,]   38         1
## [6,]   11         1
## [7,]    5         1
## [8,]   22         1
## [9,]   29         1
## [10,]  16         1
## [11,]  31         1
## [12,]  36         1
## [13,]  35         1
## [14,]  24         1
## [15,]  38         1
## [16,]  32         1
## [17,]  49         1
## [18,]  36         1
## [19,]   9         1
## [20,]  38         1
```

```
## [21,] 30 1
## [22,] 27 1
## [23,] 10 1
## [24,] 35 1
## [25,] 21 1
## [26,] 24 1
## [27,] 15 1
## [28,] 31 1
## [29,] 46 1
## [30,] 46 1
## [31,] 47 1
## [32,] 21 1
## [33,] 40 1
## [34,] 27 1
## [35,] 26 1
## [36,] 38 1
## [37,] 8 1
## [38,] 14 1
## [39,] 21 1
## [40,] 19 1
## [41,] 22 1
## [42,] 24 1
## [43,] 21 1
## [44,] 33 1
## [45,] 10 1
## [46,] 30 1
## [47,] 39 1
## [48,] 48 1
## [49,] 8 1
## [50,] 29 1
```

```
class(matriz)
```

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

1.9) A partir de la función anterior consoliden una matriz X con la edad de los individuos estandarizada y un vector de 1's asociado a una constante.

```
vect_edad_estandar <- funcion_norm_estan(l_edad[[1]])
```

```
## [1] -1.7609744 1.7349582 0.1089430 1.0032514 0.8406498 -1.3544706
## [7] -1.8422752 -0.4601623 0.1089430 -0.9479668 0.2715445 0.6780483
## [13] 0.5967476 -0.2975608 0.8406498 0.3528453 1.7349582 0.6780483
## [19] -1.5170721 0.8406498 0.1902438 -0.0536585 -1.4357714 0.5967476
## [25] -0.5414630 -0.2975608 -1.0292676 0.2715445 1.4910559 1.4910559
## [31] 1.5723567 -0.5414630 1.0032514 -0.0536585 -0.1349593 0.8406498
## [37] -1.5983729 -1.1105684 -0.5414630 -0.7040646 -0.4601623 -0.2975608
## [43] -0.5414630 0.4341460 -1.4357714 0.1902438 0.9219506 1.6536574
## [49] -1.5983729 0.1089430
```



```
matriz_edad_estandar <- funcion_lista_matriz(vect_edad_estandar)
colnames(matriz_edad_estandar)[1] <- "Edad estándar"
print(matriz_edad_estandar)
```

```
##      Edad estándar Constante
## [1,] -1.7609744      1
## [2,]  1.7349582      1
## [3,]  0.1089430      1
## [4,]  1.0032514      1
## [5,]  0.8406498      1
## [6,] -1.3544706      1
## [7,] -1.8422752      1
## [8,] -0.4601623      1
## [9,]  0.1089430      1
## [10,] -0.9479668      1
## [11,]  0.2715445      1
## [12,]  0.6780483      1
## [13,]  0.5967476      1
## [14,] -0.2975608      1
## [15,]  0.8406498      1
## [16,]  0.3528453      1
## [17,]  1.7349582      1
## [18,]  0.6780483      1
## [19,] -1.5170721      1
## [20,]  0.8406498      1
## [21,]  0.1902438      1
## [22,] -0.0536585      1
## [23,] -1.4357714      1
## [24,]  0.5967476      1
## [25,] -0.5414630      1
## [26,] -0.2975608      1
## [27,] -1.0292676      1
## [28,]  0.2715445      1
## [29,]  1.4910559      1
## [30,]  1.4910559      1
## [31,]  1.5723567      1
## [32,] -0.5414630      1
## [33,]  1.0032514      1
## [34,] -0.0536585      1
## [35,] -0.1349593      1
## [36,]  0.8406498      1
## [37,] -1.5983729      1
## [38,] -1.1105684      1
## [39,] -0.5414630      1
## [40,] -0.7040646      1
## [41,] -0.4601623      1
## [42,] -0.2975608      1
## [43,] -0.5414630      1
## [44,]  0.4341460      1
## [45,] -1.4357714      1
```

```
## [46,]      0.1902438      1
## [47,]      0.9219506      1
## [48,]      1.6536574      1
## [49,]     -1.5983729      1
## [50,]      0.1089430      1
```

Segundo punto

2.1) Programen una función que tome como input una matriz X estocástica de rango completo y un vector y_i , posteriormente, el output debe corresponder a una estimación puntual del estimador (β_1) de Mínimos Cuadrados Ordinarios (MCO) para la muestra y a su error estándar asociado ($\sigma\beta$).

#Creamos dos funciones exactas salvo que la primera devuelve el estimado beta B y la segunda su error estándar.

```
funcion_MCO_beta1 <- function(matriz_in,y_in){

  #Input
  # Matriz_in: Matriz estocástica de rango completo
  # y_in: Vector y_i

  #Fórmula 1
  # Beta_1 = (t(X)*X)^-1*(t(X)*y_i)

  beta_1 <- solve(t(matriz_in) %*% matriz_in) %*%
             (t(matriz_in) %*% y_in)

  #Fórmula 2
  # ee(Beta_1) = (Var(error)/STC)^0.5

  # Var(error) = SRC/(n-k-1) #Varianza del error
  #SRC = suma((y_pred - y_in)^2) #Sumatoria de los residuales al
cuadrado
  # k = 1 porque se van a hacer regresiones lineales simples
  #y_pred = X*beta_1 #Resultado predicho

  # STC = suma((y_in - mean(y_in))^2) #Suma Total de Cuadrados

  y_pred <- matriz_in %*% beta_1
  SRC <- sum((y_pred - y_in)^2)
  Var_error <- SRC/(length(y_in)-1-1)
  STC <- sum((y_in - mean(y_in))^2)

  eeBeta_1 <- (Var_error/STC)^0.5

  #Output
  # Beta_1
  beta_1[1]
```

```

}

funcion_MCO_ee_beta1 <- function(matriz_in,y_in){

  #Input
  # Matriz_in: Matriz estocástica de rango completo
  # y_in: Vector y_i

  #Fórmula 1
  #  $Beta_1 = (t(X)*X)^{-1}*(t(X)*y_i)$ 

  beta_1 <- solve(t(matriz_in) %*% matriz_in) %*%
    (t(matriz_in) %*% y_in)

  #Fórmula 2
  #  $ee(Beta_1) = (Var(error)/STC)^{0.5}$ 

  #  $Var(error) = SRC/(n-k-1)$  #Varianza del error
  #SRC = suma((y_pred - y_in)^2) #Sumatoria de Los residuales al
cuadrado
  # k = 1 porque se van a hacer regresiones lineales simples
  #y_pred = X*beta_1 #Resultado predicho

  #  $STC = suma((y_in - mean(y_in))^2)$  #Suma Total de Cuadrados

  y_pred <- matriz_in %*% beta_1
  SRC <- sum((y_pred - y_in)^2)
  Var_error <- SRC/(length(y_in)-1-1)
  STC <- sum((y_in - mean(y_in))^2)

  eeBeta_1 <- (Var_error/STC)^0.5

  #Output
  # Error estándar de Beta_1
  eeBeta_1
}

#Estimación de variables (Prueba)

#X <- matriz_edad_estandar
#y_in <- matrix(outcomes_nominales[[1]])

#funcion_MCO_beta1(X,y_in)
#funcion_MCO_ee_beta1(X,y_in)

```

2.2) Utilizando un loop, apliquen esta función a los diferentes outcomes en las listas de outcomes_nominales, guarden los coeficientes estimados y las desviaciones estándar en una matriz donde la primera columna corresponde al nombre del outcome, la segunda columna al coeficiente estimado (β_1) y la tercera al error estándar ($\sigma\beta$). En esta matriz, cada fila representará una estimación.

```
#Se hace un loop en el que se concatenan Los nombres de Las  
#variables, Los estimadores de La regresión por MCO y sus  
#errores estándar
```

```
#Se crea una matriz vacía con La dimensión deseada:  
# La cantidad de filas igual a La cantidad de resultados  
# La cantidad de columnas igual a 3: nombre, beta y error estándar
```

```
matriz_Estimadores <- matrix(NA,nrow = length(outcomes_nominales),  
                             ncol = 3)
```

```
for (i in 1:length(outcomes_nominales)){
```

```
#Input
```

```
# Funciones de estimación de betas y de errores estándar  
# Vectores de Los outcomes
```

```
beta <- funcion_MCO_beta1(matriz_edad_estandar,  
                           matrix(outcomes_nominales[[i]]))
```

```
error_est <- funcion_MCO_ee_beta1(matriz_edad_estandar,  
                                   matrix(outcomes_nominales[[i]]))
```

```
#Output
```

```
matriz_Estimadores [i,1] <- names(outcomes_nominales[i])  
matriz_Estimadores [i,2] <- round(beta,3)  
matriz_Estimadores [i,3] <- round(error_est,3)
```

```
}
```

```
print(matriz_Estimadores)
```

```
##      [,1]      [,2]      [,3]  
## [1,] "Salario"  "-0.064" "0.144"  
## [2,] "IndiceSalud" "0.052" "0.144"  
## [3,] "ExpLaboral" "-0.045" "0.144"
```

```
# Para probar que está bien Los errores estándar y Los betas
```

```
x = matriz_edad_estandar
```

```
y1 = c(matrix(outcomes_nominales[[1]]))
```

```
y2 = c(matrix(outcomes_nominales[[2]]))
```

```

y3 = c(matrix(outcomes_nominales[[3]]))

datos <- data.frame(x, y1, y2, y3)

modelo1 = lm(y1 ~ -1 + x, data = datos)
modelo2 = lm(y2 ~ -1 + x, data = datos)
modelo3 = lm(y3 ~ -1 + x, data = datos)

Mod1 = summary(modelo1)
Mod2 = summary(modelo2)
Mod3 = summary(modelo3)

Mod1[["coefficients"]]
##
##              Estimate Std. Error    t value Pr(>|t|)
## xEdad estándar -0.06359524  0.1628226  -0.3905799 0.6978359
## xConstante      0.11225789  0.1611862   0.6964487 0.4895069

Mod2[["coefficients"]]
##
##              Estimate Std. Error    t value Pr(>|t|)
## xEdad estándar  0.05220534  0.1345313  0.3880535 0.6996922
## xConstante      0.01756206  0.1331792  0.1318679 0.8956400

Mod3[["coefficients"]]
##
##              Estimate Std. Error    t value Pr(>|t|)
## xEdad estándar -0.04515376  0.1392506  -0.3242626 0.7471481
## xConstante     -0.09558756  0.1378511  -0.6934119 0.4913930

??summary

## starting httpd help server ... done

```

2.3) Haciendo uso de un loop hagan un print para que, automáticamente y para cada outcome, se realice la interpretación econométrica de cada coeficiente de regresión que estimaron. Recuerden tener en cuenta las distribuciones de las variables para su interpretación.

```

for (i in 1:length(outcomes_nominales)){

  print(paste("Por cada incremento de una desviación estándar de la
edad, la variable",matriz_Estimadores[i,1],"cambia
en",matriz_Estimadores[i,2],"desviaciones estándar"))
}

## [1] "Por cada incremento de una desviación estándar de la edad, la
variable Salario cambia en -0.064 desviaciones estándar"
## [1] "Por cada incremento de una desviación estándar de la edad, la
variable IndiceSalud cambia en 0.052 desviaciones estándar"
## [1] "Por cada incremento de una desviación estándar de la edad, la
variable ExpLaboral cambia en -0.045 desviaciones estándar"

```

```
#for(interpretación in 1:length(matrizest[1,])){

  # print()

#}
```

2.4) Creen una función que calcule el $MSE(y_i, x_i, \beta_0, \beta_1)$. Es decir, que tenga como input un vector de y_i , un vector/lista de x_i y los parámetros β_0 y β_1 . Como output debe proveer un escalar correspondiente al MSE de esa combinación de inputs.

#Se crea una función para determinar el MSE

```
funcion_MSE <- function(y_in,X_in,beta0,beta1){

  #Input
  #y_in: Un vector del resultado
  #X_in: Una matriz de Los valores independientes
  #beta0: Valor de la constante de la regresión lineal simple
  #beta1: Estimador de la regresión para la variable independiente

  #Output
  #MSE: Error al cuadrado promedio

  MSE <- sum((y_in - beta0 - beta1*X_in)^2)/length(y_in)

}

funcion_MSE(matrix(outcomes_nominales[[1]]),
              matrix(matriz_edad_estandar[,1]),1,1)
```

2.5) Utilizando un ciclo while generen una función que retorne el coeficiente β_{min}^1 que minimiza el error de ajuste de los datos de forma numérica.

```
funcion_Min_MSE <- function(y_in,X_in,beta) {

  #Valores iniciales

  i <- -1 #Diferencia entre MSE
  beta_1 <- -5 #Valor inicial de beta
  paso_iteracion <- 0.001 #El incremento de beta_1 por iteración
  MSE_t_1 <- funcion_MSE(y_in,X_in,0,(beta_1-paso_iteracion)) #Valor
  inicial de MSE, como es la inicial decidimos usar beta con un paso menos

  while(i<0){

    #Se utiliza la función MSE del punto anterior

    invisible(MSE <- funcion_MSE(y_in,X_in,0,beta_1))
```

```

#Se calcula la diferencia entre el MSE actual y el de la iteración
#anterior
i <- MSE - MSE_t_1

#Se actualiza el valor del MSE para pasar a la siguiente iteración

MSE_t_1 <- MSE

#Se incrementa el valor de beta en un valor constante dado por
#el valor del paso de la iteración
beta_1 <- beta_1 + paso_iteracion

}
beta_1
}

```

2.6) Apliquen las funciones desarrolladas en los puntos anteriores para encontrar el β^{\min}_1 que minimiza el error cuadrático medio por métodos numéricos, suponiendo que $\beta_0=0$, para los outcomes_nominales. Reporten sus resultados en una matriz con dos columnas: Nombre del outcome y $[\beta^{\min}]_1$. Esta matriz debe tener 3 filas, una por cada outcome.

```

matriz_Est_numerico <- matrix(NA,nrow = length(outcomes_nominales),
                              ncol = 2)

for (i in 1:length(outcomes_nominales)){

  #input
  #vector de y_in
  #matriz de X
  #beta_0 = 0

  #Se hace un loop para que abarque todos los outcomes

  matriz_Est_numerico [i,1] <- names(outcomes_nominales[i])
  matriz_Est_numerico [i,2] <- round(funcion_Min_MSE(
    matrix(outcomes_nominales[[i]]),
    matrix(matriz_edad_estandar[,1]),0),3)

}
matriz_Est_numerico

##      [,1]      [,2]
## [1,] "Salario"  "-0.062"
## [2,] "IndiceSalud" "0.054"
## [3,] "ExpLaboral" "-0.043"

```