Introducción a R

Andrés Vallone

Escuela de Ciencias Empresariales

2019

Contenidos

UCN CIENCIAS EMPRESARIALES

Repaso de la sesión anterior

Programación Intermedia en R

Aplicando todo lo aprendido

Repaso de la sesión anterior

Repaso de la sesión anterior

- Los paquetes son un conjunto de funciones y su documentación.
- Deben instalarse y ser llamados para su uso.
- Podemos utilizar paquetes en Desarrollo mediante el uso del paquete `devtools`
- Es posible trabajar con diferentes tipos de formato de archivo en R.
- La función `subset` y a indexación ayuda a filtrar los datos para trabajar.

Programación Intermedia en R

¿Cuál es el numero mas grande?

UCN CIENCIAS EMPRESARIALES

¿Cuál es el numero mas grande?

37	174	23	29	85	22	133	31	96	152	98	35
43	52	163	75	124	178	113	9	146	65	144	142
85	165	45	168	1	165	50	23	8	57	131	127
165	171	53	129	174	18	117	106	109	123	109	162
39	133	58	69	145	145	23	145	169	39	123	46
152	140	101	29	171	179	134	11	191	77	2	165
22	95	25	115	78	95	187	122	122	128	18	61
43	158	59	192	14	16	84	141	6	127	8	184
149	124	39	56	186	91	21	162	152	153	146	175
7	8	69	3	167	112	42	37	140	174	174	176

¿Cuál es el número más grande?

37	174	23	29	85	22	133	31	96	152	98	35
43	52	163	75	124	178	113	9	146	65	144	142
85	165	45	168	1	165	50	23	8	57	131	127
165	171	53	129	174	18	117	106	109	123	109	162
39	133	58	69	145	145	23	145	169	39	123	46
152	140	101	29	171	179	134	11	191	77	2	165
22	95	25	115	78	95	187	122	122	128	18	61
43	158	59	192	14	16	84	141	6	127	8	184
149	124	39	56	186	91	21	162	152	153	146	175
7	8	69	3	167	112	42	37	140	174	174	176

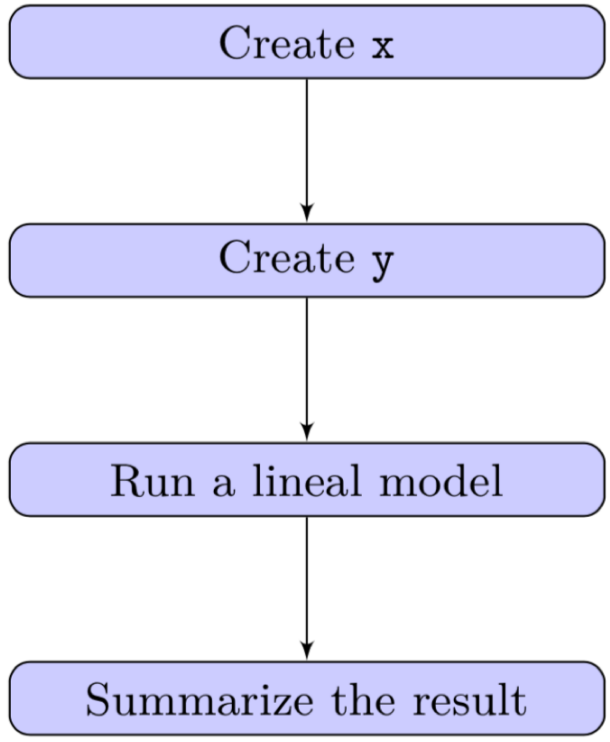
Pensado como computadoras

UCN CIENCIAS EMPRESARIALES

- Las computadores son eficientes, pero no flexibles. Ellas solo siguen reglas.
- Un `script` es un conjunto de reglas que se le entrega a la computadora.
- Al ejecutar un `script` la computadora ejecuta linea a linea el código que se le entrega.
- Un `script`puede contener tres tipos de lineas:
  - *Secuencias:* una orden de ejecución directa (ej. la asignación de un objeto)
  - *Condicionales:* evalúa una condición verdadero (`TRUE`) o falsa (`FALSE`) y ejecuta o no una acción (ej `if(gender == "male")`)
  - *Repeticiones:* repite una acción una y otra vez hasta que se cumpla una condición de corte (ej. 'while(z $<$ 5)' { do this})

Secuencias

```
x <- rnorm(100,10,1)
y <- rnorm(100,4,3)
model <- lm(y ~ x)
summary(model)
```



Condicionales I

- Las condiciones `if`
  - Ejecutan con conjunto de operaciones dependiendo de del cumplimiento de una condición `LOGICAL`
  - Evalué está condición y realice lo siguiente:

Uso

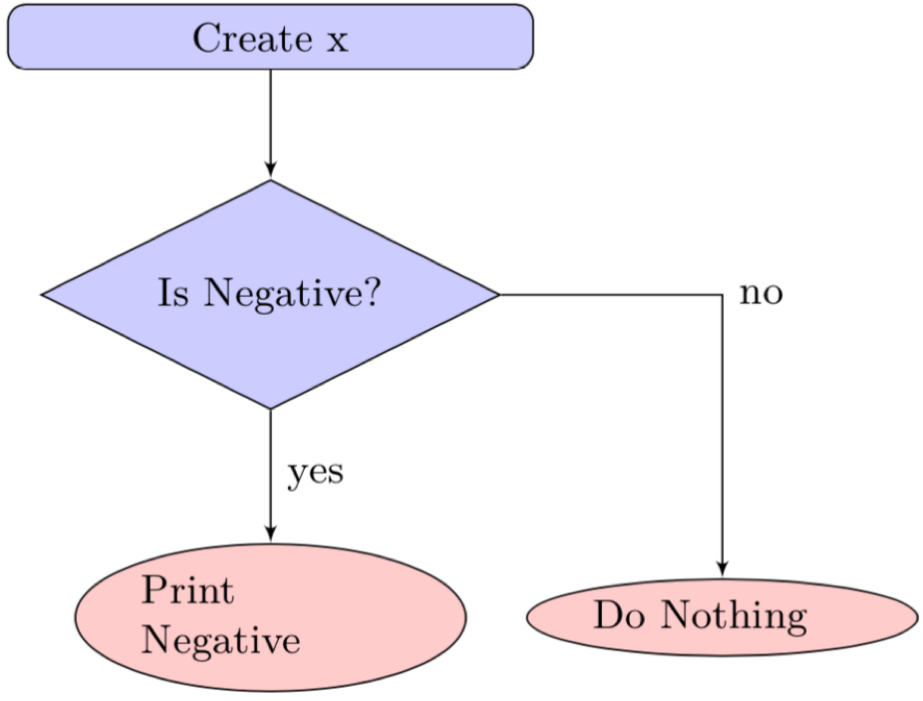
```
if(condition){
        expr
}
```

Condicionales 1

UCN CIENCIAS EMPRESARIALES

if

```
x <- -3
if (x < 0){
        print("Negative")
        }
```



- Solo si `x < 0`el resultado será `"Negative"`, en cualquier otro caso no hará resultado

Condicionales 2

UCN CIENCIAS EMPRESARIALES

- Los condicionales `if else`
  - Ejecutan con conjunto de operaciones dependiendo de del cumplimiento de una condición `LOGICAL`
  - Permite agregar una segunda expresión
  - Evalué, entonces hace esto o esto otro

Uso

```
if(condition){
            expr1
            } else {
            expr2
            }
```

Una forma abreviada

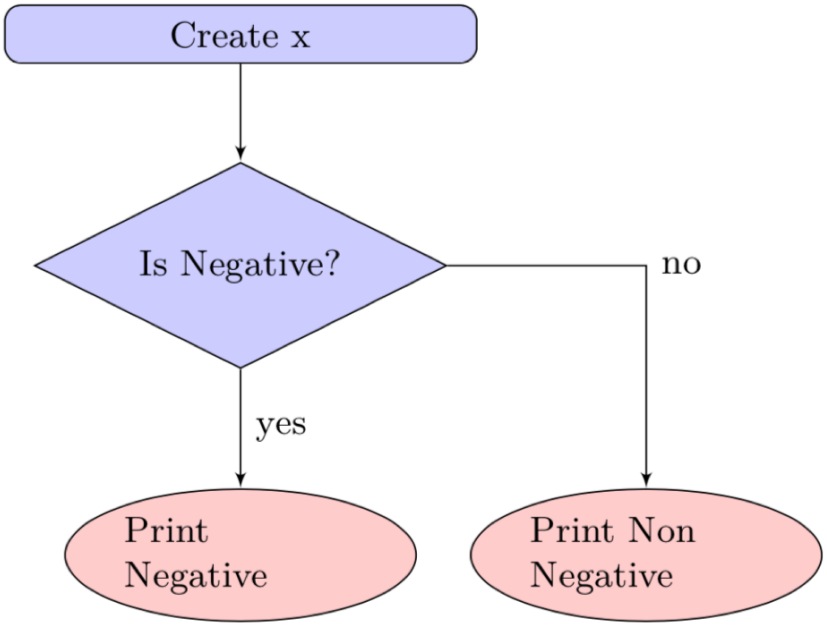
```
ifelse(condition,expr1,expr2)
```

Condicionales 2

UCN CIENCIAS EMPRESARIALES

if else

```
x <- -3
if (x < 0){
        print("Negative")
        } else {
        print("Non-negative")
        }
```



- Solo si `x < 0`el resultado será `"Negative"`, y `"Non-negative"`en cualquier otro caso ‘

Condicionales 3

UCN CIENCIAS EMPRESARIALES

- Los condicionales `else if`
  - Ejecutan con conjunto de operaciones dependiendo de del cumplimiento de una condición `LOGICAL`
  - Permite agregar múltiples expresiones
  - Evalué, entonces hace esto o esto otro o esto otro o...

Uso

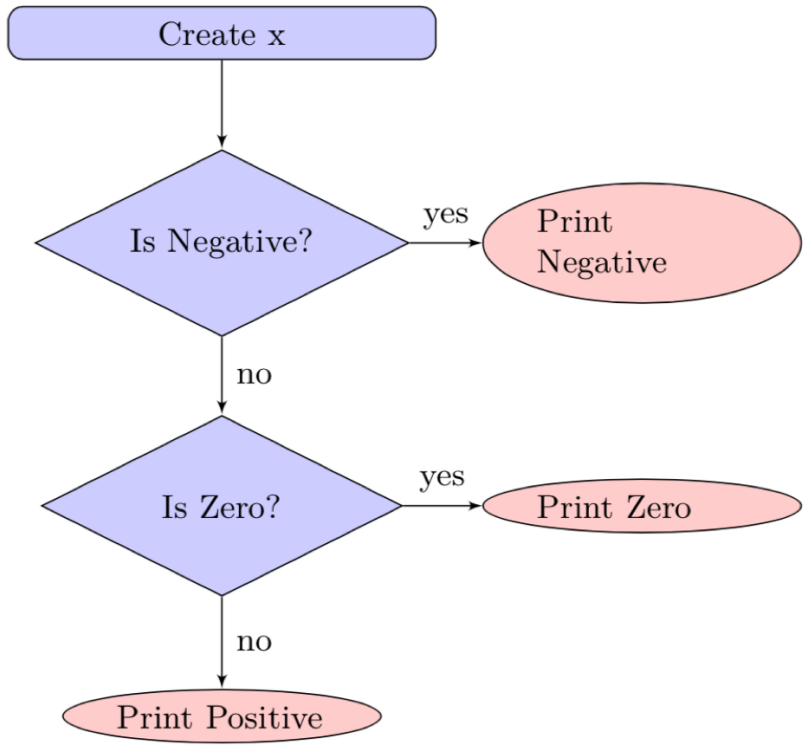
```
if(condition){
            expr1
            } else if {
            expr2
            } else {
            expr3
}
```

Condicionales 3

UCN CIENCIAS EMPRESARIALES

else if

```
x <- -3
 if (x < 0){
          print("Negative")
          } else if (x == 0) {
          print("Zero")
          } else {
          print("Poitive")
          }
```



- El código se detiene en la primer condición verdadera (`TRUE`) ejecutando la expresión asociada e ignorando el resto del código.

Cuidado!!!!!

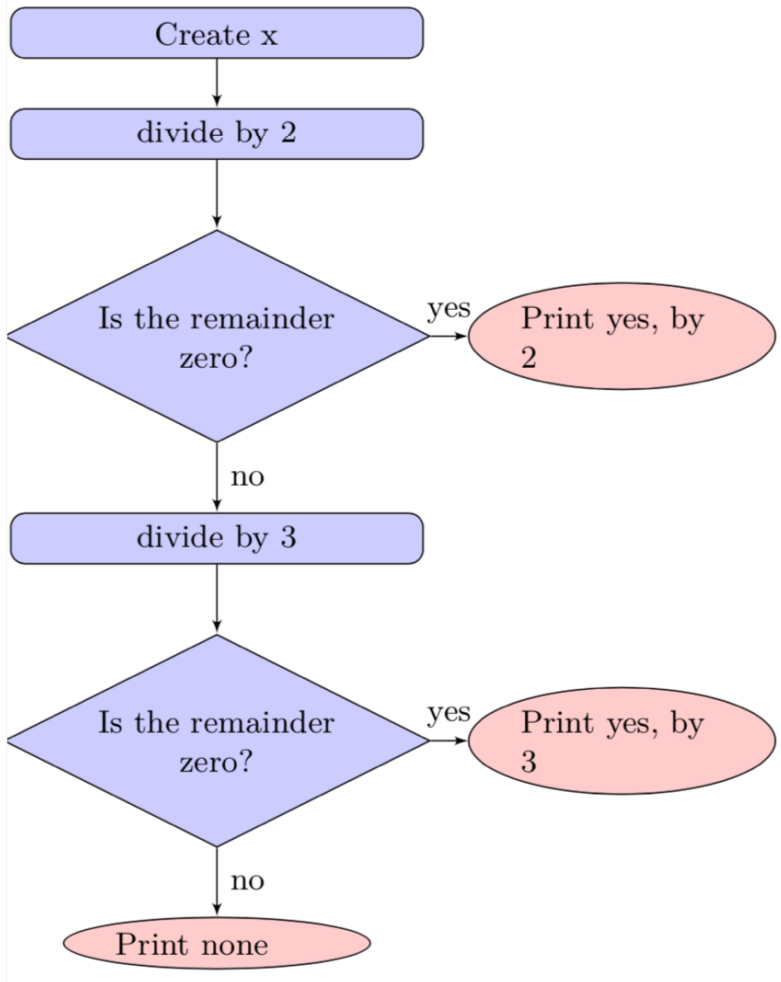
Asegúrese de establecer condiciones mutuamente excluyentes

Condicionales 3

- ¿Es el numero divisible por 2 o por 3?

else if

```
x <- 2
 if (x %% 2 == 0){
          print("yes, by 2")
          } else if (x %% 3 == 0) {
          print("yes, by 3")
          } else {
          print("none")
          }
```



- ¿qué sucede si 'x<-6'?

Repeticiones (Loops)

UCN CIENCIAS EMPRESARIALES

- Un `lopp` es una estructura de código que se ejecuta reiteradamente una cantidad de veces.
- Puede ser o no definido para que se repita un numero fijo de veces o no.
- Los loop basados en la función `for()` requieren una secuencia o variable variable de iteración las cuales controlan el numero de iteraciones.
- Un loop basado en la función `while()`requiere una condición de freno, el numero de iteraciones depende del cumplimiento de esta condición.
- Existen un conjunto de funciones que realizan loops implícitos, estas son parte de la familia de la función `apply`

Loops basados en `while`

Uso

```
while (condition){
        statement
}
```

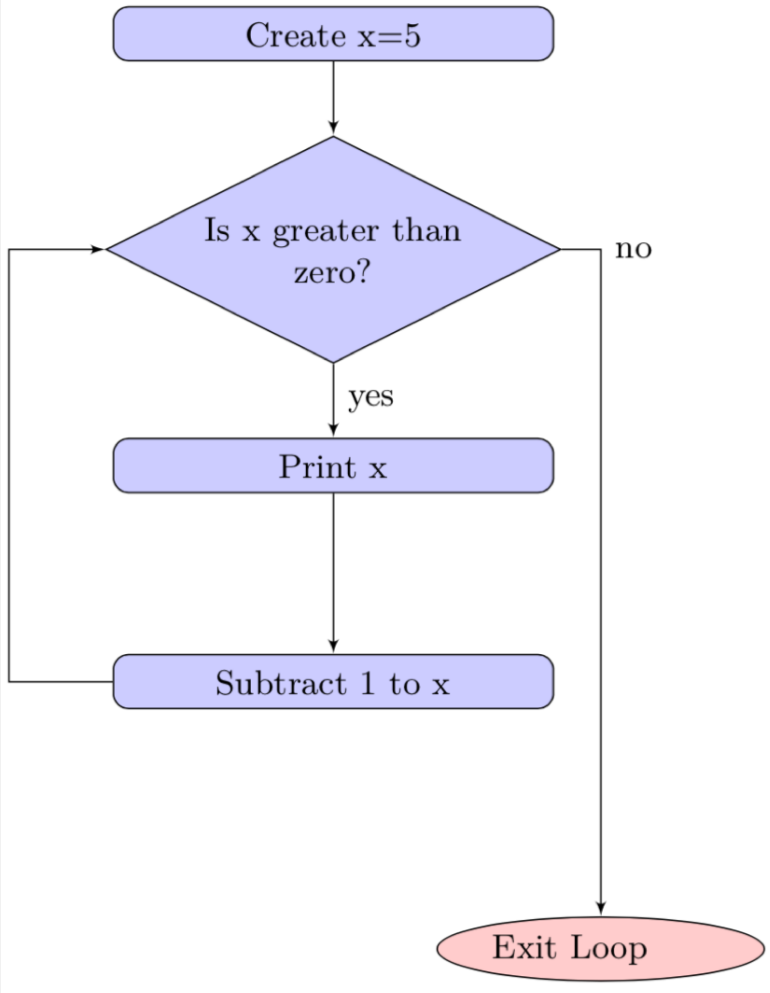
- Este tipo de loops repite un conjunto de acciones hasta que la `condition`se cumple
- El `*cuerpo*` (statment‘) contiende las lineas de código que se ejecutarán en cada iteración

Ejemplo de uso de `while`

UCN CIENCIAS EMPRESARIALES

Ejemplo de while

```
x <- 5
while(x > 0){
        print(x)
        x <- x - 1
        }
```



- ¿Qué sucedería si se usara en el cuerpo `x = x + 1`?

Ejemplo de uso de `while`

UCN CIENCIAS EMPRESARIALES

`while`combinado con `break`

```
x <- 5
while(x > 0){
        print(x)
        x <- x + 1
        if (x == 16) break
        }
```

- `break` permite interrumpir la ejecución del loop

Loops basados en `for`

Uso

```
for (sequence){
            statement
}
```

- Una secuencia debe contener dos elementos, el nombre del objeto de la iteración y los valores de este índice para iterar (ej: `for( i in 1:20)`)
- El `*cuerpo*` (statment') contiende las lineas de código que se ejecutarán en cada iteración
- La mayor parte del tiempo un loop `for` requiere un objeto para guardar los resultas del cuerpo.

Ejemplo del uso de `for`

```
A <- matrix(rnorm(100),10,10)
col.means <- c()
for (i in 1:nrow(A)){
       col.means[i] <- mean(A[,i])
     }
```

- el valor de ìes remplazado en cada iteración por uro de los valores de la secuencia.

Ejemplo del uso de `for`

- Los loops basados en 'for' se pueden anidar

`for` anidado

```
A <- matrix(runif(100,100,500),10,10)
s.A <- matrix(0,10,10)
for(j in 1:ncol(A)){
      Mean <- mean(A[,j])
      st.dev <- sd(A[,j])
      for (i in 1:nrow(A)){
            s.A[i,j] <- (A[i,j]-Mean)/st.dev
      }
  }
```

Uso de `for`con control de ejecución

UCN CIENCIAS EMPRESARIALES

Control de secuencia con `next`

```
A <- matrix(runif(100,100,500),10,10)
A.d.2 <- A
for(j in 1:ncol(A)){
      for (i in 1:nrow(A)){
                if (i!=j) next
                A.d.2[i,j] <- (A[i,j])^2
      }
  }
```

- `next` permite evitar que se itere para condiciones particular saltando al siguiente paso de la iteración.
- cuando se cumple la condición, se salta de ejecución del loop y se pasa al siguiente caso.

Uso de `for`con control de ejecución

UCN CIENCIAS EMPRESARIALES

Control de secuencia con `break`

```
A <- matrix(0,10,10)
for(i in 1:ncol(A)){
    for (j in 1:nrow(A)){
          if (i==j){
                break
              } else {
                A[j,i] <- i*j
              }
      }
  }
```

- breaedetiene *solo* el loop interior

La familia de funciones `apply`

- La familia de funciones `apply()` pertenece al paquete base de R, están pensadas para manipular los margenes de estructuras complejas como listas, matrices y data frames.
- Estas funciones permiten aplicar una función en distintas direcciones de los objetos sin realizar un loop explicito.
- Las funciones `apply()` son la base de combinaciones más complejas y ayudan a realizar operaciones con muy pocas líneas de código.- De forma especifica la familia está compuesta por las funciones: `apply()`, `lapply()` , `sapply()`, `vapply()`, `mapply()`, `rapply()`, y `tapply()`

La función apply

Uso

```
apply(X, MARGIN, FUN, ...)
```

- `X` es una matriz o un data frame
- `MARGIN` define la dirección en la que se aplica la función, si es igual a `1` se aplica sobre las filas y si es 2 se aplica sobre las columnas.
- `FUN`es la función a aplicar.
- `...` recoge los argumentos extras que requiere la función a aplicar.

Ejemplo

```
A <- matrix(rnorm(100),10,10)
col.means <- apply(A,2,mean)
```

La función apply

Uso

```
apply(X, MARGIN, FUN, ...)
```

- `X` es una matriz o un data frame
- `MARGIN` define la dirección en la que se aplica la función, si es igual a `1` se aplica sobre las filas y si es 2 se aplica sobre las columnas.
- `FUN`es la función a aplicar.
- `...` recoge los argumentos extras que requiere la función a aplicar.

Casos especiales

```
colSums (x, na.rm = FALSE)        colMeans(x, na.rm = FALSE)
rowSums (x, na.rm = FALSE)        rowMeans(x, na.rm = FALSE)
```

Las funciones `sapply` y `lapply`

Uso

```
lapply(X, FUN, ...)
sapply(X, FUN, ...)
```

- `X` es lista, una matriz o un data frame .
- `FUN`es la función a aplicar.
- `...` recoge los argumentos extras que requiere la función a aplicar.
- Output
  - `lapply` entrega una lista del mismo tamaño de la lista a la cual se aplica la función.
  - `sapply`simplifica la salida al objeto menos complejo posible.

Ejemplos de sapply y lapply

UCN CIENCIAS EMPRESARIALES

```r
data <-list(A=matrix(1:25,5,5), B=matrix(letters[1:25],5,5))
#usando lapply
col.data <- lapply(data,'[',,2)

#usando sapply
col.data <- sapply(data,'[',,2)

#un uso util de sapply para multiples data frame
dim.data <- sapply(data,dim)

# Unn buen uso para lapply
pkg <- c("ggplot2", "spdep" ,"rgdal","rvest")
lapply(pkg,require, character.only = TRUE)
```

Aplicando todo lo aprendido

Un experimento simple en R

UCN CIENCIAS EMPRESARIALES

- Escriba un `script` en R que muestre que en la estimación de un modelo del tipo $y = \beta_0 + \beta_1 x + u$ cuando $u \sim (0, 1)$ se cumple la condición $E[\hat{\beta}_1] = \beta_1$

```
#set the iteration number
n <- 10000
# set the sample size
s <- 1000
# set a seed for replicability
set.seed(111)
# set betas values
b1 <- 4 ; b2 <- 2
# create the x variable
x <- runif(s,200,400)
b2_e <- rep(0,n)
for (j in 1:n){
    e <- rnorm(s)
    y <- b1+b2*x+e
    b2_e[j] <- lm(y~x)$coefficients[2]
}
hist(b2_e); mean(b2_e)
```

Tarea 1.

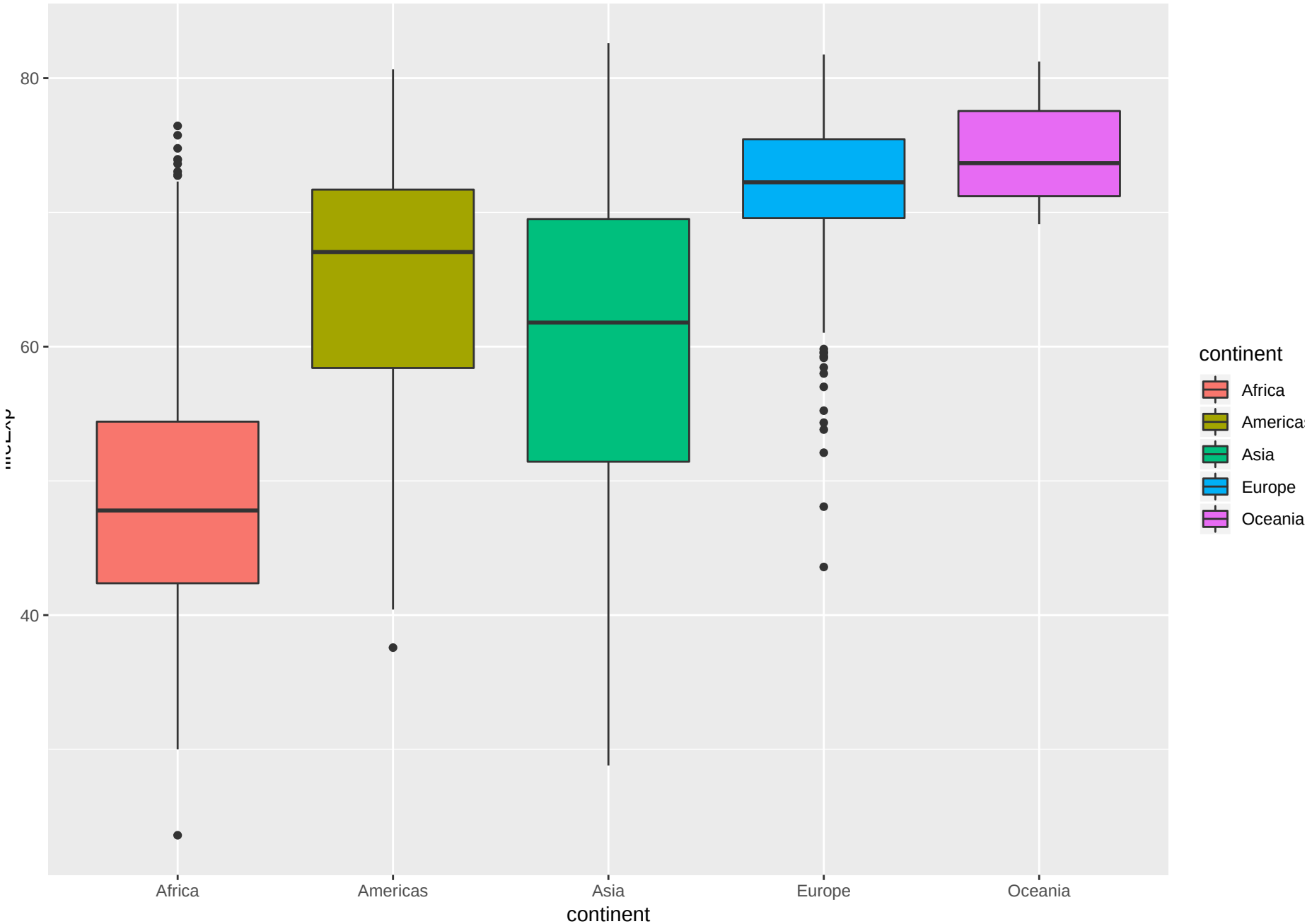
- Escriba un `script` que permita ver que la suma de dos dados sigue una distribución normal.

Analizando la base gapminder

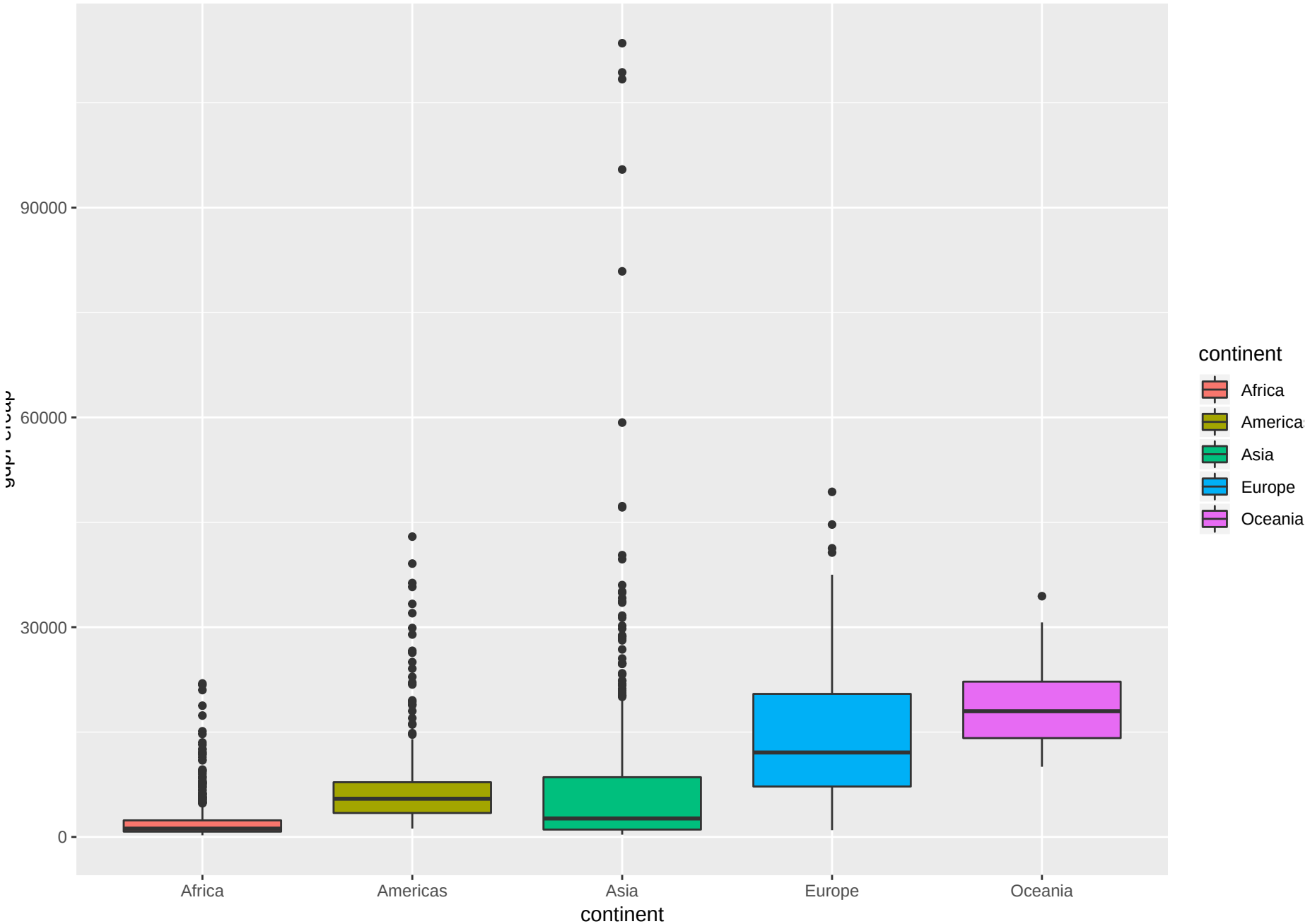
UCN CIENCIAS EMPRESARIALES

```
library(ggplot2)
library(gapminder)
# Varia la esperanza de vida entre continentes?
ggplot(data = gapminder, aes(x = continent, y = lifeExp))
               + geom_boxplot(aes(fill=continent))
# Varia PIB per cápita entre continentes?
ggplot(data = gapminder, aes(x = continent, y = gdpPercap))
               + geom_boxplot(aes(fill=continent))
# Hay alguna relación entre ambas variables?
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp))
               + geom_point(aes(color=continent))

ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp))
                + geom_point(aes(color=continent)) + scale_x_log10()

ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp))
               + geom_point(aes(color=continent)) + scale_x_log10()
               + geom_smooth(method="lm")
```

Analizando la base gapminder



Analizando la base gapminder



Analizando la base gapminder

Analizando la base gapminder

Analizando la base gapminder

Tarea 2

- Utilizando la base de datos `gapminder`
  - Analice la elocución temporal de la media del PIB per capota de los continentes.
  - Calcule la regresión lineal entre Esperanza de vida y PIB per capota.
  - Reporte la estadística descriptiva por año de la esperanza de vida y PIB per capota.