Introducción

Conceptos básicos e introducción a R

R puede ser usado para realizar cálculos complejos. Un ejemplo sencillo de aritmética se muestra a continuación.

```
3+5
```

```
## [1] 8
```

```
2-10
```

```
## [1] -8
```

```
2*3
```

```
## [1] 6
```

```
2/5
```

```
## [1] 0.4
```

Además de los operadores básicos +,-, *, /, se tiene %% que entrega el resto de una división, por ejemplo,

```
5%%2
```

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

```
5%%5
```

```
## [1] 0
```

Por otra parte una potencia puede ser indicada mediante los operadores ^ o **

```
2^3
```

```
## [1] 8
```

```
2**3
```

```
## [1] 8
```

En R es posible realizar comparaciones lógicas mediante <,>,==,!=,<=,>=

```
3 > 5
```

```
## [1] FALSE
```

```
3 < 4
```

```
## [1] TRUE
```

```
2 == 2.1
```

```
## [1] FALSE
```

```
3 != 3
```

```
## [1] FALSE
```

```
3 >= 3
```

```
## [1] TRUE
```

R dispone de un sin fin de funciones f() que al ingresar un objeto e indicaciones, retornan un resultado. Algunos ejemplos sencillos con `log()`,`exp`.

```
log(2)
```

```
## [1] 0.6931472
```

```
log(2, base = 10)
```

```
## [1] 0.30103
```

```
exp(-2)
```

```
## [1] 0.1353353
```

```
exp(100)**10
```

```
## [1] Inf
```

Las variables categóricas en una encuesta de hogares se pueden ver como vectores de caracteres, los caracteres son indicados mediante las comillas `"ABCDE"`. La función `rep()` genera un vector que contiene el objeto repetido el número de veces indicado.

```
rep(3, times = 5)
```

```
## [1] 3 3 3 3 3
```

```
rep("A", times = 10)
```

```
##  [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
```

Objetos: Factores y vectores

R está enfocado en la programación orientada a objetos. La asignaciónde un valor a objeto se realiza con los caracteres `<-` o `=`, para la creación de vectores asociados a objetos es necesario el comando `c()`.

```
x <- 5
x + 2
```

```
## [1] 7
```

```
is(x)
```

```
## [1] "numeric" "vector"
```

```
y = "Hombre"
y
```

```
## [1] "Hombre"
```

En un vector es posible almacenar valores numéricos o caracteres, pero no ambos. En caso de incluir un número seguido de un caracter, el objeto numérico será considerado caracter.

```
vector1 = c(1,2,3,4,5)
vector1
```

```
## [1] 1 2 3 4 5
```

```
vector2 <- c("Hello","world","!")
vector2
```

```
## [1] "Hello" "world" "!"
```

```
vector3 = c(vector1,vector2)
vector3
```

```
## [1] "1"     "2"     "3"     "4"     "5"     "Hello" "world" "!"
```

Es posible realizar operaciones con cada objeto y vector definido. Para el caso de los vectores las operaciones se realizan sobre vectores del mismo largo pues se ejecutan componente a componente.

```
x1 <- c(1,2,3,4)
x2 <- c(10, 10, 100, 1000)
5 * x1
```

```
## [1]  5 10 15 20
```

```
x1 + x2
```

```
## [1]   11   12  103 1004
```

```
x1 / x2
```

```
## [1] 0.100 0.200 0.030 0.004
```

Mediante las funciones `sum()`,`prod()`, etc. es posible realizar operaciones numéricas con los elementos del vector.

```
sum(x1)
```

```
## [1] 10
```

```
sum(x2)
```

```
## [1] 1120
```

```
sum(x1)+sum(x2)
```

```
## [1] 1130
```

```
prod(x1)
```

```
## [1] 24
```

```
prod(x1)-prod(x2)
```

```
## [1] -9999976
```

Otros ejemplos para la creación de vectores

```
b <- c(2:20)
a <- c(1:3)
rep(a,4)
```

```
##  [1] 1 2 3 1 2 3 1 2 3 1 2 3
```

```
seq(1,10, by = 2)
```

```
## [1] 1 3 5 7 9
```

```
seq(0,1, by = 0.1)
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
seq(0,1, length.out = 13)
```

```
##  [1] 0.00000000 0.08333333 0.16666667 0.25000000 0.33333333 0.41666667
```

```
##  [7] 0.50000000 0.58333333 0.66666667 0.75000000 0.83333333 0.91666667
## [13] 1.00000000
```

```
d <- c(3,-1,0,4)

sort(d)
```

```
## [1] -1  0  3  4
```

```
order(d)
```

```
## [1] 2 3 1 4
```

Componentes de objetos

Definido un objeto, es posible extraer los elementos que uno desee mediante diferentes comandos.

```
a <- c(-3:11)
a
```

```
##  [1] -3 -2 -1  0  1  2  3  4  5  6  7  8  9 10 11
```

```
a[4]
```

```
## [1] 0
```

```
a[-2]
```

```
##  [1] -3 -1  0  1  2  3  4  5  6  7  8  9 10 11
```

```
a[c(1,6,7)]
```

```
## [1] -3  2  3
```

```
a > 0
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE
```

```
a[a>0]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11
```

Operadores Boleanos

1. Disyunción: `|` Una proposición es verdadera si alguno de sus componentes lo es.
2. Conjunción: `&` Una proposición es verdadera si y sólo si ambas componentes lo son,

```
x <- c(1:10)
x[(x>8)|(x<5)]
```

```
## [1]  1  2  3  4  9 10
```

```
x[(x>8)&(x<5)]
```

```
## integer(0)
```

Factores para variables categóricas

En R la creación de factores hacen explícita las categorías de las variables discretas.

```
sexo <- c("Hombre", "Hombre", "Mujer", "Mujer", "Hombre", "Mujer")
sexo
```

```
## [1] "Hombre" "Hombre" "Mujer"  "Mujer"  "Hombre" "Mujer"
```

```
sexo1 <- as.factor(sexo)
sexo1
```

```
## [1] Hombre Hombre Mujer  Mujer  Hombre Mujer
## Levels: Hombre Mujer
```

```
sexo1 == "Hombre"
```

```
## [1]  TRUE  TRUE FALSE FALSE  TRUE FALSE
```

```
which(sexo1=="Hombre")
```

```
## [1] 1 2 5
```

Mediante la función `summary()` se obtiene información resumida del objeto.

```
summary(sexo1)
```

```
## Hombre  Mujer
##      3      3
```

Objetos: archivos de datos

Un archivo de datos es un tipo de objeto que define agrupación de variables que pueden ser numéricas, categóricas o mixtas. Las filas de un `data.frame()` corresponden a los registros individuales de las unidades de observación mientras que las columnas corresponde a las variables medidas de los individuos.

A continuación crearemos algunas variables para la formación de un archivo de datos. Generamos un identificador mediante la función `paste0()` la cuál crea un objeto con caracteres que une los vectores indicados

```
# Identificación
id <- paste0("CEPALIDk", c(1:5))
id
```

```
## [1] "CEPALIDk1" "CEPALIDk2" "CEPALIDk3" "CEPALIDk4" "CEPALIDk5"
```

```
# Otras variables de interés
ingreso <- c(450, 500, 250, 1000, 500)
gasto <- 100 + ingreso * 0.25
zona <- c(rep("urbano", 3), rep("rural", 2))
sexo <- c(rep("hombre", 2), rep("mujer", 3))
```

La función `data.frame()` crea un archivo de datos.

```
datos <- data.frame(ID = id, INC = ingreso, EXP = gasto, ZON = zona, SEX = sexo)
datos
```

```
##          ID  INC   EXP    ZON    SEX
## 1 CEPALIDk1  450 212.5 urbano hombre
## 2 CEPALIDk2  500 225.0 urbano hombre
```

```
## 3 CEPALIDk3  250 162.5 urbano  mujer
## 4 CEPALIDk4 1000 350.0  rural  mujer
## 5 CEPALIDk5  500 225.0  rural  mujer
```

Otras funciones útiles a considerar para el manejo de `data.frame()`.

1. `head()` imprime los primeros seis registros de la base.
2. `tail()` imprime los últimos seis registros de la base.
3. `names()` muestra los nombres de la base de datos.
4. `str()` muestra la estructura de las variables de la base de datos.

Una comando de importancia son las llaves y la coma [,] la cuál se utiliza para seleccionar casos particulares o variables particulares desde un conjunto de datos. 1. Para seleccionar casos [casos,] 1. Para seleccionar variables [, variables] 1. Para seleccionar casos y variables[casos, variables]

```
datos[1,]
```

```
##          ID INC   EXP    ZON    SEX
## 1 CEPALIDk1 450 212.5 urbano hombre
```

```
datos[,3]
```

```
## [1] 212.5 225.0 162.5 350.0 225.0
```

```
datos[c(1,4),]
```

```
##          ID  INC   EXP    ZON    SEX
## 1 CEPALIDk1  450 212.5 urbano hombre
## 4 CEPALIDk4 1000 350.0  rural  mujer
```

```
datos[,c(2,5)]
```

```
##    INC    SEX
## 1  450 hombre
## 2  500 hombre
## 3  250  mujer
## 4 1000  mujer
## 5  500  mujer
```

```
datos[c(1,4),c(2,5)]
```

```
##    INC    SEX
## 1  450 hombre
## 4 1000  mujer
```

Para trabajar con variables específicas de una base de datos se utiliza el operador `$`.

```
datos$ID
```

```
## [1] CEPALIDk1 CEPALIDk2 CEPALIDk3 CEPALIDk4 CEPALIDk5
## Levels: CEPALIDk1 CEPALIDk2 CEPALIDk3 CEPALIDk4 CEPALIDk5
```

```
datos$EXP
```

```
## [1] 212.5 225.0 162.5 350.0 225.0
```

Este operador permite añadir tantas variables como sea requerido a la `data.frame()`, por otro lado permite crear nuevas variables mediante la operación de otras variables.

```
datos$ESC <- c("Lee", "Lee", "Lee", "No lee", "Lee")
datos$IOE <- datos$INC/datos$EXP
datos
```

```
##          ID  INC   EXP    ZON    SEX    ESC      IOE
## 1 CEPALIDk1  450 212.5 urbano hombre    Lee 2.117647
## 2 CEPALIDk2  500 225.0 urbano hombre    Lee 2.222222
## 3 CEPALIDk3  250 162.5 urbano  mujer    Lee 1.538462
## 4 CEPALIDk4 1000 350.0  rural  mujer No lee 2.857143
## 5 CEPALIDk5  500 225.0  rural  mujer    Lee 2.222222
```

Librerias en R

Una librería de R es un conjunto de funciones y bases de datos encapsuladas en un objeto.

1. `install.packages("paquete")`: Descarga del paquete
2. `library(paquete)`: Uso de un paquete
3. `help(paquete)`: Ayuda e información del paquete

Teaching sampling

La librería TeachingSampling fue desarrollada para seleccionar muestras probabilísticas bajo diferentes esquemas de muestreo en varias etapas. De la misma manera, puede ser usada para realizar los procesos apropiados de estimación de varianzas, entre otros.

```
install.packages("TeachingSampling")
library("TeachingSampling")
citation("TeachingSampling")
```

samplesize4surveys

La librería samplesize4surveys fue creada con el fin de permitirle al investigador evaluar rápidamente los escenarios apropiados para la determinación del tamaño de muestra en una encuesta de hogares.

```
install.packages("samplesize4surveys")
library("samplesize4surveys")
citation("samplesize4surveys")
```

La función `Domains()`

```
library(TeachingSampling)
help("Domains")
?Domains

z <- c("Jefe","hijo","Jefe","Conyuge")
Domains(z)
```

```
##      Conyuge hijo Jefe
## [1,]       0    0    1
## [2,]       0    1    0
## [3,]       0    0    1
## [4,]       1    0    0
```

Las librerias incluyen bases de datos que se pueden acceder mediante la función `data("Nombre")`

```r
data(BigCity)
names(BigCity)
```

```
##  [1] "HHID"        "PersonID"    "Stratum"     "PSU"         "Zone"
##  [6] "Sex"         "Age"         "MaritalST"   "Income"      "Expenditure"
## [11] "Employment"  "Poverty"
```

```r
head(BigCity)
```

```
##        HHID PersonID   Stratum     PSU  Zone    Sex Age MaritalST Income
## 1 idHH00001  idPer01 idStrt001 PSU0001 Rural   Male  38   Married 555.00
## 2 idHH00001  idPer02 idStrt001 PSU0001 Rural Female  40   Married 555.00
## 3 idHH00001  idPer03 idStrt001 PSU0001 Rural Female  20    Single 555.00
## 4 idHH00001  idPer04 idStrt001 PSU0001 Rural   Male  19    Single 555.00
## 5 idHH00001  idPer05 idStrt001 PSU0001 Rural   Male  18    Single 555.00
## 6 idHH00002  idPer01 idStrt001 PSU0001 Rural   Male  35   Married 298.34
##   Expenditure Employment  Poverty
## 1      488.33   Employed  NotPoor
## 2      488.33   Employed  NotPoor
## 3      488.33   Inactive  NotPoor
## 4      488.33   Employed  NotPoor
## 5      488.33   Inactive  NotPoor
## 6      216.70   Employed Relative
```

Creación de bases de datos mediante libreria `dplyr`

Mediante la siguiente sintaxis generamos las variables para la fuerza de trabajo

```r
library(dplyr)
# Creación base de UPMS
BigCity$Ocupados = ifelse(BigCity$Employment == "Employed", 1, 0)
BigCity$Desocupados = ifelse(BigCity$Employment=="Unemployed", 1, 0)
BigCity$FT = BigCity$Ocupados + BigCity$Desocupados
```

Con el comando `%>%` indexamos un objeto a la función deseada, es decir, si `x` es el objeto cargado en el entorno de `r` y deseamos aplicarle una funcion `funcion(x)` podemos equivalentemente escribir `x %>% funcion()`.

Finalmente con la libreria utilizamos la función `group_by()` para agrupar las variables a nivel `PSU` y `HHID` luego `summarise()` para calcular los resumenes agrupados

```r
UPMS = BigCity %>% group_by(PSU) %>%
                   summarise(Hogares = length(unique(HHID)), # Nro de hogares en la UPM
                             Personas = n(), # Nro de personas en la UPM
                             estratos = unique(Stratum), # Estrato a la que pertenece la UPM
                             area = unique(Zone), # Area de la UPM
                             hombres = mean(Sex == "Male"), # Proporción de hombres en la UPM
                             edad.media = mean(Age), # Media edad de la UPM
                             ingreso.medio = mean(Income), # Media ingreso de la UPM
                             gasto.medio = mean(Expenditure), # Media gasto de la UPM
                             Desempleo = sum(Desocupados)/sum(FT)) %>% # Tasa desempleo de la UPM
                    as.data.frame()
head(UPMS)
```

```
##       PSU Hogares Personas  estratos  area   hombres edad.media ingreso.medio
## 1 PSU0001      26      118 idStrt001 Rural 0.5254237   27.16949      600.9468
```

```
## 2 PSU0002      32      136 idStrt001 Rural 0.4705882   26.39706      506.5191
## 3 PSU0003      24       96 idStrt001 Rural 0.4791667   32.60417      387.6365
## 4 PSU0004      22       88 idStrt001 Rural 0.4090909   29.59091      419.6189
## 5 PSU0005      28      110 idStrt001 Rural 0.5090909   39.29091      522.6716
## 6 PSU0006      30      116 idStrt001 Rural 0.4655172   29.48276      648.8971
##   gasto.medio Desempleo
## 1    374.8456        NA
## 2    282.2199        NA
## 3    203.0706        NA
## 4    273.0766        NA
## 5    283.1124        NA
## 6    374.7697        NA
```

```r
# Creación base de hogares

Hogares = BigCity %>% group_by(HHID) %>%
                summarise(Personas = n(),
                          estratos = unique(Stratum),
                          upm = unique(PSU),
                          area = unique(Zone),
                          hombres = mean(Sex == "Male"),
                          edad.media = mean(Age),
                          ingreso.medio = mean(Income),
                          gasto.medio = mean(Expenditure),
                          Desempleo = sum(Desocupados)/sum(FT),
                          pobreza = unique(Poverty)) %>%
                as.data.frame()
```