FACULTAD DE MATEMÁTICAS
PONTIFICIA UNIVERSIDAD
CATÓLICA DE CHILE

EYP1113 - Probabilidad y Estadística

Laboratorio 03

Pilar Tello Hernández
pitello@uc.cl

Facultad de Matemáticas
Departamento de Estadística
Pontificia Universidad Católica de Chile

Segundo Semestre 2021

Recordatorio funciones útiles

Para la materia que han visto hasta ahora la cátedra algunas funciones útiles para recordar son:

Nombre de la función	Descripción
`sum(x)`	suma de los componentes un vector `x`
`prod(x)`	producto de los componentes de un vector `x`
`choose(n,r)`	combinatoria de `n` sobre `r`
`factorial(n)`	factorial de `n`
`sample(x,n,replace=)`	obtiene una muestra de tamaño `n` del vector `x`
`round(x,n)`	redondea el número `x` con `n` decimales

Estadística descriptiva

Las medidas de resumen más comunes para variables numéricas se pueden clasificar de la siguiente manera:

- Tendencia Central: Media, Moda, Mediana.
- Posición: Percentil, Mínimo, Máximo.
- Dispersión: Varianza, Desviación Estándar, Coeficiente de Variación, Rango, Rango Intercuantil.
- Forma: Coeficiente de Asimetría, Kurtosis.

Mientras que en las variables no numéricas solo se pueden trabajar como tablas de frecuencias

Estadística Descriptiva

Medidas descriptivas

A continuación se presentan distintos comandos de las principales medidas de resumen en `R`.

Nombre de la función	Descripción
`mean`	Media
`var`	Varianza
`sd`	Desviación estándar
`summary`	Resumen de un vector numérico
`quantile`	Cuantiles de una muestra
`min`	Mínimo de una muestra
`max`	Máximo de una muestra
`range`	Rango de una muestra
`median`	Mediana de una muestra
`table`	Tabla de conteo de una muestra

Estadística Descriptiva

Ejercicios:

a) Importe la base de datos `Tenis.txt` a `R`, adjunte los datos y reconozca los nombres de las variables.

b) ¿Cuál es el pronóstico de `Temperatura` más frecuente?

c) ¿Cuál es el día con el menor pronóstico de `Temperatura_Minima`?

d) Obtenga estadísticas de resumen para las variables `Temperatura_Maxima` y `Temperatura_Minima`.

e) ¿Cuál es la mediana para la `Temperatura_Maxima`?

Programación básica

`if`, `else` y `else if`

Para seguir con esta introducción al lenguaje de `R`, es necesario repasar las herramientas básicas de programación en este nuevo lenguaje:

Para utilizar un `if` el formato es el siguiente:

```
if(condición lógica){
expresión...
}
```

Un ejemplo:

```
x <- 10
if(x%%2==0){
print(paste0(x," es par"))
}
```

Programación básica

`if`, `else` y `else if`

Luego, si queremos introducir más casos se puede usar `else if` y `else`. La manera de utlizarlos es la siguiente:

```
if(condición lógica){
expresión...
}else if(condición){
expresión...
}else{
expresión...
}
```

Programación básica

`if`, `else` y `else if`

Un ejemplo:

```
x <- 10
if(x%%5==0){
print(paste0(x," es múltiplo de 5"))
}else if(x%%3==0){
print(paste0(x," es múltiplo de 3"))
}else{
print(paste0(x," no es múltiplo de 5, ni de 3"))
}
```

Programación básica

Loops: `for` y `while`

El formato para usar un `while` es el siguiente:

```
while(condición lógica){
expresión...
}
```

Un ejemplo:

```
x <- 0
suma=0
while(suma<100){
print(paste0("x vale: ",x,"y la suma es ",suma))
x <- x+1
suma <- suma+x
if(suma>=100){
print(paste0("La suma es mayor o igual a 100 en ",x,
" con un total de ",suma))
}
}
```

Programación básica

Loops: `for` y `while`

Otra forma de romper un while es con el comando `break` de la siguiente manera:

```
while(condición lógica){
if(condición lógica){
break
}
}
```

Programación básica

Loops: `for` y `while`

Para utilizar un `for` el formato es el siguiente:

```
for(variable in vector){
expresión...
}
```

Ejemplo:

```
x <- 1:10
for(i in x){
print(i)
}
```

Programación básica

Loops: `for` y `while`

Si queremos usar el for para recorrer una matriz o una base de datos podemos ocupar indices para recorrerla:

```
x <- matrix(1:20,ncol=4)
for(i in 1:nrow(x)){
for(j in 1:ncol(x)){
print(paste0(x[i,j]," está en la coordenada: ",i,",",j))
}
}
```

ifelse

Podemos aplicar la misma lógica usada con el loop `if()...else...for()` pero ahora usando vectores.

El comando a usar es `ifelse(test, A, B)` donde `test` es una expresión lógica, `A` es lo que se ejecuta si la expresión lógica es verdadera y `B` es lo que se ejecuta si la expresión lógica es falsa.

Veamos el siguiente ejemplo:

```
x <- c(-2, -1, 1, 2)
ifelse(x > 0, "Positivo", "Negativo")
```

function

Para crear funciones en R se utiliza el comando function de la siguiente manera:

```
function(argumentos){
expresión...
return(resultado) o list(resultado)
}
```

Una función puede o no recibir argumentos y retornar o no un resultado, un ejemplo con lo básico:

```
f1 <- function(){
print("Hola mundo")
}
f1()
```

function

Otros ejemplos de funciones:

```
f2 <- function(x){
suma <- sum(x)
return(suma)
}
f2(1:10)
```

Si queremos retornar más de un elemento, podemos retornar una lista:

```
f3 <- function(x){
traspuesta <- t(x)
inversa <- solve(x)
list(tr=traspuesta,inv=inversa)
}
x <- matrix(c(1,3,2,5),ncol=2)
resultados <- f3(x)
resultados$tr
resultados$inv
```

Otras funciones útiles

Nombre de la función	Descripción
`seq(from=a,to=b,by=d)`	secuencia desde `a` hasta `b` cada `d` unidades
`rep(x,n)`	repite `x`, `n` veces
`sort(x)`	ordena el vector `x` de menor a mayor
`rev(x)`	da vuelta el vector `x`
`pmin(`x_1`,...,`x_n`)`	mínimo de cada componente de los vectores
`pmax(`x_1`,...,`x_n`)`	máximo de cada componente de los vectores

Otras funciones útiles

Comandos `sapply`, `apply`, `tapply` y `lapply`

- La familia de funciones `apply` de `R` permiten aplicar funciones a vectores o matrices.
- El comando `sapply(X, FUN, ...)` calcula para cada elemento del vector `X` la función `FUN`. Si la función `FUN` tiene más de un argumento, éstos los podemos agregar en los argumentos `...` de la función `sapply`.
- Ejemplo:
  `x1 <- 1:1000/1000; x1`
  `sapply(x1, round)`
  Repite lo anterior, pero que ahora cada valor redondeado tenga 2 decimales.
  `sapply(x1, round, digits=2)`

Otras funciones útiles

Comandos `sapply`, `apply`, `tapply` y `lapply`

- La función `apply(X, MARGIN, FUN,...)` calcula para cada fila o columna de la matriz `X` la función `FUN`. Si ésta tiene más de un argumento, los agregamos en el argumento `...` de la función `apply`. Para determinar si el cálculo se hace por filas, usamos el argumento `MARGIN=1`, si es por columnas, usamos el argumento `MARGIN=2`

- Ejemplo: Define una matriz de (10×3) de valores desde el 1 hasta el 30 y calcula el promedio de cada columna.
  ```
  X <- matrix(1:30, ncol=3, nrow=10); X
  apply(X, 2, mean)
  ```
  Repite lo anterior pero calculando la media recortada al 20 % para cada columna.
  ```
  apply(X, 2, mean, trim=0.1)
  ```

Otras funciones útiles

Comandos `sapply`, `apply`, `tapply` y `lapply`

- La función `tapply(X, INDEX, FUN , ...)` calcula la función `FUN` al vector `X` dependiendo de los valores del argumento `INDEX` que por defecto se asume categórico y es un vector del mismo largo que `X`. Si la función `FUN` tiene más de un argumento, los agregamos en los argumentos `...` de la función `tapply`.

- Ejemplo:

```
edad <- 30:59;edad
genero <- rep(c("F","M","NB"),10); genero
tapply(edad, genero, mean)
```

Repetir lo anterior, pero calculando la media recortada al 20 %.

```
tapply(edad, genero, mean, trim=0.1)
```

Otras funciones útiles

Comandos `sapply`, `apply`, `tapply` y `lapply`

- El comando `lapply(X, FUN, ...)` opera para cada columna del `data.frame` `X` la función `FUN`. Si la función `FUN` tiene más de un argumento, éstos los podemos agregar en los argumentos `...` de la función `lapply`.

- Ejemplo:

```
m <- matrix(genero,ncol=3); m
m <- as.data.frame(m); m
lapply(m,as.factor)
```