Programación científica en R

Mapas

Marcos Ehekatzin García Guzmán

Octubre de 2024

Funciones

Funciones

- Las funciones son utilizadas para automatizar tareas de una manera general.
  1. Podemos darles un nombre que haga referencia al proceso que estamos automatizando.
  2. Es más fácil y rápido modificar o actualizar el código.
  3. Reducimos las probabilidades de realizar errores.
- Consideraremos escribir funciones siempre que necesitemos copiar y pegar un block de código más de dos veces.

- Ejemplo:

```
df <- tibble::tibble(a = rnorm(10),
                     b = rnorm(10),
                     c = rnorm(10),
                     d = rnorm(10))
df2 <- df %>%
  mutate(a = (a-min(a,na.rm = T))/
             (max(a, na.rm = T)-min(a, na.rm = T)),
         b = (b-min(b,na.rm = T))/
             (max(b, na.rm = T)-min(b, na.rm = T)),
         c = (c-min(c,na.rm = T))/
             (max(b, na.rm = T)-min(c, na.rm = T)),
         d = (d-min(d,na.rm = T))/
             (max(d, na.rm = T)-min(d, na.rm = T)))
```

- En el ejemplo anterior hay un error (difícil de ver) que surge de copiar y pegar el mismo código.
- En este caso, es mejor crear una función.
- Primero vamos a analizar el proceso que queremos automatizar e identificar:
  1. Los inputs
  2. Duplicaciones en el código
  3. Posibles simplificaciones

```
> a = a = (a-min(a,na.rm = T))/
    (max(a, na.rm = T)-min(a, na.rm = T))
```

1. El único input que tenemos es la columna `a`. Propiamente `df$a`. Vamos a darle un nombre temporal a los inputs de la función (`x`).
2. Noten que podemos obtener el mínimo y el máximo utilizando la función `range()` en lugar de duplicar procesos.
3. Para simplificar el código guardaremos el rango en una variable temporal (`rng`).

- Tendríamos:

```
x <- df$a
rng <- range(x, na.rm = T)
(x-rng[1]) /(rng[2]-rng[1])
```

```
##   [1] 0.4357069 0.3490763 0.4536536 0.4548176 0.4648115 0
##   [8] 0.0000000 1.0000000 0.7240155
```

- Para hacer la función seguiremos los siguientes pasos:

1. Elegit un nombre para la función. Por ejemplo `resclae01`.
2. Listar los inputs o argumentos de la función dentro del comando `function`.
3. Pondremos nuestro código en el cuerpo de la función.

```
rescale01 <- function(x){
  rng <- range(x, na.rm = T)
  (x-rng[1]) /(rng[2]-rng[1])
}
rescale01(df$a)
```

```
##  [1] 0.4357069 0.3490763 0.4536536 0.4548176 0.4648115 0
##  [8] 0.0000000 1.0000000 0.7240155
```

- Ahora podemos aplicar la función al data frame de la siguiente manera:

```
df2 <- df %>%
  mutate(a = rescale01(a),
         b = rescale01(b),
         c = rescale01(c),
         d = rescale01(d))
```

Funciones: Ejecución condicional

- Podemos darle condiciones a la función para que solamente se ejecute en algunos casos con las funciones `if` y `else`.

```
>   if(condition){
      #Código a ejecutar conado la condición es TRUE
    } else {
      #Código a ejecutar cuando la condición es FALSE
    }
```

- Podemos utilizarala función `else_if` para añadir condiciones.

Funciones: Ejecución condicional

- Las condiciones siempre debe evaluar si es TRUE o FALSE.
  - Si es un vector tendremos un mensaje de advertencia que nos indicará que solamente el primer elemento será usado.
  - Si la condición es `NA` entonces tendremos un error.
- También podemos usar `||` y `&&`para combinar múltiples expresiones lógicas.
- Nunca hay que usar `|` o `&` porque estos son operaciones vectorizadas (se aplican a diferentes valores).
- Si tenemos un vector lógico podemos colapsarlo a un solo valor con `any()` o `all()`.
- Para la igualdad, utilizaremos la función `identical()`.

Funciones: Ejecución condicional

```
compara <- function(x,y){
  if(x==y){
    "Iguales"
  }else if(x<y){
    "x es menor que y"
  }else if(x>y){
    "x es mayor que y"
  }
}
x <- 5
y <- 10

compara(x,y)
```

```
## [1] "x es menor que y"
```

Funciones: Argumentos inválidos

- A veces es necesario indicar a la función que se detenga.
- Ejemplo:
  - Tenemos una función que hace un promedio ponderado y tenemos los vectores: `x <- c(2*1:10)` y `w <- (.003*100:115)`.
  - Si hacemos el promedio ponderado habrá un error. ¿Cuál?
  - Para detener la función antes de que ocurra el problema utilizaremos la función `stop()`.

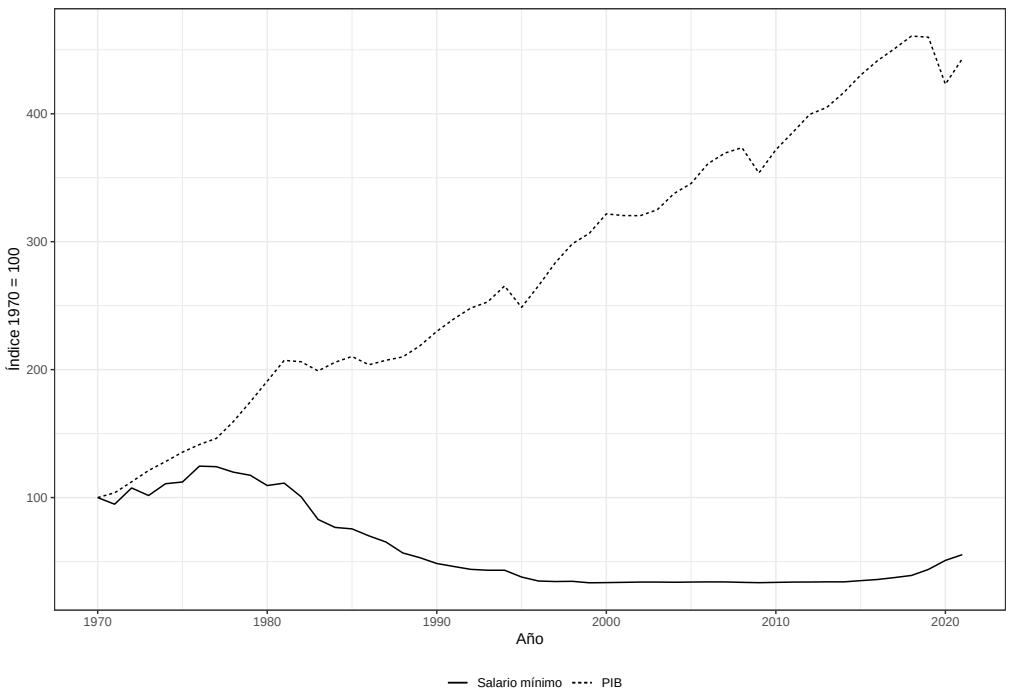
Funciones: Argumentos inválidos

```
wt_mean <- function(x, w){
  if(length(x) != length(y)){
    stop("'x' y 'w' deben tener la misma longitud")
  }else{
    sum(w*x)/sum(x)
  }
}
```

Funciones: Ejercicio 1

- Utilicemos el archivo `s_pib.csv`

1. Haremos una función que nos permita transformar una variable a un índice base 100 en nuestro año de elección.
2. Usemos la función que hicimos para transformar las variables de salario mínimo y PIB a índices 1970 = 100
3. Grafiquemos ambas variables.



Funciones: Ejercicio 2

- La fórmula para calcular una correlación ponderada es la siguiente:

$$r_w = \frac{\sum w_i(X_i - \bar{X}_w)(Y_i - \bar{Y}_w)}{\sqrt{\sum w_i(X_i - \bar{X}_w)^2}\sqrt{\sum w_i(Y_i - \bar{Y}_w)}}$$

- w_i son los pesos que cumplen $\sum_{i=1}^{n} w_i = 1$
- $\bar{X}_w$ y $\bar{Y}_w$ son los promedios ponderados de las variables X y Y.
- Haga una función para calcular la correlación ponderada y aplíquela en las variables de ingresos (ln) y 1) años de escolaridad, 2) sexo, 3) edad.