Funciones

2023-09-25

Funciones

Si te encontrás una y otra vez copiando la misma linea de código para realizar una operación entonces es hora de escribir una función que haga esa tarea por vos. Hasta ahora estuvimos utilizando funciones que vienen con R (lo que llamamos R base) y funciones de librerías que están disponibles para extender las funcionalidades de R. Ahora vamos a ver como definir funciones propias. \

Esta función será un conjunto de comandos o líneas de código que podrá ser definida para usarla en el futuro, con un nombre (con suerte uno que represente la acción u operación para la cual la estamos creando),elementos de entrada y el resultado final.

Genero funciones

Vamos a definir la primera función para convertir viento en nudos a viento en metros por segundo que vamos a llamar nudos_a_ms().

Sintaxis:

```
nudos_a_ms <- function(viento) {
    ms <- viento * 0.5144
    return(ms)
}
```

Genero funciones

function define nuestra función nudos_a_ms() y al correr esas lineas de código se generará un nuevo elemento en nuestro ambiente, la función está lista para ser usada. Pero antes de probarla revisemos la receta para definir una función.

Necesitamos declarar que esto es una función con function(), luego le sigue el cuerpo de la función, las líneas de código que se ejecutarán, y que van entre llaves {}. Usualmente indentamos el código para que se lea mejor, al igual que pasa cuando escribimos el código para generar un gráfico. Pero estos espacios antes del comienzo de cada renglón no son necesarios para que el código funcione.

La función puede recibir uno o más elementos de entrada: los argumentos. En este caso solo recibe un argumento al que llamamos viento. Al final usamos la función return(), que devolverá un valor o elemento que resulte de los cálculos previos.

Uso de funciones

Ahora probemos calcular cuantos metros por segundos son 15 nudos usando la función.

```
nudos_a_ms(viento = 15)
```

```
## [1] 7.716
```

Uso de funciones

Ejercicio: Definir una nueva función que se llame ms_a_nudos() que tome el viento en metros por segundos y lo convierta a nudos. Pista: ahora tenés que multiplicar por 1.944.

Funciones con condiciones

Ahora que ya tenemos un par de funciones escritas, es importante asegurarnos de que funcionen de la manera en la que queremos. En este caso esperamos que dado un valor numérico nos devuelva otro valor numérico. ¿Qué pasaría si recibe un caracter?

Error in nudos_a_ms(viento = “fuerte”) : could not find function “nudos_a_ms”

Funciones con condiciones

Podríamos querer que la función primero revise si el o los elementos de entrada cumplen con determinadas condiciones, por ejemplo que sea un valor numérico.

```
nudos_a_ms <- function(viento) {
  if (!is.numeric(viento)) stop("viento no es numérico")
  # Para R >= 4.0
  # stopifnot("No es numerico" = is.numeric(viento))
  # ¿Es de tipo numérico?
  ms <- viento * 0.5144
  return()
}
```

Funciones con condiciones

Podríamos querer que la función primero revise si el o los elementos de entrada cumplen con determinadas condiciones, por ejemplo que sea un valor numérico.

Error in `nudos_a_ms()`: ! viento no es numérico

El error ahora es un poco más informativo. Con la función stopifnot() que es equivalente a incluir un if, podemos incluir y evaluar más de una condición al mismo tiempo, por ejemplo que el valor de entrada viento sea numérico y menor a un valor máximo.

La gran ventaja de usar funciones

Imaginemos que tenemos un data.frame con información de viento en nudos a lo largo de un día y necesitamos convertirla a metros por segundo, ¿cómo podemos usar las funciones que tenemos?

```
datos_viento <- data.frame(hora = seq(0, 9),
viento = c(21.58, 18.08, 7.19, 7.19, 7.19, 7.19, 7.19,
           3.69, 3.69, 7.19))
head(datos_viento)
```

```
##   hora viento
## 1    0  21.58
## 2    1  18.08
## 3    2   7.19
## 4    3   7.19
## 5    4   7.19
## 6    5   7.19
```

La gran ventaja de usar funciones

Podríamos querer que la función primero revise si el o los elementos de entrada cumplen con determinadas condiciones, por ejemplo que sea un valor numérico.

```
datos_viento$viento_ms <- nudos_a_ms(datos_viento$viento)
head(datos_viento)
```

```
##   hora viento
## 1    0  21.58
## 2    1  18.08
## 3    2   7.19
## 4    3   7.19
## 5    4   7.19
## 6    5   7.19
```

Argumentos por defecto

Podríamos querer que la función primero revise si el o los elementos de entrada cumplen con determinadas condiciones, por ejemplo que sea un valor numérico.

```
nudos_a_ms <- function(viento, conversion = 0.51) {
  if (!is.numeric(viento)) stop("viento no es numérico")
  ms <- viento * conversion
  return(ms)
}
```

Argumentos por defecto

Ahora si convertimos 15 nudos a metros por segundo el resultado podría cambiar según la presición de la constante de conversión que definamos.

```
nudos_a_ms(15) # usa la constante por defecto
```

```
## [1] 7.65
```

```
nudos_a_ms(15, conversion = 0.514444)
```

```
## [1] 7.71666
```

Ejercicio en grupos

Generar una nueva función convertir_viento() que que unifique las dos funciones anteriores (nudos_a_ms() y ms_a_nudos()).

La función deberá incluir un argumento unidad que reciba una variable caracter que permita al usuario indicar si quiere transformar el viento a m/s (“ms”) o nudos (“nudos”).

Posible solucion

```
convertir_viento <- function(viento, unidad = "ms") {
  if (!(unidad %in% c("ms", "nudos"))) stop("Argumento unic
  if (unidad == "nudos") {
  out <- ms_a_nudos(viento)
  } else if (unidad == "ms") {
  out <- nudos_a_ms(viento)
  }
return(out)
}
```

Buenas prácticas: documentación y organización

Nombrar funciones (y cualquier otro elemento) es todo un desafío, necesitamos que se informativo pero al mismo tiempo corto y simple de escribir. Es importante pensar bien este paso.

Nunca está de más documentar la función, es decir, describir para que funciona, que elementos requiere, que es lo que devuelve y por que no, un ejemplo. No necesitamos generar una documentación tan detallada como la que encontramos en la ayuda de cada función de un paquete publicado en CRAN, pero es bueno tener cierto orden.

Buenas prácticas: documentación y organización

Una posible estructura para la documentación sería la siguiente

- Título corto
- Descripción: ¿qué hace la función?
- Lista de argumentos (tipo de objetos, a que corresponde cada cosa, son obligatorios para que funcionen o tienen valores por defecto)
- Salida o resultado (tipo de objeto, estructura)

Ejemplo

```r
# Conviernte variables de viento
# La función convierte viento de nudos a m/s y viseversa.
# Argumentos
## viento: variable numérica. Vector, matrix, etc.
## unidad: caracter, opcional. Permite definir si
## la salida será en m/s (unidad = "ms") o nudos
# Salida
## Vector numérico de igual dimension que la entrada.
convertir_viento <- function(viento, unidad = "ms") {
  if (!(unidad %in% c("ms", "nudos"))) stop("Arg unidad
                                            incorrecto")
  if (unidad == "nudos") {
  out <- ms_a_nudos(viento)
  } else if (unidad == "ms") {
  out <- nudos_a_ms(viento)
  }
return(out)
}
```