# Creación de paquetes en R

**Joselyn Cristina Chávez Fuentes**

**27 de septiembre de 2024**

**Este material posee una licencia tipo Creative Commons Attribution-ShareAlike 4.0 International License.**

**Para conocer más sobre esta licencia, visite http://creativecommons.org/licenses/by-sa/4.0/**

## Material disponible en:

**https://github.com/R-Ladies-Morelia/CreacionPaquetes_R2024**

## Basado en

"R packages" by Hadley Wickham

"Building Tidy Tools" by Charlotte and Hadley Wickham

# Los primeros pasos

# Revisar si podemos usar el nombre del paquete

```
available::available("mipaquete")
```

# Crear la estructura inicial del paquete

```
usethis::create_package("mipaquete")
```

# Podemos agregar la estructura de biocthis

## Pedir que Git ignore el archivo .Rproj

```
usethis::use_git_ignore("*.Rproj")
```

## Crear el respositorio de GitHub

```
usethis::use_github()
```

## Crear el archivo Description estilo Bioconductor

```
biocthis::use_bioc_description()
```

# Crear el archivo README estilo Bioconductor

```
biocthis::use_bioc_readme_rmd()

devtools::build_readme()
```

Recuerda guardar los cambios, hacer commit y push.

# Crear el archivo NEWS estilo Bioconductor

```
biocthis::use_bioc_news_md()
```

# Crear los archivos de ayuda para usuarios y contribuidores

```
biocthis::use_bioc_coc()
usethis::use_tidy_contributing()
biocthis::use_bioc_support()
biocthis::use_bioc_issue_template()
biocthis::use_bioc_citation()
```

# Buenas prácticas para escribir funciones

# Nombre de la función

- Cortos pero descriptivos
- Recomendable: Separar las palabras con _
- Establecer una palabra en común al inicio para familias de funciones

```
use_bioc_citation() # es mejor que

citation()
bioc_cit()
usebioccitation()
useBiocCitation()
use.bioc.citation()
```

# Estructura de la función

- Indentar las líneas de código.
- Agregar comentarios para separar/describir las secciones importantes.
- Usar la sintaxis paquete::funcion() cuando hacemos llamado a funciones de otros paquetes.

```
usethis::use_r("subset_heatmap")
```

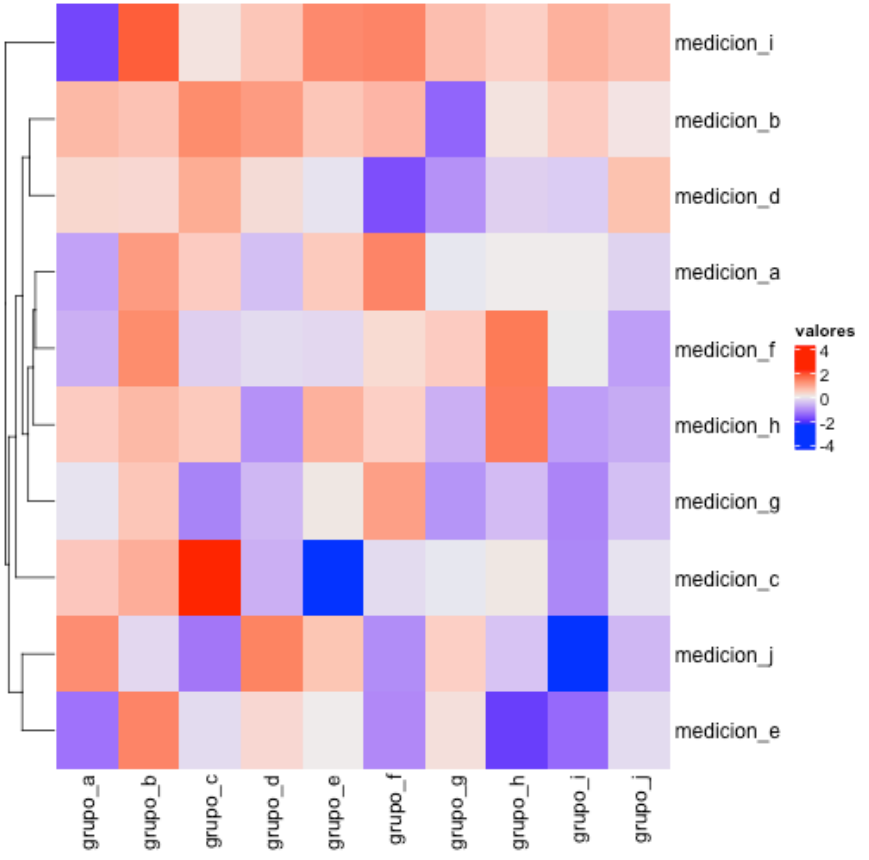
Generemos el código de manera regular.

Simulemos una matriz con diversas mediciones y grafiquemos los datos en un heatmap.

```
mi_matriz <- matrix(rnorm(100), nrow = 10)
rownames(mi_matriz) <- paste0("medicion_",
colnames(mi_matriz) <- paste0("grupo_",let

library(ComplexHeatmap)

Heatmap(mi_matriz,
        cluster_columns = FALSE,
        heatmap_legend_param = list(title
```


medicion_i
medicion_b
medicion_d
medicion_a
medicion_f
medicion_h
medicion_g
medicion_c
medicion_j
medicion_e
valores
4
2
0
-2
-4
grupo_a
grupo_b
grupo_c
grupo_d
grupo_e
grupo_f
grupo_g
grupo_h
grupo_i
grupo_j

Escribamos una función que permita seleccionar algunos grupos de interés y genere el heatmap.

```
library(ComplexHeatmap)

subset_heatmap <- function(x,mediciones=NU
x_subset <- x[mediciones,grupos]
Heatmap(mi_matriz,
        cluster_columns=FALSE,
        heatmap_legend_param=list(title="v
}
```

No la mejor opción

Un poco mejor

```
library(ComplexHeatmap)
subset_heatmap <- function(x, mediciones =
                           grupos = NULL)
    x_subset <- x[mediciones,grupos]
    Heatmap(mi_matriz,
            cluster_columns = FALSE,
            heatmap_legend_param = list(ti
}
```

Mucho mejor

```
subset_heatmap <- function(x, mediciones =
                           grupos = NULL)
    # subset matrix
    x_subset <- x[mediciones, grupos]

    # plot heatmap
    ComplexHeatmap::Heatmap(
      x_subset,
      cluster_columns = FALSE,
      heatmap_legend_param = list(title =
}

subset_heatmap(
  mi_matriz,
  mediciones = c("medicion_a", "medicion_b
  grupos = c("grupo_d","grupo_e","grupo_f"
```

¡Tu turno!

Escribe una función que:

- Filtre la matriz y mantenga sólo los valores por encima de cierto valor.
- Genere el heatmap filtrado.

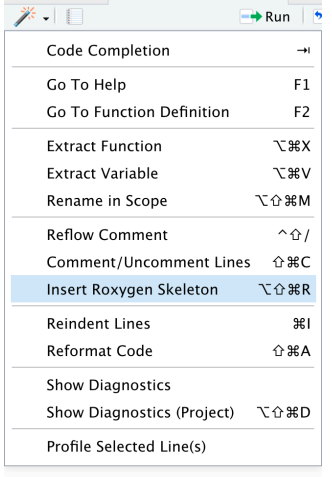
Recuerda seguir las recomendaciones para escribir funciones.

# Documentación

# Documentación

- Descripción de los argumentos/parámetros.
- Ejemplos reproducibles.

Usaremos el paquete roxygen2




- Coloca el cursor dentro de la función.
- Usa la varita mágica para crear el esqueleto de documentación de

Llena los campos de la documentación:

- Para cada argumento/parámetro se debe agregar la descripción. Por ejemplo, el tipo de objeto que esperas que el usuario introduzca.
- Describe lo que hace tu función.
- Indica qué tipo de salida tendrá la función.

```
#' Grafica un heatmap usando un subgrupo de especies y genes
#'
#' @param x Una matriz de genes x especies
#' @param especies character() especies a seleccionar
#' @param genes character() genes a seleccionar
#'
#' @return
#' Un objeto ComplexHeatmap::Heatmap
#' @export
#'
#' @examples
subset_heatmap <- function(x, especies = NULL, genes = NULL) {
```

# Agrega un ejemplo sencillo y reproducible

```
#' @export
#'
#' @examples
#' expresion_genes <- matrix(rnorm(100), nrow = 10)
#' rownames(expresion_genes) <- paste0("gene_",letters[1:10])
#' colnames(expresion_genes) <- paste0("especie_",letters[1:10])
#'
#' subset_heatmap(expresion_genes,
#'                especies = c("especie_a", "especie_b","especie_c"),
#'                genes = c("gene_d","gene_e","gene_f"))
subset_heatmap <- function(x, especies = NULL, genes = NULL) {
```

Una vez que terminamos de escribir la documentación, generamos el archivo de ayuda.

```
devtools::document()
```

Construye el paquete, después reinicia la sesión y carga tu paquete.

```
devtools::build()
```

Esto nos permitirá consultar la ayuda de la función usando:

```
help(subset_heatmap)
?subset_heatmap
```

¡Tu turno!

- Genera la documentación de la función que creaste para filtrar la matriz y graficar un heatmap.

- Comprueba que se pueda consultar la ayuda de tu función.

# Archivos de prueba

# Archivos de prueba

¿Para qué nos sirven los archivos de prueba?

- Estar seguros que la función tiene el comportamiento deseado.
- Verificar que la función es capaz de detectar las entradas correctas y reaccionar ante las entradas incorrectas.
- Verificar que la salida de la función es la esperada.
- Detectar fácilmente cuando una actualización interna o externa rompe nuestro código.

Vamos a utilizar la función subset_heatmap.

Ahora hagamos el archivo de pruebas.

```
usethis::use_testthat()

usethis::use_test("subset_heatmap")

usethis::use_coverage()
```

Pensemos en formas de romper nuestra función:

- ¿Qué pasa si le damos como entrada un data frame?
- ¿Qué pasa si solamente le damos como entrada las mediciones y no los grupos?
- ¿Qué pasa si el vector de mediciones o grupos es numérico en lugar de caracter?

# Escribamos algunas pruebas:

```
mi_matriz <- matrix(rnorm(100), nrow = 10)

rownames(mi_matriz) <- paste0("medicion_",

colnames(mi_matriz) <- paste0("grupo_",let

test_that("Output is a ComplexHeatmap", {
    resultado <- subset_heatmap(
      mi_matriz,
      grupos = c("grupo_a", "grupo_b", "gr
      mediciones = c("medicion_d", "medici

    expect_s4_class(resultado, "Heatmap")
})
```

Evalúa la prueba usando

```
testthat::test_file("tests/testthat/test-s
```

# Siguiente prueba

```
test_that("Empty argument grupos are detec
    expect_error(subset_heatmap(mi_matriz,
                 grupos = c("grupo_a", "gr
})
```

Una más:

```
test_that("Error with data frame works",{
    expect_error(subset_heatmap(as.data.fr
})
```

Modifiquemos la función para poder detectar errores

```
subset_heatmap <- function(x, grupos = NUL
                           mediciones = NU
    # evalate class of x
    stopifnot("x must be a matrix" = inher

    # subset matrix
    x_subset <- x[mediciones, grupos]

    # plot heatmap
    ComplexHeatmap::Heatmap(
      x_subset,
      cluster_columns = FALSE,
      heatmap_legend_param = list(title =
}
```

Vuelve a correr el test:

```
testthat::test_file("tests/testthat/test-s
```

# Viñetas

# Cómo crear una viñeta?

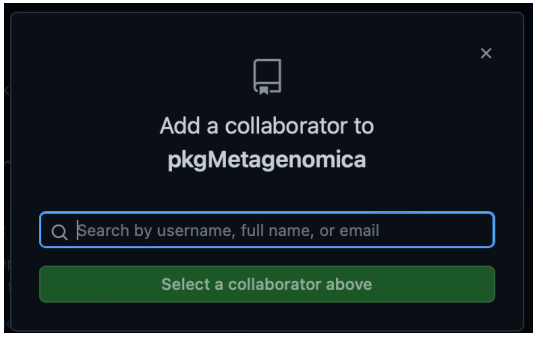
- Debe mostrar un flujo de análisis explotando el potencial de tu paquete.
- Implementa tantas funciones de tu paquete como sea posible, pero no es necesario que incluya todas.
- Los datos a usar deben ser pequeños o fáciles de acceder.
- Puedes crear múltiples viñetas para mostrar diferentes casos de análisis y cubrir una mayor cantidad de funciones.

```
biocthis::use_bioc_vignette(
  name = "introduccion_a_mipaquete",
  title = "Introduccion a mipaquete")
```

# Creado paquetes de forma colaborativa

# Agregando colaboradores al repositorio

- Ve al repositorio de tu paquete.
- Entra a settings > collaborators > Add people
- Agrega el nombre de usuario de tu colaborador.
- Tu colaborador debe aceptar la invitación



¡Tu turno!

- Agrega un colaborador al repositorio de tu paquete.
- Pídele que agregue un archivo de pruebas/test para la función que creaste (filtra la matriz con valores mayores al que provee el usuario).
- Verifica que los cambios se ven reflejados en tu repositorio.
- Cambien de rol y agrega una nueva función al repositorio de tu colega.

# ¡Gracias!