

Introducción al uso de "R" en emergencias de salud

Módulo 1 – Introducción a R / Clase 5

Flujo de trabajo: directorio, entorno de proyecto y tidydata



```
modulo_1 <- ("Clase 5")
```

- > Directorio de trabajo
- > Entorno de proyecto
- > Principios de Tidydata
- > Paquete Tidyverse
- > Piping (%>%)

FICHA_TÉCNICA

Introducción al uso de "R" en emergencias de salud
Módulo 1 – Introducción a R / Clase 5
Flujo de trabajo: directorio, entorno de proyecto y tidydata

ORGANIZACIÓN PANAMERICANA DE LA SALUD/ORGANIZACIÓN MUNDIAL DE LA SALUD (OPS/OMS)

Coordinación General

Dra. Socorro Gross Galiano – Representante de la OPS/OMS – Brasília/Brasil

Dra. Maria Almiron – Advisor, Detection, Verification and Risk Assessment/Health Emergency Information & Risk Assessment (HIM)/ PAHO Health Emergencies Department (PHE) – Washington/United States

Coordinación Ejecutiva

Unidad Técnica de Vigilancia, Preparación y Respuesta a Emergencias y Desastres/ OPS/OMS – Brasília/Brasil

Walter Massa Ramalho

Juan Cortez-Escalante

Laís de Almeida Relvas-Brandt

Mábia Milhomem Bastos

Amanda Coutinho de Souza

Health Emergency Information & Risk Assessment (HIM)/ PAHO Health Emergencies Department (PHE) – Washington/United States

Cristian Hertlein

Wildo Navegantes de Araújo

Equipo Técnico

Unidad Técnica de Vigilancia, Preparación y Respuesta a Emergencias y Desastres/ OPS/OMS
Contenido

Laís de Almeida Relvas-Brandt

Pedro Amparo Leite

Amanda Coutinho de Souza

Revisión

Amanda Coutinho de Souza

Flávia Reis de Andrade

Laís de Almeida Relvas-Brandt

Equipo Pedagógico

Mônica Diniz Durães – Colaboradora – Consultora Nacional de Capacidades Humanas para la Salud/ OPS/OMS

Creación Digital

Pedro Augusto Jorge de Queiroz

Flávia Reis de Andrade

> 1

> Para seguir la clase

Para seguir esta clase, abra RStudio y luego abra el *script* `r_opas_mod1_clase5_flujo.R`, ubicado en la carpeta `r_opas/scripts`. O cree un *script* en blanco para que pueda ingresar los códigos, como aprendió en la clase anterior. Luego ejecute la función `pacman::p_load()` para instalar y/o cargar los paquetes de la clase: el mismo `pacman` y el paquete `tidyverse` (para manejo y tratamiento de datos).

```
# ORGANIZACIÓN PANAMERICANA DE LA SALUD (OPS)
# UNIDAD TÉCNICA DE VIGILANCIA, PREPARACIÓN Y RESPUESTA A EMERGENCIAS Y DESASTRES
# CURSO PRINCIPIANTE DE R APLICADO A EMERGENCIAS DE SALUD
# MÓDULO 1 - INTRODUCCIÓN A R
# CLASE 5 - FLUJO DE TRABAJO: DIRECTORIO, ENTORNO DE PROYECTO Y TIDYDATA

# 1. Para seguir la clase ####
# Instalar/Cargar paquetes
pacman::p_load(pacman,      # Instalar y cargar paquetes
               tidyverso) # Manejo y tratamiento de datos
```

> 2

> Directorio de trabajo

Una de las cosas más importantes para organizar un buen flujo de análisis de datos en R es definir correctamente el directorio de trabajo. Esta es una tarea simple que, si se ignora, puede generar errores o causarle confusión en el futuro.

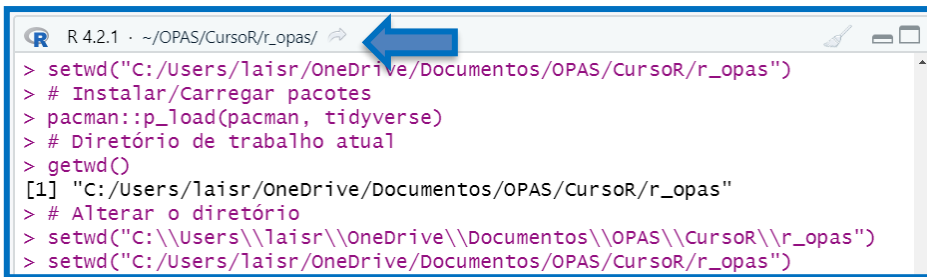
El directorio de trabajo es la carpeta en su computadora que se considerará la raíz para su trabajo. Este es el lugar donde RStudio busca y guarda los archivos de forma predeterminada. Cuando abrimos R y no especificamos el directorio de trabajo, generalmente se considerará que es la carpeta en su computadora donde se guardó R. Usted puede encontrar el directorio de trabajo actual con la función `getwd()` del paquete `base`, como en el código a continuación.

```
# 2. Directorio de trabajo ####  
# Directorio de trabajo actual  
getwd()  
[1] "C:/Users/laisr/OneDrive/Documentos"
```

Usted puede consultar y cambiar el directorio de trabajo accediendo al menú *Tools* → *Global Options...* *R General* → *R Session*. Para cambiar el directorio con código, use la función `setwd()` del paquete base. Para ello, copie la ruta de la carpeta que desea definir como directorio en su computadora e informe como un argumento de la función `setwd()`, entre comillas. Tenga en cuenta en el código a continuación que R acepta dos notaciones para especificar la ruta de su carpeta: con barras invertidas o barras dobles, que no pueden aparecer mezcladas.

```
# 2. Directorio de trabajo ####  
# Cambiar el directorio  
setwd("C:\\Users\\laisr\\OneDrive\\Documentos\\OPAS\\CursoR\\r_opas")  
setwd("C:/Users/laisr/OneDrive/Documentos/OPAS/CursoR/r_opas")
```

Vea que, al definir su directorio de trabajo, él se muestra en la parte superior del panel *console* (Figura 1). Compare el directorio definido en el comando anterior, el resultado presentado antes al ejecutar la función `getwd()` y la ruta que se muestra en la barra de la consola. Antes el directorio estaba en la carpeta *Documentos*, que es donde está instalado R en la computadora que usamos para preparar el curso. La carpeta *Documentos* también participa en la ruta del nuevo directorio definido (el nuevo directorio está dentro de la carpeta *Documentos*). Por eso toda la ruta antes de la carpeta *Documentos* se ha suprimido en la barra de la consola y se ha reemplazado por una tilde (~).



```
R 4.2.1 ~ /OPAS/CursoR/r_opas/
> setwd("C:/Users/laisr/OneDrive/Documentos/OPAS/CursoR/r_opas")
> # Instalar/Carregar pacotes
> pacman::p_load(pacman, tidyverse)
> # Diretório de trabalho atual
> getwd()
[1] "C:/Users/laisr/OneDrive/Documentos/OPAS/CursoR/r_opas"
> # Alterar o diretório
> setwd("C:\\Users\\laisr\\OneDrive\\Documentos\\OPAS\\CursoR\\r_opas")
> setwd("C:/Users/laisr/OneDrive/Documentos/OPAS/CursoR/r_opas")
```

Figura 1: Panel *console* con los códigos para definir y cambiar el directorio de trabajo.

> 3

> Entorno de proyecto


Ahora que usted ya sabe cómo definir un directorio, lo cual es fundamental, le presentaremos una alternativa para evitar que tenga que realizar esta tarea cada vez que inicie una sesión de trabajo en RStudio. Lo recomendado para organizar el flujo de trabajo del curso (y quizás sus futuros análisis) es trabajar con entorno de proyecto.

Los proyectos se crean en nuestra computadora como un archivo con la extensión .Rproj. Son herramientas que ayudan a consolidar todos los recursos necesarios para su análisis. Esto se debe a que, al trabajar en un entorno de proyecto, el directorio de trabajo se dirigirá a la carpeta de su computadora que contiene el archivo ".Rproj", independientemente de dónde se encuentre. Esto es especialmente útil al trabajar en equipo y en diferentes computadoras.

Para nuestras clases, adoptamos el entorno de proyecto y enviamos el archivo .Rproj dentro de la carpeta r_opas que usted descargó con el material del curso¹. De esa forma, R dirigirá el directorio de trabajo a la carpeta r_opas, independientemente de dónde la guarde en su computadora. Y así organizamos los *scripts* de las clases, conjunto de datos, exportación de figuras y otras actividades del curso sin preocuparnos por el directorio de trabajo en la computadora de los(las) estudiantes.

¹ Si es necesario, consulte el documento r_opas_instrucciones_material.pdf, disponible en la misma carpeta de las clases (r_opas/clases).

Abrir un proyecto

Para abrir un proyecto, haga clic en el icono  **Project: (None)** en la parte superior derecha de los paneles en RStudio y luego en *Open Project* (Figura 2). Se abrirá una ventana, donde usted debe ubicar la carpeta `r_opas`, seleccionar el archivo de proyecto homónimo (`r_opas.Rproj`) dentro de ella y hacer clic en *Open*, como en la Figura 3. Al realizar este paso, observe que el ícono del proyecto en la parte superior a la derecha de los paneles en RStudio ha cambiado automáticamente de *None* al nombre del proyecto y el directorio de trabajo en la barra de la *console* ha cambiado a la carpeta `r_opas`, donde está guardado el archivo de proyecto.

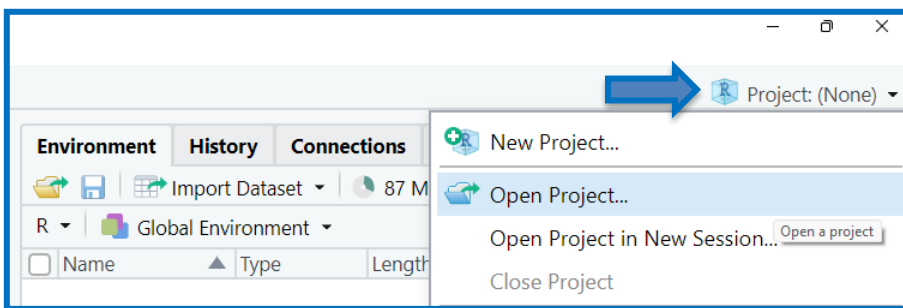


Figura 2: Vista del icono *Project* referente al proyecto activo.

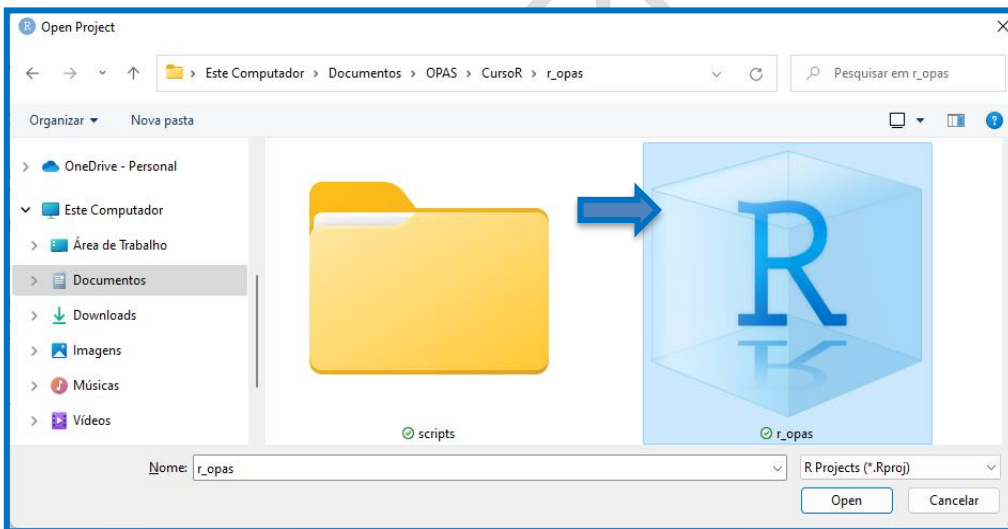


Figura 3: Vista del archivo de proyecto (`r_opas.Rproj`) dentro de la carpeta `r_opas`.

Usted también puede abrir un proyecto directamente desde su carpeta `r_opas`. Para probarlo, guarde sus archivos y cierre RStudio. Ahora encuentre la carpeta `r_opas` en su computadora y haga doble clic en el archivo `r_opas.Rproj`, como solemos hacer para abrir

un archivo o carpeta. Tenga en cuenta que RStudio iniciará con el proyecto del curso abierto y el directorio definido.

Crear un proyecto

Para crear un proyecto y usarlo en sus análisis futuros, usted puede hacer clic en el mismo icono de proyecto que se muestra en la Figura 2 y elegir la opción *New Project*. En la ventana que se abrirá (Figura 4), usted tiene la opción de crear una nueva carpeta para el proyecto (*New Directory*) o elegir una carpeta que ya existe en su computadora (*Existing Directory*). Si usted es principiante, prefiera la segunda opción.

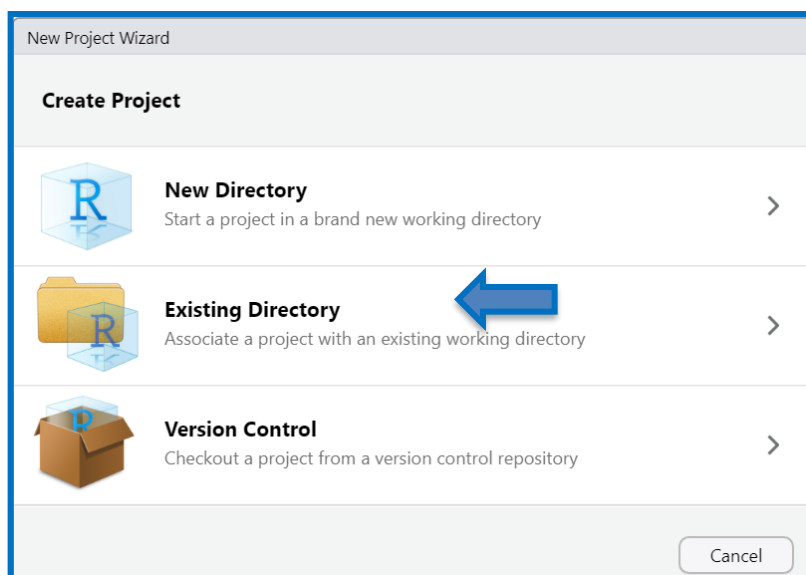



Figura 4: Vista de la ventana *New Project Wizard* con las opciones contenidas en el ítem *New Project*.

En la nueva ventana que se abrirá, haga clic en *Browse* (Figura 5) y elija la carpeta en su computadora donde guardará el archivo .Rproj. Luego asigne un nombre a su archivo y haga clic en *Open*. Se creará el archivo y luego se cargará el proyecto. También es posible crear proyectos desde el menú *File* → *New Project* o desde el botón , ubicado en la línea inferior del menú *File*.

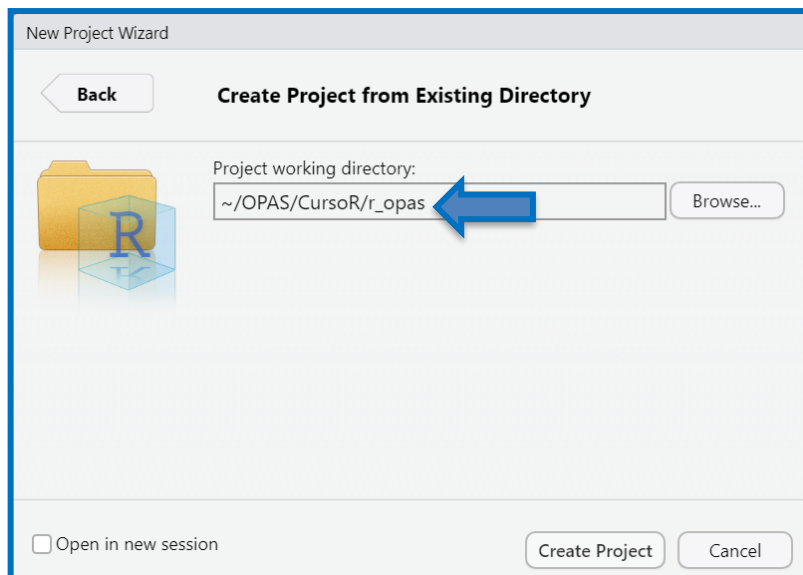


Figura 5: Vista de la ventana *New Project Wizard* con la opción *Browse* para guardar el nuevo proyecto (archivo .Rproj).

Para organizar su flujo de trabajo, al crear un proyecto, recuerde concentrar todo lo que necesita en esta carpeta. Por ejemplo, digamos que su tarea es preparar un informe epidemiológico para describir los casos hospitalizados por un cuadro clínico de etiología por aclarar que se notificaron en dos hospitales de su ciudad. Vale la pena tener las dos listas de casos en la misma carpeta donde se encuentra el archivo de su proyecto .Rproj (o al menos dentro de ella), además de guardar los *scripts* y otros documentos en esa misma carpeta.

Cerrar un proyecto

Para cerrar un proyecto, simplemente haga clic en el mismo icono de proyecto que se muestra en la Figura 2 y elija la opción *Close Project*. Antes de eso, recuerde guardar su *script* u otros documentos. Si usted configuró su RStudio según las instrucciones de la clase 3 y si tiene objetos en el panel *environment*, en este momento se le preguntará si desea guardar el *workspace* (archivo con extensión .RData). Le recomendamos que elija no guardar hasta que comprenda qué son estos archivos, como veremos en la siguiente clase. Tenga en cuenta que se iniciará una nueva sesión cada vez que migre

entre proyectos. Al igual que un proyecto se cerrará automáticamente si usted reinicia la sesión de trabajo actual.

> 4

> Principios de Tidydata

Cuando iniciamos el análisis de una base de datos, rara vez encontramos datos organizados, tratados y listos para exportar los resultados de interés. Esto se debe a que los datos a menudo se organizan para facilitar un propósito que no sea el análisis, como facilitar la entrada de datos por parte de los digitadores o exportar una tabla de rutina, entre otros.

Cuando un servicio de vigilancia epidemiológica construye un cuestionario para recolectar datos primarios, es posible planificar las preguntas que se realizarán, las respectivas variables que se crearán en el futuro en la base de datos y cómo configurarlas. Por ejemplo, si se construye un formulario para entrevistas domiciliarias durante una encuesta serológica o vacunal en un barrio, el servicio puede elaborar el cuestionario con enfoque en el resultado que se quiere presentar y así optimizar el trabajo que se realizará en la etapa de análisis.

Pero eso no suele ser lo que sucede. Es mucho más común que necesitemos trabajar con datos secundarios de los servicios de salud o con datos primarios que han sido recolectados por otras personas, casi nunca con un instrumento de recolección planificado y, en ocasiones, sin ninguna estandarización. De hecho, las bases de datos "desordenadas" suelen ser la regla y no la excepción en un análisis epidemiológico, y ordenarlas suele tardar más de la mitad del tiempo necesario para exportar sus resultados.

En el siguiente módulo, comenzaremos a tratar los datos que se utilizarán en este curso y usted podrá basarse en las soluciones que se presentarán para preparar cualquier conjunto de datos de su interés para el análisis. Antes de comenzar, conviene considerar tres reglas generales propuestas por Hadley Wickham & Garret Grolemund en el libro *R for Data Science*. Como se ilustra en la Figura 6, para ordenar los datos de la base que vamos a analizar (dejarlos *tidy*):

- 1) Cada variable debe tener su propia columna;

- 2) Cada observación debe tener su propia línea;
- 3) Cada valor debe tener su propia celda.



Figura 6: Ilustración de la organización de la base de datos según principios *tidydata*.

Fuente: adaptada de <https://r4ds.had.co.nz/tidy-data.html>.

> 5

> Paquete Tidyverse

Tidyverse es una colección de otros paquetes útiles para actividades comunes de ciencia de datos. El término tidyverse puede entenderse como “versión ordenada”. Creado para facilitar y agilizar determinados pasos de manejo y visualización de datos, todos los paquetes incluidos en tidyverse comparten la misma filosofía y gramática de funciones, lo que se ha demostrado que optimiza el flujo de análisis desde su creación y amplia difusión en la ciencia de datos.

Para obtener más información, visite la página web <https://www.tidyverse.org/> y explore la pestaña *Packages* (Figura 2). Usted encontrará una lista de los principales paquetes incluidos en el tidyverse, que se cargan automáticamente junto con él. Todos los demás paquetes deben estar específicamente activados, aunque estén incluidos en el tidyverse, como en el caso del paquete lubridate para el manejo de variables de tipo fecha.

Para usar el tidyverse en RStudio, se debe instalarlo y activarlo como un paquete, como aprendimos en la última clase. Recuerde que en este curso usaremos la función

`pacman::p_load(tidyverse)` para instalar y/o cargar los paquetes al inicio de cada clase, como hicimos en la primera sección. Usted también puede elegir declarar los paquetes de tidyverse de forma individualizada (por ejemplo: `pacman::p_load(dplyr, ggplot2)`) para que no tenga que cargar todos al mismo tiempo.

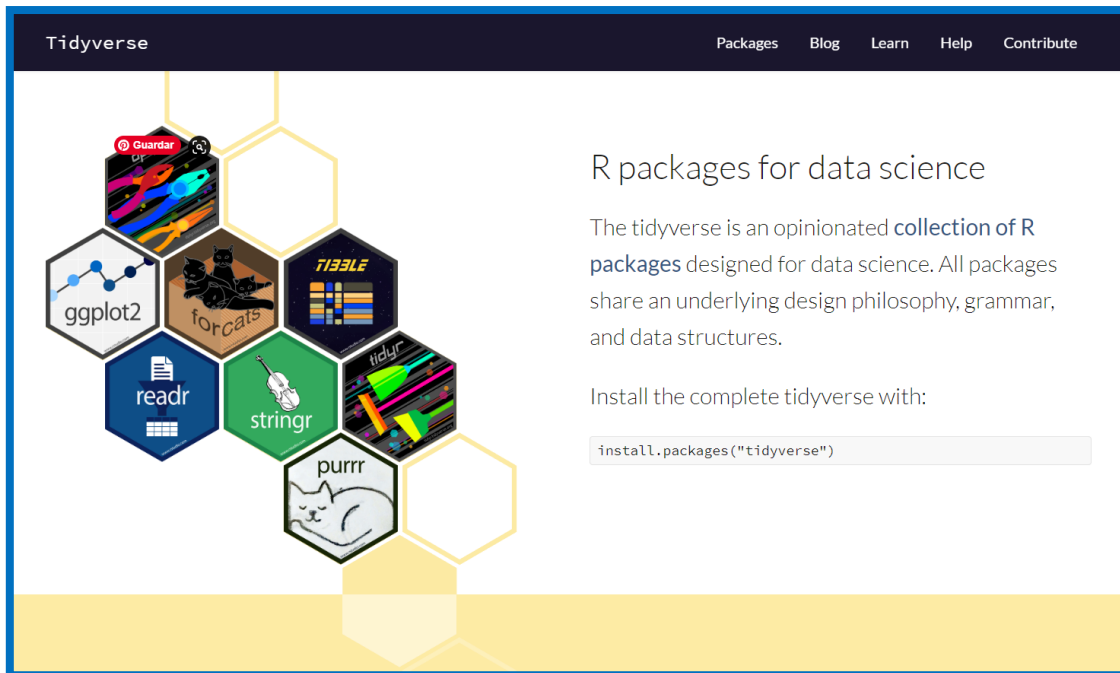


Figura 7: Página web con algunos de los paquetes incluidos en la colección tidyverse.

Entre los principales paquetes del tidyverse, tenemos: `ggplot2` (visualización de los datos), `dplyr` (manejo de los datos), `tidyr` (organización de los datos), `readr` (importación de datos), `purrr` (manejo de funciones y vectores), `tibble` (manejo de `data.frame`), `stringr` (manejo de datos en el formato carácter) y `forcats` (manejo de factores). No todos se utilizarán en este curso.

> 6

> Piping (%>%)

El concepto de *pipe* en programación existe al menos desde la década de 1970 y, desde entonces, ha aparecido en las más diversas aplicaciones, desde HTML hasta R. El operador puede tener diferentes apariencias según la aplicación, pero su objetivo es

siempre el mismo: simplificar comandos y transportar sus resultados a otros comandos. Por ejemplo, ejecute el siguiente código y observe cómo solicitamos a R que calcule la media redondeada del objeto vector: primero pedimos la media con la función `mean()` sin ningún argumento y entregamos ese resultado a la función `round()`, ambas del paquete `base`.

```
# 6. Piping (%>%) ####  
# Crear un vector  
vector <- c(10,12,15,18,14,12,16,18,17,19,18,15,16,14,18)  
  
# Calcular la media y redondear  
vector %>%  
  mean() %>%  
  round(digits=1)  
[1] 15.5
```

El término *pipe* en inglés significa “tubo”, “tubería” o “encadenamiento”, pero en el día a día de la programación es más habitual utilizar el término en inglés. En R, el *pipe* tiene un símbolo algo raro (`%>%`), pero en el fondo es solo una función. También puede usar los comandos de teclado `Ctrl + Shift + M` para agregar un *pipe* sin tener que escribir la expresión exacta (`%>%`).

Para escribir un código con múltiples instrucciones antes del *pipe*, era necesario descomponer el problema en una serie de pasos. En este flujo de trabajo, a menudo se creaban objetos para guardar cada paso, muchas veces con un sufijo para diferenciar sus nombres (por ejemplo, `casos`, `casos_confirmados`, `casos_conf_cerrados`, `casos_conf_cerr2`). Por ejemplo, el comando para calcular la media redondeada del objeto vector podría reescribirse como en el código siguiente.

```
# 6. Piping (%>%) ####  
# Crear un objeto con el valor de la media  
media <- mean(vector)  
media  
[1] 15.46667
```

```
# Crear un objeto con la media redondeada
```

```
media2 <- round(media, digits=1)
```

```
media2
```

```
[1] 15.5
```

En el código anterior, observe que se realizaron dos operaciones para llegar al resultado deseado (fue necesario ejecutar dos líneas de comando), mientras que con *pipe* se ejecutan a la vez. Decimos que la sintaxis de *pipe* se enfoca en acciones y no en objetos. ¿Pero, por qué actualmente no se recomienda el uso de programas centrados en los objetos?

Primero, por la necesidad de crear o nombrar cada objeto intermedio, lo que puede ser un impedimento para el procesamiento de grandes bases de datos (incluso si usamos la estrategia de sobrescribirlos). El ejemplo de la media se construyó con base en un simple vector, pero recuerde que habitualmente trabajamos con objetos de tipo *data.frame*, que pueden requerir más de la memoria de su computadora.

En segundo lugar, porque debemos recordar cómo se configuró el conjunto de datos en cada paso intermedio y tener cuidado de hacer referencia al correcto. Si identificamos el objeto equivocado, no obtendremos la respuesta correcta. Peor aún, no tenemos un mensaje de un error explícito porque el código funciona con el objeto incorrecto. R ejecuta el comando, sin embargo, no sabe cómo advertirnos de que usamos la base incorrecta.

Sin embargo, es importante señalar que no hay correcto o incorrecto. Comúnmente encontramos más de una forma de escribir los mismos pasos de manejo de nuestros datos en R, y elegir uno de ellos también se relaciona con el estilo de escritura y la preferencia de quien está programando.

Consejos para la legibilidad de su código

Un buen estilo de programación es comparable a una puntuación correcta: usted puede entender el texto sin ella, sin embargo, ella hace que sea mucho más fácil leer las cosas.

- Comience el código con el nombre del objeto, ingrese un *pipe* (*%>%*) y declare la secuencia de acciones (funciones) que se realizará, insertando también un *pipe*

entre cada una de ellas, para entregar el resultado de una a la otra hasta el final del código;

- El pipe `%>%` debe estar precedido por un espacio en blanco y preferiblemente seguido por una nueva línea para enfatizar la secuencia de acciones que se están realizando;
- Mantenga los espacios creados automáticamente por R cuando usted inicia una nueva línea, para que sea más fácil identificar visualmente las funciones y sus argumentos (por ejemplo, al escribir el nombre del objeto + *pipe* e iniciar una nueva línea, se crean dos espacios);
- Si una función tiene muchos argumentos, también prefiera poner cada uno en una línea;
- Use espacios entre las diferentes líneas de comando;
- Use comentarios para dejar notas sobre los pasos de análisis para que usted o sus compañeros de trabajo puedan seguir lo que se hizo.

> 7

> Referencias

Batra, Neale, et al. The Epidemiologist R Handbook. 2021. DOI:10.5281/zenodo.475264610.528 Disponible en: <https://epirhandbook.com/en/>.

Silva, Henrique Alvarenga da. Manual Básico da Linguagem R: Introdução à análise de dados com a linguagem R e o RStudio para área da saúde. [recurso electrónico] / Henrique Alvarenga da Silva - 1ª edición – São João del Rei: Editora do Autor, 2018. ISBN: 978-65-900132-0-0 (e-book).

Wickham, H., & Grolemund, G. R for data science: import, tidy, transform, visualize, and model data. O'Reilly Media, Inc. 2016. ISBN: 9781491910399. Disponible en: <https://r4ds.had.co.nz/>.