Econometría I

Pablo Astudillo-Estevéz Elías José Mantilla

2023-10-04

Iniciándose en R

La programación de R ha evolucionado a través de los años y es utilizada para manipular datos, estadísticas avanzadas, visualizaciones y machine learning. Aquí te explicaremos cómo empezar a usarlo.

Instalación

La instalación de R y Rstudio es un proceso sencillo.

R: Primero debes descargar el software estadístico R desde CRAN y seguir las instrucciones que te proporciona el sitio para instalarlo en tu computadora.

RStudio: Es una interfaz gráfica más amigable para R. Puedes descargarlo desde el sitio web de Rstudio después de haber instalado R.

La Interfaz de R y Rstudio

La Interfaz de R es bastante simple. Cuando se abre, se muestra una sola ventana con un prompt de comando donde puedes escribir código.

RStudio, por otro lado, es una interfaz más sofisticada. Tiene cuatro vistas principales.

Consola de R: Aquí es donde se ejecutan los códigos de R.

Script: Aquí es donde se pueden escribir y guardar códigos más largos.

Entorno/Historia: Muestra los conjuntos de datos y variables que se están utilizando.

Tablas/Gráficos/Paquetes/Ayuda: Muestra tablas y gráficos, permite instalar y cargar paquetes y proporciona ayuda.

Scripts

Scripts en R son archivos de texto simple que contienen código. RStudio permite la creación de estos scripts para la ejecución de tareas más complejas.

Scripts de R

Los Scripts de R son archivos con la extensión .R que contienen código R. Se pueden ejecutar línea por línea, o todo a la vez. Cuando se abren en RStudio, se muestran en la pestaña “Script”.

Scripts de Rmarkdown

Rmarkdown es una función de RStudio que permite combinar texto y código R en el mismo archivo, y exportar los resultados como un documento HTML, PDF, Word, etc. Los scripts de Rmarkdown tienen la extensión .Rmd.

Scripts de Quarto

Quarto es una herramienta reciente que mejora las capacidades de Rmarkdown. Permite la creación de documentos reproducibles utilizando varios lenguajes de programación, incluido R. Los scripts de Quarto tienen la extensión .qmd y pueden ser convertidos a varios formatos como html, pdf, docx, y muchos más.

```
# Establecer una ventana al CRAN - Comprehensive R Archive Network
options(repos = c(CRAN = "https://cloud.r-project.org"))

# Instalar el paquete wooldrige, si no está instalado
if (!requireNamespace("wooldridge", quietly = TRUE)) {
  install.packages("wooldridge")
}

# Loading up packages
library(wooldridge)
library(tidyverse)

# Print the current working directory
here::here()
```

Objetos de la distribución base de R

R se basa en un conjunto de objetos para mantener y manipular los datos. En R, una variable es un nombre almacenado en un espacio de nombres que tiene un valor y cantidades asociadas a objetos de datos.

Variables en R

Las variables en R son entidades que almacenan datos con el fin de ser manipulado. La flexibilidad de R radica en que estas variables no necesitan ser declaradas como algún tipo de dato particular, lo cual hace que la programación sea más fácil. El tipo de datos de la variable se decide en tiempo de ejecución.

```r
# Definición de variables
x <- 10
y <- - 5

# Operaciones sobre variables numéricas
x - y

x + y

x * 2

x ** 2

x / y

# Funciones útiles
abs(y)

sqrt(4)

exp(1)

log(1)

log(1, base = 10)

round(4.234123599, digits = 2)

factorial(10)
```

Vectores en R

Un vector es la estructura de datos más básica en R. Un vector es simplemente una serie de datos que comparten el mismo tipo (numérico, caracter o lógico). Los vectores se crean utilizando la función c().

```r
# Definición de un vector
randc <- c(1, 2, 3, 4, 5)

# Broadcasting de operaciones metemáticas
randc - 1
```

```r
randc * 2

# Realizaciones aleatorias que podemos guardar en vectores
rand_norm <- rnorm(5)
rand_norm

rand_poi <- rpois(5, lambda = 5)
rand_poi
```

Matrices en R

Una matriz es un vector bidimensional con elementos de la misma clase. Las matrices se crean con la función matrix() y se pueden dar nombres a sus filas y columnas.

```r
v <- runif(16)

A <- matrix(v, nrow = 4)

A

rbind(A, c(0.25, 0.5, 0.75, 1))

cbind(A, c(0.25, 0.5, 0.75, 1))
```

Arreglos o Listas en R

El arreglo o lista es otra estructura de datos en R que puede contener elementos de diferentes clases. En este sentido, las listas en R son similares a las listas en Python. Una lista puede incluso tener otra lista como elemento.

```r
# Definiendo una lista
my_list <- list(
  name = "John",
  age = 25,
  grades = c(90, 85, 92),
  is_student = TRUE
)

# Acesso a los objetos dentro de una lista
cat("Name:", my_list$name, "\n")
```

```r
cat("Age:", my_list$age, "\n")
cat("Grades:", my_list$grades, "\n")
cat("Is Student?", my_list$is_student, "\n")

# Adición de elementos a una lista
my_list$city <- "New York"
cat("City:", my_list$city, "\n")

# Modificando la lista
my_list$age <- 26
cat("Updated Age:", my_list$age, "\n")

# Removiendo elementos de la lista
my_list$city <- NULL

# Checking the updated list
str(my_list)
```

Dataframes en R

El DataFrame `data.frame` es una estructura de datos tabular en R donde los datos están almacenados en filas y columnas indexadas. Es similar a una hoja de cálculo. Cada columna de un DataFrame puede tener un tipo de dato diferente cual lista de vectores. Es posible agrupar o circumscribirse a variables numéricas de una matriz y obtener la funcionanlidad de matrices. No obstante, Los tibble son una mejora de los data.frame en R que ofrecen múltiples ventajas. Son más informativos al mostrar de manera clara las primeras 10 filas y todas las columnas, facilitando la visualización de datos, mientras que los data.frame pueden volverse difíciles de leer en conjuntos de datos grandes. Los tibble aplican reglas consistentes en la interpretación de nombres de columnas, evitando conversiones automáticas de caracteres especiales. Además, manejan de manera más efectiva los datos faltantes y no convierten caracteres en factores automáticamente. Son más eficientes en memoria, tienen una indexación más amigable, son compatibles con tidyverse y se imprimen de manera ordenada en cuadernos interactivos, lo que simplifica el trabajo con datos en R.

Funciones en R

Las funciones son conjuntos reutilizables de instrucciones que realizan tareas específicas. R tiene una serie de funciones incorporadas y también permite a los usuarios definir sus propias funciones.

La sintáxis para definir una función es la siguiente:

```r
.func <- function(arg) {
```

```
  recipe
}
```

```r
# Definiendo una función
square <- function(x) {
  return(x ** 2)
}

square(4)
```

El Tidyverse

El Tidyverse es un conjunto de paquetes en R diseñados para la ciencia de datos. Todos los paquetes comparten una filosofía de diseño subyacente y gramáticas y estructuras de datos comunes.

Tibbles

Los tibbles son una versión moderna de los data frames en R pero con algunas características adicionales y mejor comportamiento por defecto.

```r
# Establecer una ventana al CRAN - Comprehensive R Archive Network
options(repos = c(CRAN = "https://cloud.r-project.org"))

# Instalar el paquete wooldrige, si no está instalado
if (!requireNamespace("MASS", quietly = TRUE)) {
  install.packages("MASS")
}

# Parámetros de la simulación
mu <- c(0, 0)
sigma <-
  tribble(
  ~x, ~y,
  1, 0.5,
  0.5, 1
)

# Generación de muestras aleatorias
msamples <- MASS::mvrnorm(100, mu = mu, Sigma = sigma)
```

```r
# Guardar las mustras aleatorias en un tibble
data <- msamples %>%
  as_tibble() %>%
  rename(x = `1`,
         y = `2`)
```

dplyr y tidyr

dplyr y tidyr son paquetes esenciales para manipular y limpiar datos. dplyr proporciona un conjunto de herramientas para manipular eficientemente marcos de datos, mientras que tidyr ayuda en la limpieza de datos.

```r
# Asesinatos por condado del libro de Wooldrige
data <- wooldridge::countymurders

# Inspeccionar los datos
data %>% View()
data %>% glimpse()

# ¿Qué condado tiene más asesinatos?
data %>%
  filter(year == 1996) %>%
  arrange(desc(murders)) %>%
  slice_head() %>%
  pull(countyid)

# ¿Qué condado tiene más alta tasa de asesinatos?
data %>%
  filter(year == 1996) %>%
  arrange(desc(murdrate)) %>%
  slice_head() %>%
  pull(countyid)

# ¿Cúal es la correlación de la tasa de homicidios con la \
# densidad poblacional? ¿Y con el porcentage de población \
# negra?
data %>%
  filter(year == 1996) %>%
  # summarise(across(c(density, percblack), ~cor(murdrate, .))) %>%
  summarise(cor(murdrate, density), cor(murdrate, percblack))
```

```r
# ¿Cúales son los promedios de la tasa de homicidios, \
# densidad poblacional y porcentaje de población negra por \
# condado?
data %>%
  group_by(countyid) %>%
  # summarise(across(c(murdrate, density, percblack), ~mean(.))) %>%
  summarise(mean(murdrate), mean(density), mean(percblack))

# Entendiendo la distribución de poblaciones de los condados
data %>%
  filter(year == 1996) %>%
  select(popul) %>%
  summarise(mean = mean(popul),
            sd = sd(popul),
            mad_sd = mad(popul),
            median=quantile(popul, c(0.5))) %>%
  pivot_longer(cols = everything(),
               names_to = "statistic",
               values_to = "estimate")
```

ggplot2

ggplot2 es un sistema para crear gráficos en R basado en la gramática de gráficos, lo que permite la creación de gráficos complejos a partir de bloques de construcción gráficos simples y coherentes.

```r
# Función de Distriución de Probabilidad Acumulada Empírica
ecdf <- function(v) {
  .n <- length(v)
  .x <- sort(v)
  .probs <- seq(1, .n) / .n
  .tbl <-
    tibble(x = .x, y = .probs)
  return(.tbl)
}

# Gráfica de Distribución de Probabilidad Acumulada Empírica
data %>%
  filter(year == 1996) %>%
  pull(popul) %>%
  ecdf() %>%
```

```
  ggplot(aes(x, y)) +
  geom_point()

# Gráfica de Distribución de Probabilidad Acumulada Empírica
data %>%
  filter(year == 1996) %>%
  pull(popul) %>%
  ecdf() %>%
  ggplot(aes(x, y)) +
  geom_point() +
  scale_x_log10()

# Gráfica de Dispersión de la tasa de homicidios y la \
# densidad poblacional
data %>%
  filter(year == 1996) %>%
  ggplot(aes(murdrate, density)) +
  geom_point()

# Controlando límites de los ejes
data %>%
  filter(year == 1996) %>%
  ggplot(aes(murdrate, density)) +
  geom_point() +
  scale_x_continuous(limits = c(0, 8)) +
  scale_y_continuous(limits = c(0, 20000)) +
  labs(title="Tasa de homicidios vs. Densidad poblacional",
       x = "Tasa de homicidios",
       y = "Densidad poblacional")
```

purrr

purrr mejora la programación funcional en R al proporcionar un conjunto completo y coherente de herramientas para trabajar con funciones y vectores.

```
data %>%
  group_by(year) %>%
  nest() %>%
  mutate(model = map(data, ~lm(murdrate ~ percblack, data = .x))) %>%
  transmute(slopes = map_dbl(model, ~coef(.x)[2]))
```

Importar y Exportar Datos en R

Datos CSV

R proporciona funciones para leer archivos CSV. Una forma común de importar datos desde un archivo CSV es usar la función `read.csv()`:

```
datos <- read.csv("ruta.csv")
```

Los argumentos comunes que puedes querer especificar incluyen:

- `file`: la ruta y el nombre del archivo CSV que deseas leer.
- `header`: un valor booleano que indica si la primera fila del archivo CSV contiene nombres de columnas.
- `sep`: el carácter utilizado para separar las columnas.

Datos Excel

Para importar datos desde un archivo Excel, puedes usar el paquete `readxl` y su función `read_excel()`:

```
library(readxl)
datos <- read_excel("ruta.xlsx")
```

Los argumentos que puedes querer especificar incluyen:

- `path`: la ruta y el nombre del archivo Excel que deseas leer.
- `sheet`: el nombre o número de la hoja que deseas leer.

Datos en la Web

Puedes leer archivos CSV directamente desde la web utilizando la función `read.csv()`:

```
grants <- read.csv("https://github.com/rfordatascience/tidytuesday/raw/master/data/2023/20

grants %>% glimpse()
```

Exportar datos y gráficas en R

Guardar Tablas y Tibbles

La función `write.csv()` permite escribir datos en un archivo CSV:

```
write.csv(datos, "ruta.csv")
```

Los argumentos comunes que puedes querer especificar incluyen:

- `x`: el marco de datos que deseas escribir.
- `file`: la ruta y el nombre del archivo CSV donde deseas escribir.

Guardar gráficas con ggplot2

Para guardar una gráfica que has creado con `ggplot2`, puedes usar la función `ggsave()`:

```
ggplot(datos, aes(x=GDPperCapita, y=LifeExpectancy)) +
  geom_point()

ggsave("ruta.png")
```

Los argumentos más comunes que puedes querer especificar incluyen:

- `filename`: la ruta y el nombre del archivo donde deseas guardar la gráfica.
- `plot`: el objeto de trazado que deseas guardar. Si no se especifica, `ggsave()` guardará el último trazado que hayas creado.
- `device`: el tipo de salida (por ejemplo, “png”, “pdf”).
- `width`, `height`, `units`: las dimensiones de la salida (por defecto, las unidades son pulgadas).