1 Introducción

En las últimas décadas la economía ha ido adoptando a los computadores para realizar econometría aplicada. Sin embargo, la manera en que se enseña econometría sigue siendo eminentemente teórica. La gran mayoría de los textos de econometría deriva los modelos como ejercicios de álgebra lineal, y más tarde (o nunca) intentan conectar esto con el trabajo aplicado, usando ejemplos específicos y pequeños extractos de código (si es que hay algo de código). Este libro intenta dar la vuelta a este paradigma, apoyándose en R.

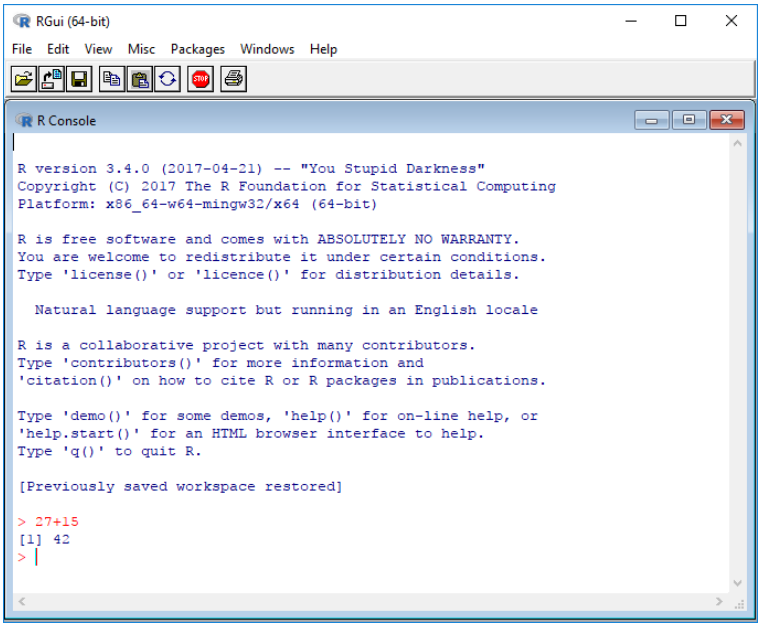
1.1 ¿Qué es R?

R es un ambiente de software y un lenguaje de programación interpretado para manipular datos, hacer análisis estadístico y visualizar datos. Es una implementación de S, un lenguaje de programación matemático orientado a objetos más antiguo. Es software libre y de código abierto, constantemente desarrollado y ampliado por colaboradores de múltiples disciplinas.

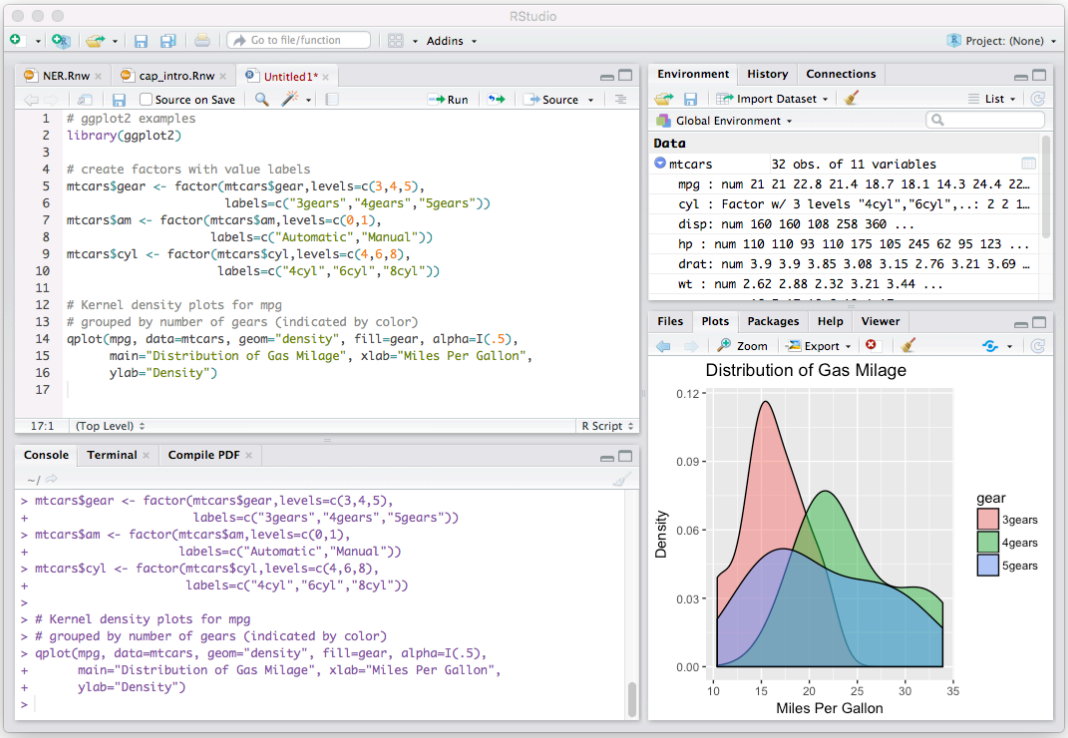
R es mucho más flexible que la mayoría de los paquetes estadísticos normalmente usados por economistas. Es un lenguaje de programación completamente desarrollado, no sólo un programa con tests y métodos pre-programados.

1.2 Obtener R

La instalación base de R puede obtenerse directamente de `https://cloud.r-project.org/`. Una vez instalado puedes usar RGui para interactuar con R. Sin embargo, esta opción no es demasiado sexy:



Mi recomendación es usar RStudio, un IDE (*integrated development environment*) que nos ayudará a trabajar mucho más cómodamente con R. RStudio incluye una consola, un editor de código con resaltado de sintaxis, explorador de objetos y una larga lista de cosas que puede que no entiendas aún, pero que seguro harán tu vida más fácil. Con toda seguridad se *ve* mejor que RGui. Puedes descargar RStudio de `https://www.rstudio.com/`, y también es gratis (asegúrate de haber instalado R antes de instalar RStudio).



Usar R interactivamente y a través de scripts

Trabajando con datos

Objetos y asignaciones

R es un lenguaje interpretado, lo que significa que ejecuta nuestras instrucciones directamente, sin compilar un programa previo. Podemos usar R interactivamente a través de la consola:

```
1+2

## [1] 3
```

La mayoría de las operaciones y funciones en R no guardan el resultado de su ejecución. Por ejemplo, el resultado anterior (`3`) es calculado pero no puede ser reutilizado sin vol-

verse a calcular. Para lograr esto tenemos que asignar el resultado de la operación a un objeto:

```
x <- 1+2
```

1.3 R base vs. tidyverse

Una de las críticas más comunes que se hace a R es que hay muchas (¿demasiadas?) maneras de lograr un mismo resultado. Por ejemplo, R base tiene un sistema propio para crear gráficos, pero también existe **ggplot2** para lograr el mismo objetivo. Esta multiplicidad de métodos puede hacer que R sea más difícil de aprender que lo necesario.

R es un lenguaje antiguo, y muchas de sus funcionalidades básicas —lo que llamamos **R base**— operan bajo paradigmas anticuados. Uno de los más grandes aportes a la programación en R del útltimo tiempo fue realizado por Hadley Wickham, autor del **tidyverse**. Este es un conjunto de paquetes que modernizan cómo usamos R para manipular datos, y están diseñados para trabajar bien entre ellos.

En un artículo para R-bloggers[1], David Robinson resume los dos principales "curriculums" de aprendizaje de R:

- **R base primero**: enseñar elementos de sintaxis como `$` y `[[]]`, loops, condicionales, tipos de datos y funciones base como `tapply()`. Este enfoque no se concentra en un sólo marco de datos.
- **tidyverse primero**: comenzar usando **dplyr** para manipular marcos de datos y **ggplot2** para crear gráficos. Luego introducir rápidamente el uso de **tidyr** y **purrr**. Usar el operador `%>%` casi inmediatamente, pero dejar el uso de `$` y `[[]]` para más adelante. Este enfoque se concentra en un sólo marco de datos.

Mi opinión (y la de muchos otros) es que aprender a usar las herramientas del **tidyverse** es más fácil y más productivo. La filosofía detrás del **tidyverse** es similar a la de Python: "Debería haber una —y preferiblemente sólo una— forma obvia de lograr algo". Esta filosofía es buena al aprender un lenguaje de programación, ya que entrega consistencia y evita confusiones. Es por esto que en este documento prefiero usar herramientas del **tidyverse** cada vez que sea posible.

Por ejemplo, en R base existen al menos tres formas de crear una variable nueva a partir

[1] `https://www.r-bloggers.com/teach-the-tidyverse-to-beginners/`

de otra existente. Por otro lado, con el **tidyverse** hay una sóla forma de lograr esto, que además (a mi) me parece más legible:

```
# Agregar una variable con R base
mtcars$libras <- mt$wt * 1000
mtcars[["libras"]] <- mtcars[["wt"]] / 1000
mtcars[, "libras"] <- mtcars[, "wt"] / 1000

# Agregar una variable con el tidyverse
mtcars <- mtcars %>% mutate(libras = wt / 1000)
```

Sin embargo, también hay elementos de R base que son importantes de aprender; de hecho, muchas cosas en R no son posibles sin ellos. Iremos introduciendo estos elementos a medida que nos sean útil para el tema en cuestión.