Introducción a las finanzas quantitativas

Gabriel Cabrera G.

2018-07-26

Contents

List of Tables

List of Figures

Prefacio

Este es un apunte en proceso pensado en el curso de Finanzas I de Ingeniería Comercial de la Universidad de Chile, las aplicaciones en R son pensados con un enfoque pedagógico y cualquier comentario o sugerencias son bienvenidas.

La información de la sesión de R cuando se compila este apunte es la siguiente:

```
sessionInfo()
```

```
## R version 3.4.4 (2018-03-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.1 LTS
##
## Matrix products: default
## BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## locale:
##  [1] LC_CTYPE=es_CL.UTF-8
##  [2] LC_NUMERIC=C
##  [3] LC_TIME=es_CL.UTF-8
##  [4] LC_COLLATE=es_CL.UTF-8
##  [5] LC_MONETARY=es_CL.UTF-8
##  [6] LC_MESSAGES=es_CL.UTF-8
##  [7] LC_PAPER=es_CL.UTF-8
##  [8] LC_NAME=C
##  [9] LC_ADDRESS=C
## [10] LC_TELEPHONE=C
## [11] LC_MEASUREMENT=es_CL.UTF-8
## [12] LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets
## [6] methods   base
##
## other attached packages:
```

```
##  [1] bindrcpp_0.2.2  forcats_0.3.0   stringr_1.3.1
##  [4] dplyr_0.7.6     purrr_0.2.5     readr_1.1.1
##  [7] tidyr_0.8.1     tibble_1.4.2    ggplot2_3.0.0
## [10] tidyverse_1.2.1 gapminder_0.3.0
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.4 xfun_0.3         haven_1.1.2
##  [4] lattice_0.20-35  colorspace_1.3-2 htmltools_0.3.6
##  [7] yaml_2.1.19      rlang_0.2.1      later_0.7.3
## [10] pillar_1.3.0     glue_1.3.0       withr_2.1.2
## [13] modelr_0.1.2     readxl_1.1.0     bindr_0.1.1
## [16] plyr_1.8.4       munsell_0.5.0    gtable_0.2.0
## [19] cellranger_1.1.0 rvest_0.3.2      htmlwidgets_1.2
## [22] evaluate_0.11    labeling_0.3     knitr_1.20
## [25] httpuv_1.4.5     crosstalk_1.0.0  highr_0.7
## [28] broom_0.5.0      Rcpp_0.12.18     xtable_1.8-2
## [31] promises_1.0.1   scales_0.5.0     backports_1.1.2
## [34] DT_0.4           jsonlite_1.5     mime_0.5
## [37] hms_0.4.2        digest_0.6.15    stringi_1.2.4
## [40] shiny_1.1.0      bookdown_0.7     grid_3.4.4
## [43] rprojroot_1.3-2  cli_1.0.0        tools_3.4.4
## [46] magrittr_1.5     lazyeval_0.2.1   crayon_1.3.4
## [49] pkgconfig_2.0.1  xml2_1.2.0       lubridate_1.7.4
## [52] assertthat_0.2.0 rmarkdown_1.10   httr_1.3.1
## [55] rstudioapi_0.7   R6_2.2.2         nlme_3.1-131
## [58] compiler_3.4.4
```

Sobre el Autor

Aún no hay mucho que decir, pero te invito a visitar mi página web donde la mayoría del tiempo estoy publicando post sobre programación y/o data science con aplicaciones a las finanzas y al mundo real.

Me agradan las citas...

> "It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."
>
> Sir Arthur Conan Doyle

Chapter 1

Introducción

Antes de comenzar tú viaje a través del lenguaje de programación R, necesitaras cuatro "herramientas" básicas para trabajar con este apunte, R (lenguaje), Rstudio(IDE[1]), una megalibrería que contiene *R packages* llamada *tidyverse* y librerías extras para trabajar en finanzas[2].

1.1 R

1.1.1 Un poco de historia

R es un lenguaje de programación creado por Ross Ihaka y Robert Gentleman del departamento de estadística de la universidad de Auckland en 1992 (Nueva Zelanda), teniendo su versión estable el 29 de Febrero del 2000.

1.1.2 Los primeros pasos

El primer paso es descargar R, para esto debes ir a CRAN[3], *comprehensive R archive network*. CRAN esta compuesto por un conjunto de *mirror servers* distribuidos alrededor del mundo y se usa para compartir los *R packages*. Como recomendación no elegir un *mirror* lejano a tú posición geográfica, por ende, usa https://cloud.r-project.org que los seleccionará automáticamente.

Como aún no instalas RStudio solo tendras el lenguaje, que puede ser ejecutado desde el *command Shell* o *prompt*, no obstante, esto es ineficiente desde el punto de vista que no tendrenmos todas las opciones que Rstudio nos entrega.

[1] *Integrated Development Environment*

[2] Estas librerías las iremos cargando/utilizando a medida que avancemos en los capítulos

[3] Existe otra distribución de R por parte del area de *open source* de Microsoft , la ventaja de MRAN es que si bien funciona con CRAN, su objetivo va más orientado a computación en paralelo (paralleling computing)

```
knitr::include_graphics("images/R_logo.png")
```

Figure 1.1: logo de R

Si trabajas sin IDE veras algo como en la Figura 1.2.

```
knitr::include_graphics("images/r_propmt.png")
```

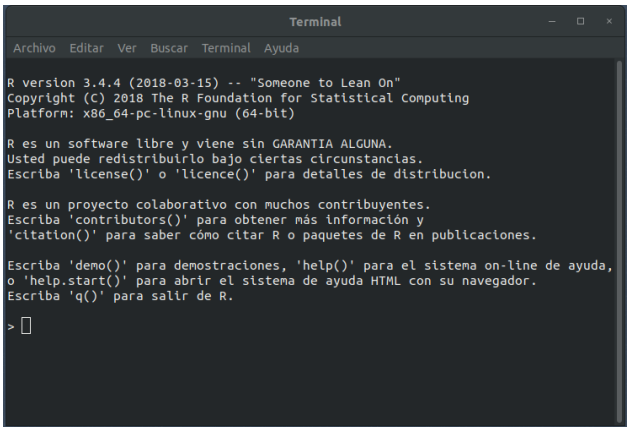


Figure 1.2: R desde la terminal

Las versiones de R cambian una vez al año, y de 2 a 3 veces con cambios pequeños, por eso es una buena idea que mantengas actualizada tu versión.

1.2 RStudio

¿Qué es una IDE? IDE es el acrónimo de *Integrated Development Environment (Entorno de Desarrollo Integrado)*. Esto quiere decir que RStudio es una aplicación que nos entrega herramientas para hacer más fácil el desarrollo de proyectos usando R y sobre todo cuando estemos trabajando con datos.

Para descargar e instalar Rstudio debes ir a http//www.rstudio.com/download y seleccionar *RStudio Desktop Open Source License* (gratuita) , cuando exista una actualización Rstudio te avisará.

Si quedó todo bien instalado, cuando abras Rstudio deberías ver algo así:

```
knitr::include_graphics("images/rstudio.png")
```

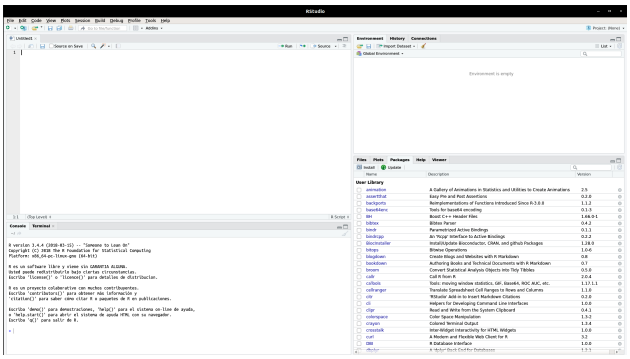


Figure 1.3: Rstudio

IMPORTANTE: Si te aparece algún error durante este proceso, lo más probable es que sea por alguna configuración de tu sistema operativo. En ese caso, la mejor manera de buscar una solución es copiar el error que arroja R, pegarlo en tu motor de búsqueda favorito y ver cómo alguien que se enfrentó a eso antes lo resolvió.

1.2.1 Partes de RStudio

¿Para qué sirven estos paneles? Comentemos primero el panel de abajo a la derecha. Si te fijas, el panel tiene varias ventanas:

1. **Files** muestra el directorio (la carpeta) en la que te encuentras actualmente. En mi caso, no hay nada ahí porque por defecto RStudio me muestra el Escritorio (Desktop) y no tengo nada en él. Es posible que a ti te muestre otra carpeta (por ejemplo, Documentos).

2. **Plots** es el lugar donde aparecerán los gráficos que vayas creando. No hemos hecho ninguno por ahora, así que este panel también está vacío.

3. **Packages** muestra la lista de paquetes que tienes instalados en tu computador. Si recorres el panel verás que algunos tiene una marca al lado izquierdo. Eso quiere decir que el paquete está activo en ese momento (ya veremos cómo hacer eso). Solo los paquetes vinculados a R base se activan al abrir RStudio.

4. **Help**, como su nombre lo indica, es la pestaña en la que podemos encontrar ayuda. Si buscamos el nombre de un paquete o de una función, RStudio nos remitirá a la documentación asociada.

5. **Viewer** es el panel para ver contenido web generado por algún paquete de R (gráficos para la web o aplicaciones interactivas). Por el momento no lo utilizaremos.

El panel de arriba a la derecha, por su parte, contiene el historial de funciones que hemos ejecutado (History), la opción para generar conexiones a bases de datos externas (Connections) y el Environment. Este último panel es muy importante y entender lo que nos muestra es fundamental para comprender cómo funciona R.

1.3 *Scripts*

El script podemos decir que es un cuarto panel[4], en donde escribiras tus códigos que queremos que ejecute R, para crear un script debes:

1. ir a `file > New File > R Script`
2. Otra forma es un atajo de teclado, `control + shift + n` (Linux/Windows) y `comando + shift + n` (Mac OS).
3. O bien ir a la barra superior de la ventana y seleccionar el tipo de archivo a trabajar.

1.4 Proyectos

Una de las ventajas de RStudio es que permite crear "proyectos". Un projecto es un espacio o contexto de trabajo asociado a una carpeta en particular, en la que se guardan nuestro(s) script(s), archivos de datos, etc. Cuando creamos un proyecto en RStudio, se crea un tipo especial de archivo (.Rproj) que lo que hace es vincular todo lo que se encuentra dentro de esa carpeta.

1.4.1 ¿Por qué esto es útil?

Si parte de nuestro script, por ejemplo, implica abrir un archivo que está en la carpeta de nuestro proyecto, no necesito indicar en mi código toda la ruta del archivo: lo que hará RStudio será buscarlo en el entorno/carpeta del proyecto. Si movemos la carpeta a otro lugar de nuestro computador o la compartimos con otra persona, nuestro código seguirá funcionando, ya que el archivo .Rproj mantendrá todo unido. Si no creara un proyecto, tendría que indicar al inicio de mi script cuál es la ruta de la carpeta que ocuparé como espacio de trabajo. El problema de esa opción es que si muevo la carpeta o le cambio el nombre, tendría que volver a escribir la ruta para que todo funcione. Al crear un proyecto eso deja de ser una preocupación.

[4] El tercer panel es la consola

1.4.2 ¿Cómo crear un projecto?

1. Puedes hacerlo desde el menú File > New Proyect.
2. Lo primero que nos pregunta es si queremos crearlo en una carpeta nueva o en una ya existente. Elegiremos esta vez una carpeta nueva, así que seleccionaremos New Directory.
3. La siguiente pregunta es qué tipo de proyecto queremos crear. En esta ocasión, elegiremos la primera: *New Project*.
4. Finalmente, le damos un nombre al proyecto y decidimos en qué parte de nuestro computador queremos que viva la carpeta que lo contiene.
5. Luego de apretar Create Project, RStudio se reinicia y se producen algunos cambios. El panel Files (abajo a la derecha) ahora nos muestra la carpeta de nuestro proyecto y el único archivo que hay en ella por ahora. Ese es el archivo mágico que mantiene unido todo lo que hay dentro de la carpeta. Cuando queramos volver a trabajar en nuestro proyecto, solo tenemos que abrir ese archivo.

> IMPORTANTE: RStudio ejecuta sesiones independientes de R para cada proyecto. Es decir, si tuvieras otro proyecto abierto te aparecería otro ícono, con su respectivo nombre. Esto nos permite trabajar en dos proyectos en paralelo sin que se nos mezclen los objetos del entorno, el código, los archivos, etc. Cada cosa en su lugar.

1.5 *Packages*

Cuando instalamos R por primera vez en nuestro computador, lo que estamos instalando es lo que se conoce como "R Base", es decir, los elementos centrales del lenguaje de programación. Una de las ventajas de R es que se trata de un lenguaje extensible: la propia comunidad de usuarios puede desarrollar nuevas posibilidades para utilizarlo. La manera de compartir estos nuevos desarrollos es a través de "paquetes", que incluyen, entre otras cosas, código y datos. Una analogía que se suele utilizar para explicar esto es que R Base es un teléfono celular tal como viene de fábrica y los paquetes las apps que descargamos para que tenga más funcionalidades.

Para usar las librerias ("packages") debemos usar el siguiente código:

```
# instala el package
install.packages("acá va el package")

# lo llama
library("acá va el package")
```

1.5.1 *tidyverse*

tidyverse es un "megapaquete" que incluye otros paquetes en su interior. Todos los paquetes que conforman "el Tidyverse" comparten la misma visión sobre el trabajo con datos y la escritura de código. Viene a formar parte de la nueva forma de programar en R, cuyo enfoque es netamente en realizar Data Science. Algunas librerias relevantes son:

1. ggplot2: Esta librería te permite realizar graficos avanzados.
2. dplyr: Su objetivo es la manipulación de datos (filtrar, seleccionar, generar, renombrar,etc).
3. magrittr: Contiene la denominada *pipe* (%>%), se explicará más adelante
4. purrr: Para realizar iteraciones.
5. readr: Para cargar datos en csv, lo importante que los transforma en tibble.

Para instalarlo basta escribir:

```
# instala el package
install.packages("tidyverse")

# lo llama
library("tidyverse")
```

Si te vas a la pestaña de *Packages* verás que estan seleccionadas aquellas librerias que se encuentran en el tidyverse.

1.6 Ejemplo

Este paquete contiene una parte de los datos de Gapminder, una base de datos que incluye información mundial sobre población, expectativa de vida, PIB per cápita y otros. Su autor, Hans Rosling, ha hecho varias charlas TED que vale la pena mirar.

```
# instala el package
install.packages("gapminder")
```

```
# lo llama
library("gapminder")
library("tidyverse")
```

```
world.data <- gapminder

mean.lifeExp <- world.data %>%
                filter(year == 2007) %>%
                group_by(continent) %>%
                summarize(mean(lifeExp))
```

```
amean.lifeExp
```

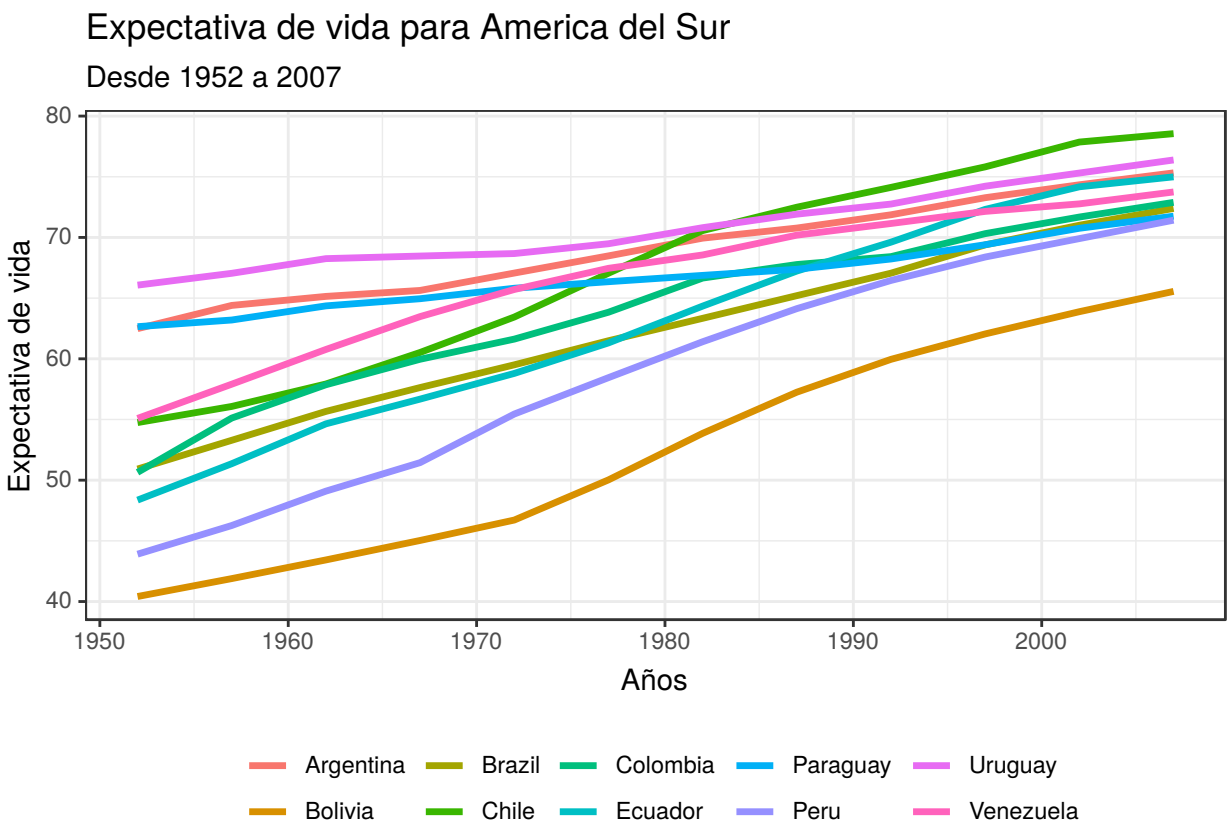
Show 10 entries Search:

	continent	mean(lifeExp)
1	Africa	54.8060384615385
2	Americas	73.60812
3	Asia	70.7284848484849
4	Europe	77.6486
5	Oceania	80.7195

Showing 1 to 5 of 5 entries Previous 1 Next

```
world.data <- world.data %>%
              filter(country %in% c("Argentina", "Bolivia", "Brazil", "Chile", "Colombia

g1 <- ggplot(world.data) + geom_line(mapping = aes(year,lifeExp, colour= country), size
g1 <- g1 + theme_bw() + labs(title = "Expectativa de vida para America del Sur", subtitl
g1 <- g1 + xlab("Años") + ylab("Expectativa de vida")
g1 <- g1 + theme(legend.position="bottom")
g1
```




```
ggsave("Grafico.png")
```