Introducción a R

Luciano Selzer

28 June, 2018

¿Qué es R?

R es:

- Es un lenguaje de programación
  - Interactivo
  - Programática
- Software Libre
- Muchos usuarios
  - Nuevas técnicas
  - Infinidad de usos

Usos

- Análisis de datos experimentales
  - Regresiones: Lineal, GLM, GLMM, GAM
  - Análisis Multivariado: CCA, PCA
  - Análisis no Paramétrico
  - Text Mining
- Usado en:
  - Laboratorios de Investigación
  - Agencias Gubernamentales
  - Empresas

¿Qué se puede hacer?

- Realizar pre-procesado de datos
- Analizar datos
- Crear aplicaciones web
- Realizar reportes dinámicos
- Esta presentación
- Crear nuestras propias rutinas o algoritmos

Prerrequisitos

R

Para bajarlo ir a https://cran.r-project.org/,

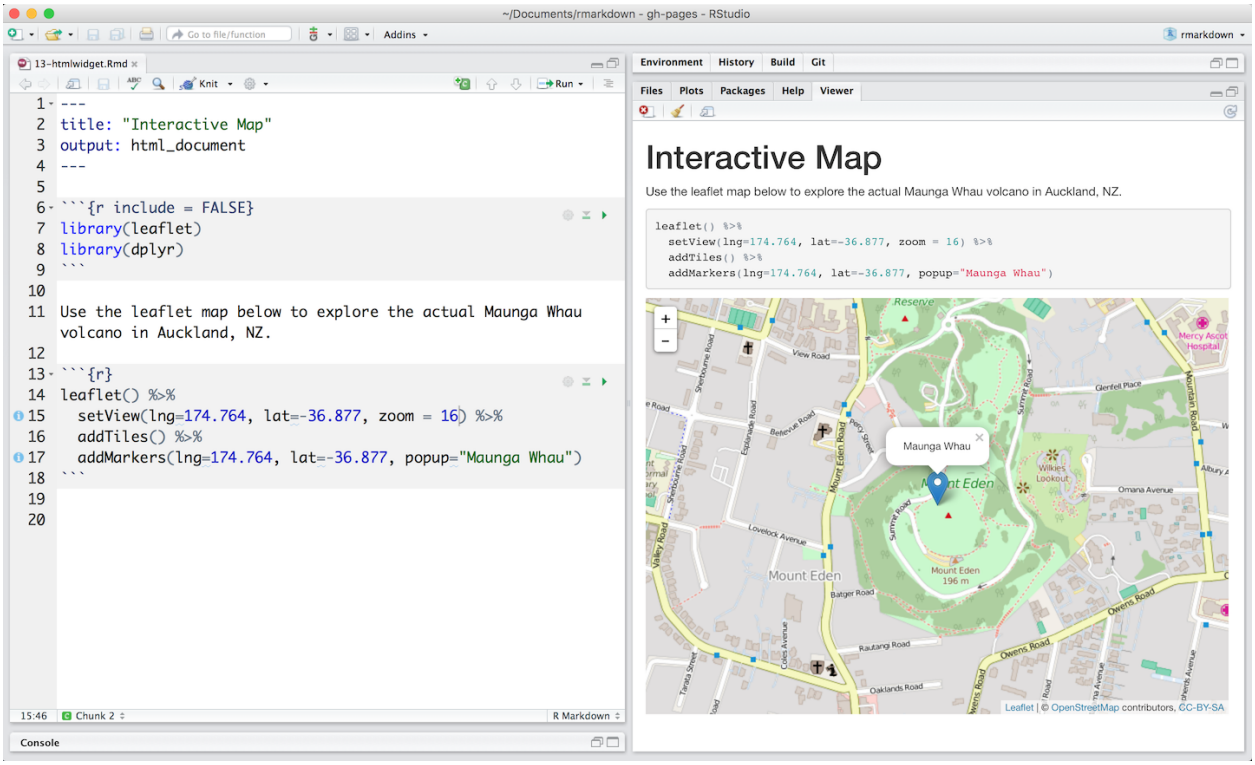

Figure 1: interactive widget

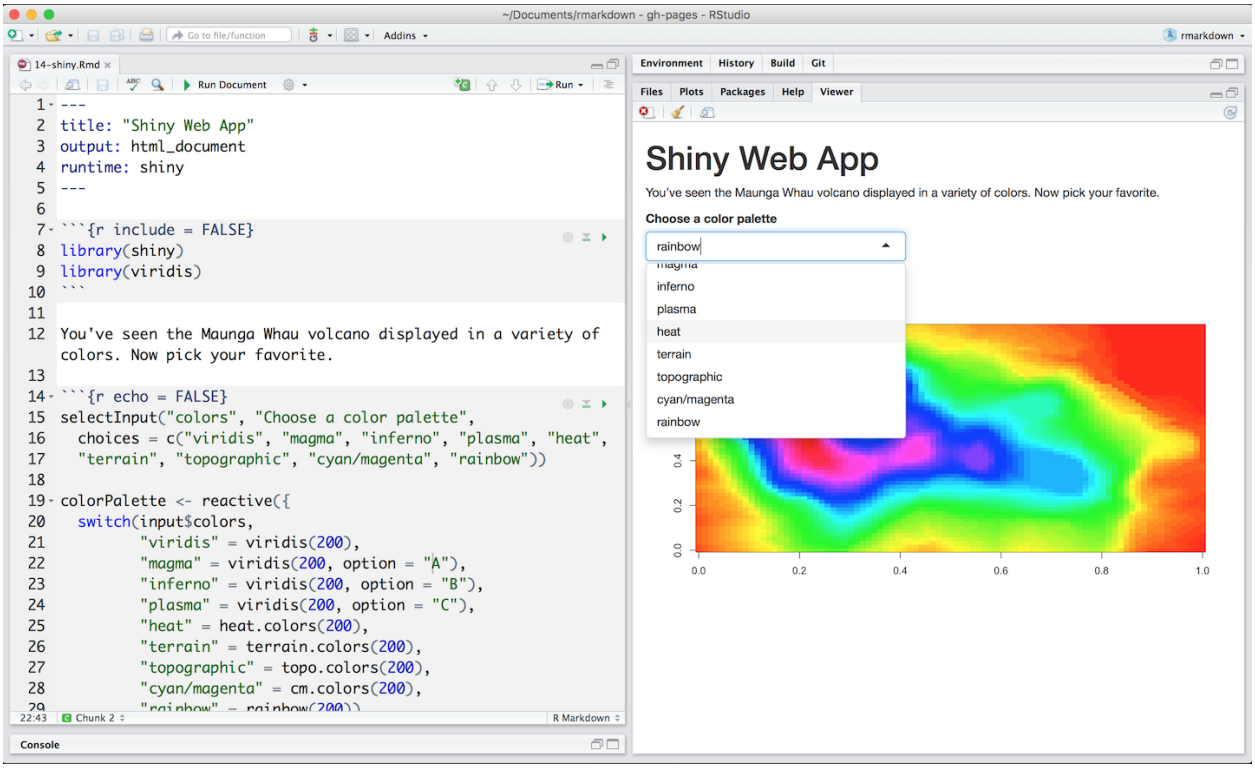

Figure 2: interactive widget

GIT

Control de Versión
https://git-scm.com/

RStudio

RStudio Desktop https://www.rstudio.com/products/rstudio/download2/

Crear Cuenta en GitHub

https://github.com/

- Completar el formulario
- Presionar *Sign Up for GitHub*.
- Verificar cuenta de correo

Configuración de RStudio

Para esto hay que:

- abrir RStudio
- ir al menú `Tools\Global Options`
- luego ir a `Git\SVN`, el anteúltimo icono.

Si utilizaron la instalación por defecto de Git la ruta es `C:\Program Files\Git\bin` si tienen Windows 32 bits o `C:\Program Files (x86)\Git\bin` si tienen Windows 64 bits. La carpeta `Program Files` puede llamarse `Archivos de Programa` si tienen Windows en español.

- Presionar `Ok`.

Configurar GitHub

En Rstudio:

1. Vayan a `Tools\Global Options...`, luego a la pestaña de `Git\SVN`.
2. Ahí deben hacer clic en `create RSA key`. Pueden poner una `Passphase`.
3. Presionan `Create`, les va a aparecer un cartel y ponen `Ok`.
4. Luego, presionan en `View public key`. Copian la clave.

En el sitio web de Github:

1. En la esquina superior derecha hay un icono que es su foto de usuario. Hacen clic ahí y luego en `Settings`.
2. Van la pestaña `SSH and GPG keys`.
3. Luego, presionan en `New SHH Key`.
4. En *Title* ponen el nombre del programa que lo va a usar. **RStudio** en nuestro caso.
5. En *Key* pegan lo que han copiado de RStudio.
6. Finalmente, presionan `Add SSH Key`

Introducción

Introducción a RStudio

- Interfaz de desarrollo integrada (IDE)
- gratis
- código abierto.
- Editor de código,
- Múltiples plataformas (incluyendo servidores)
- Muchas ventajas:
  - Integración con control de versión
  - Manejo de proyectos

Disposición Básica

- La consola interactiva de R (a la izquierda)
- El Entorno/Historial, en inglés Environment/History (en pestañas en la parte superior derecha)
- Archivos/Gráficos/Paquetes/Ayuda/Visualizador, también en inglés Files/Plots/Packages/Help/Viewer (en la parte inferior derecha)

Una vez que abre un archivo, como por ejemplo un script de R, un panel de edición se abre en la parte superior izquierda.

Flujo de trabajo con RStudio

Hay dos maneras básicas de trabajar con RStudio.

1. Probar y jugar dentro de la consola interactiva y luego copiar el código a un archivo .R para ejecutar más tarde.
   - Funciona bien cuando se hacen pequeñas pruebas y recién se comienza.
   - Pero rápidamente se vuelve laborioso.
2. Empezar a escribir un archivo .R y luego usar el comando / atajo de RStudio para ejecutar la línea actual o las líneas seleccionadas en la consola.
   - Es la mejor forma de empezar; así todo el código queda guardado para más tarde
   - Podrás ejecutar el archivo desde RStudio o usando la función de R `source()`.

Tip: Ejecutando partes de tu código

1. hacer click en el botón `Run` arriba del panel de edición,
2. Seleccionar “Run Lines” desde el menú “Code”, o
3. apretar Ctrl+Enter en Windows o Linux o Command+Enter en OS X (Este atajo también puede verse dejando el puntero del mouse sobre el botón `Run`).

Para ejecutar un bloque de código: + Seleccionálo y después solo presiona `Run`.

Si has modificado una parte del bloque que has ejecutado recién podés usar el botón a continuación, `Re-run the previous region`.

Introducción a R

La consola interactiva

Read, Eval, Print

R como calculadora

```
1 + 100
```

[1] 101

Tip: Cancelando comandos

Si estás usando R desde la linea de comando en vez de dentro de RStudio, hay que presionar `Ctrl+C` en vez de `Esc` para cancelar el comando. Esto también se aplica a usuarios de Mac también.

Cancelar un comando no es útil solamente para anular comandos incompletos: también podés cancelar una computación que R está ejecutando (por ejemplo, si está tomando demasiado tiempo), o para eliminar el código que estás escribiendo actualmente.

Orden de ejecución

De mayor a menor precedencia:

- Paréntesis: `(`, `)`
- Exponentes: `^` o `**`
- División: `/`
- Multiplicación: `*`
- Adición: `+`
- Substracción: `-`

Orden de Ejecución

```
(3 + (5 * (2 ^ 2))) # difícil de leer
3 + 5 * 2 ^ 2       # claro, si recordás las reglas
3 + 5 * (2 ^ 2)     # si olvidás algunas reglas esto puede ayudar.
```

Notación Cientifica

Los números muy grandes o pequeños se imprimen en notación científica:

```
2/10000
```

```
[1] 2e-04
```

que es una abreviatura de multiplicar por `10^XX"`. Por lo que `2e-4` es una abreviatura de `2 * 10^(-4)`.

Uno mismo puede entrar números en notación científica:

```
5e3 #No hay menos aquí
```

```
[1] 5000
```

Funciones matemáticas

R tiene muchas funciones matemáticas incorporadas. Para llamar una función simplemente escribimos su nombre, seguido por paréntesis de apertura y cierre. Cualquier cosa dentro de los paréntesis es llamado argumentos de la función:

```
sin(1)  # funciones trigonométricas
```

```
[1] 0.841471
```

Ayuda Memoria

No se preocupen por recordar todas las funciones en R. Se pueden buscar en Google, o si recuerdan como comienza el nombre de la función, pueden presionar la tecla `Tab` para usar la función de autocompletado de RStudio.

Escribiendo un signo de pregunta `?` antes del nombre la funcion abrirá la página de ayuda de esa función.

Comparando cosas

También podemos hacer comparaciones en R:

```
1 == 1 # igualdad (notar que son dos signos igual, se lee como "es igual a")
```

```
[1] TRUE
```

Operador	Comparación
==	igual
!=	distinto,
>	mayor
<	menor
>=	mayor o igual
<=	menor o igual

Tip: Comparando números

Un advertencia acerca de la comparación de números: nunca usar `==` para comparar dos números a menos que sean integers (un tipo de data que se usa para representar números enteros específicamente).

Las computadoras solo pueden representar números decimales con un cierto grado de precisión. Por lo que dos números que cuando son impresos por R se ven iguales, pueden tener representaciones subyacentes diferentes y por lo tanto ser diferentes por un pequeño margen de error (llamado *Machine numeric tolerance*)

En su lugar, deberías usar la función `all.equal`

Para más información, leer: http://floating-point-gui.de/

Variables y asignación

Podemos guardar los valores en variables usando el operador de asignación `<-`, de esta forma:

```
x <- 1/40
```

La asignación de valores puede contener la variable que esta siendo asignada:

```
x <- x + 1 # notar como RStudio actualiza la descrición de x en la pestaña Environment
```

Reglas para nombrar variables

Los nombres de la variables pueden contener letras, números, guiones bajos y puntos.

No pueden empezar con un número o guión bajo ni pueden contener espacios.

Distintas personas usan distintas convenciones para nombrar variables con nombres largos, incluidos:

- puntos.entre.palabras
- guiones_bajos_entre_palabras
- camelCaseParaSepararPalabras

Cual usar es una decisión individual (o del proyecto), pero hay que ser **consistente**

Vectorización

Una consideración final de lo que hay estar atento es que R está *vectorizado*, lo que significa que las variables y funciones pueden tener vectores como valores. Por ejemplo:

```
1:5
```

```
[1] 1 2 3 4 5
```

```
2^(1:5)
```

```
[1]  2  4  8 16 32
```

```
x <- 1:5
2^x
```

```
[1]  2  4  8 16 32
```

Manejando el entorno

Hay algunas funciones muy útiles para interactuar con la sesión de R.

`ls` listará todas los objetos (funciones y variables) almacenados en el entorno global (tu sesión de trabajo en R)

Tip: objetos ocultos

`ls` ocultará cualquier objeto cuyo nombre empiece con un punto "." por defecto. Para listar todos los objetos, hay que tipear `ls(all.names=TRUE)`.

Manejando el entorno

Podés usar `rm` para eliminar objetos que no sean necesarios:

```
rm(x)
```

Si tenés muchos objetos en tu entorno y querés borrar todos ellos, podés pasar el resultado de `ls` como argumento de la función `rm`:

```
rm(list = ls())
```

En este caso hemos especificado que el resultado de `ls` sea usado por el argumento `list` en `rm`. Cuando asignamos valores a argumentos por nombre, **¡hay que** usar el operardor `=`!

Tip: Warnings vs Errors

¡Presten atención cuando R hace algo inesperado! R tira errores, como el de arriba, cuando no puede proceder con un cálculo. Los *Warnings* (advertencias) por otro lado generalmente significa que la función se ejecutó, pero que probablemente no ha funcionado como se esperaba.

En ambos casos, el mensaje que imprime R generalmente contiene pista de como arreglar el problema. Aunque a veces pueden ser algo crípticos y requieran cierta experiencia saber que es lo que está pasando.

Paquetes R

Los paquetes de R son colecciones de funciones o a veces datos, que son justamente empaquetadas. Añaden funciones que no están presentes en R.

- Para ver que paquetes están instalados tipear: `installed.packages()`
- Para instalar paquetes tipear `install.packages("packagename")`, donde `packagename` es el nombre del paquete, entre comillas.
- Para actualizar los paquetes instalados, tipear `update.packages()`
- Para eliminar un paquete tipear `remove.packages("packagename")`
- Para usar un paquete en la sesión actual tipear `library(packagename)`

Ejercicios

Ejercicio 1

¿Cuál de los siguientes nombres son válidos como nombre de variable?

```
min_height
max.height
_age
.mass
MaxLength
min-length
2widths
celsius2kelvin
```

Ejercicio 2

¿Cuál será el valor de cada variable luego cada comando en el siguiente código?

```
mass <- 47.5
age <- 122
mass <- mass * 2.3
age <- age - 20
```

Ejercicio 3

Ejectuta el código anterior y escribe un código para comparar mass y age ¿Es mass más grande que age?

Ejercicio 4

Limpia el entorno eliminando las variables mass y age.

Ejercicio 5

Instala los paquetes: `ggplot2`, `plyr`, `gapminder`