What is R?

The R Bootcamp
Twitter: @therbootcamp

September 2017

R

From Wikipedia (emphasis added):

> R is an **open source programming language** and software environment for **statistical computing and graphics** that is supported by the R Foundation for Statistical Computing. The R language is **widely used among statisticians and data miners** for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that **R's popularity has increased substantially in recent years**.
>
> R is a GNU package. The source code for the R software environment is written primarily in **C, Fortran, and R**. R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. While R has a command line interface, there are several **graphical front-ends available**.

Programming language

From Wikipedia (emphasis added):

> A programming language is a **formal language** that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of **instructions for a computer**. Programming languages can be used to create programs that **implement specific algorithms**.

Algorithm

1. Load data
2. Extract variables
3. Run analysis
4. Print result

Implementation in R

```r
data <- read.table(link)
variables <- data[,c('group','variable')]
analysis <- lm(variable ~ group, data = variables)
summary(analysis)
```

R is purpose specific

R has been build for **statistical computing and graphics** and that is basically it:

Use R for...

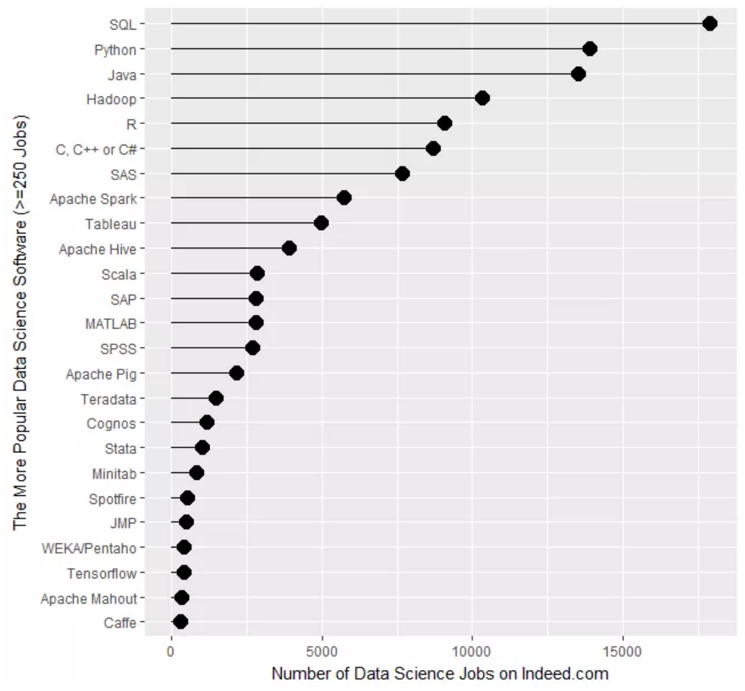
1. Loading and handling data
2. Run statistical analyses
3. Run analyses
4. Prepare reproducible reports

Don't use R for...

1. OS programs
2. GUIs
3. (Dynamic) Websites
4. Behvioral experiments

R is widely used

R steadily **grows in popularity**. Today, R is one of the **most popular languages for data science** and overall. In terms of the number of data science jobs, **R beats SAS and Matlab**, and is on par with Python:



source: https://i0.wp.com/r4stats.com/

R is so popular because

Although R has been implemented in **C, Fortran, and R**, R is often a slow and inefficient language. Yet, R's there are many good reasons to use R.

Pro

1. **It's free**
2. Relatively **easy**
3. **Extensibility** (CRAN, packages)
4. **User base** (e.g., stackoverflow)
5. **Tidyverse** (`dplyr`, `ggplot`, etc.)
6. **RStudio**
7. **Producitivity** options: Latex, Markdown, GitHub

Con

1. Slow and wordy
2. Limited (no iterators, pointers, etc.)

$\rightarrow$ Rcpp, rPython

RStudio: R's favorite environment

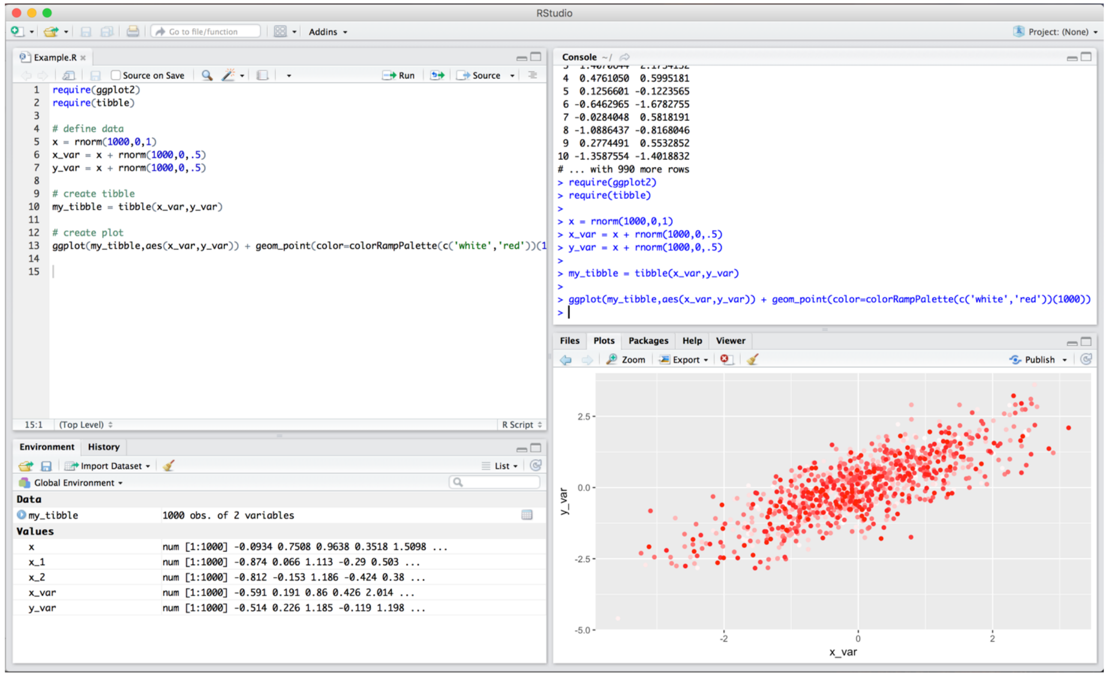
Next to many useful packages, R users greatly benefit from R's integrated development environment **RStudio**. Rstudio is a **graphical user interface** that allows you to (a) edit code, (b) run code, (c) access files and progress, and (d) create plots. In addition RStudio helps you with **version control** via Github, to write **reports** using markdown and knitr, integrating **C++** into R, writing **clean code**, and to **debug** code.

Script editor

This is where you write your code.

Environment & History

Here you can track what you have done.



```r
require(ggplot2)
require(tibble)

# define data
x = rnorm(1000,0,1)
x_var = x + rnorm(1000,0,.5)
y_var = x + rnorm(1000,0,.5)

# create tibble
my_tibble = tibble(x_var,y_var)

# create plot
ggplot(my_tibble,aes(x_var,y_var)) + geom_point(color=colorRampPalette(c('white','red'))(1

```

```
 4  0.4761050  0.5995181
 5  0.1256601 -0.1223565
 6 -0.6462965 -1.6782755
 7 -0.0284048  0.5818191
 8 -1.0886437 -0.8168046
 9  0.2774491  0.5532852
10 -1.3587554 -1.4018832
# ... with 990 more rows
> require(ggplot2)
> require(tibble)
>
> x = rnorm(1000,0,1)
> x_var = x + rnorm(1000,0,.5)
> y_var = x + rnorm(1000,0,.5)
>
> my_tibble = tibble(x_var,y_var)
>
> ggplot(my_tibble,aes(x_var,y_var)) + geom_point(color=colorRampPalette(c('white','red'))(1000))
> 
```

```
Data
my_tibble     1000 obs. of 2 variables
Values
  x           num [1:1000] -0.0934 0.7508 0.9638 0.3518 1.5098 ...
  x_1         num [1:1000] -0.874 0.066 1.113 -0.29 0.503 ...
  x_2         num [1:1000] -0.812 -0.153 1.186 -0.424 0.38 ...
  x_var       num [1:1000] -0.591 0.191 0.86 0.426 2.014 ...
  y_var       num [1:1000] -0.514 0.226 1.185 -0.119 1.198 ...
```

Console

This is where you talk to R. Here you run your code.

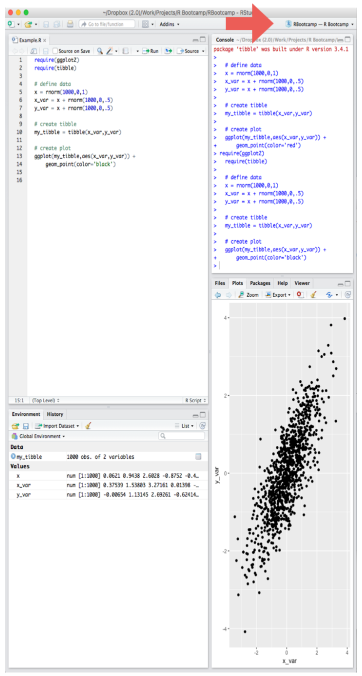
Plot, Help, Files, etc.

This window pane is mostly used for plotting and help files.

Project management

RStudio facilitate project management via the use of *projects*. Projects support:

1. **File management** by automatically setting the working directory (see `setwd()`)
2. **Project transitioning** by saving re-opening scripts, history, and workspace.
3. **Customization** by enabling project specific settings.
4. **Version control** by linking projects to repositories (e.g., using GitHub)

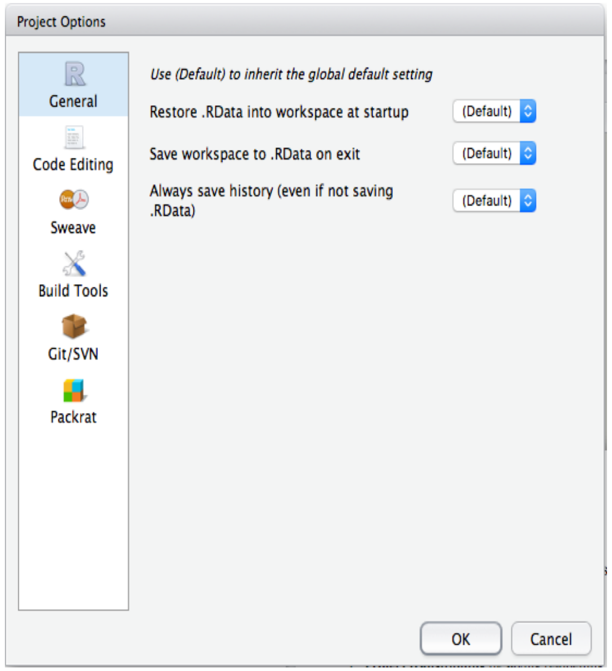


```
require(ggplot2)
require(tibble)

# define data
x = rnorm(1000,0,1)
x_var = x + rnorm(1000,0,.5)
y_var = x + rnorm(1000,0,.5)

# create tibble
my_tibble = tibble(x_var,y_var)

# create plot
ggplot(my_tibble,aes(x_var,y_var)) +
    geom_point(color='black')
```



Use (Default) to inherit the global default setting

Restore .RData into workspace at startup: (Default)

Save workspace to .RData on exit: (Default)

Always save history (even if not saving .RData): (Default)

The almighty **tidyverse**

Among its many packages, R contains a collection of high-performance, easy-to-use packages (libraries) designed specifically for handling data know as the tidyverse. The tidyverse includes:

1. `ggplot2` -- creating graphics.
2. `dplyr` -- data manipulation.
3. `tidyr` -- tidying data.
4. `readr` -- read wild data.
5. `purrr` -- functional programming.
6. `tibble` -- modern data frame.

Essentials of the R language

> "To understand computations in R, two slogans are helpful:
>
> (1) Everything that exists is an object and
>
> (2) everything that happens is a function call."

John Chambers

Author of S and developer of R

statweb.stanford.edu

Calls, assignments, and expressions

In R every action is a function call. Specifically, R programs advance by **passing on arguments to functions**, **calling the function**, and **receiving and storing its output**. And this goes deep, many operations are functions in disguise.

```r
# defining a function - arithm. mean
my_fun <- function(x, b){ x * b }

# define some data
my_data <- c(1, 5, 7, 3)

# pass on arguments and call function
my_fun(my_data, 5)
```

```
## [1]  5 25 35 15
```

```r
# store output by assignment
my_out <- my_fun(my_data, 5)
```

```r
# a basic expression
2 + 2
```

```
## [1] 4
```

```r
# is also a function
'+'(2,2)
```

```
## [1] 4
```

Object-orientation

R is an object-oriented language. This means that for R that **everything is an object** (including functions). This also means that there are several **generic functions** that respond to the **object's class**. Another important feature of R regarding objects is that R **always copies deep**. This is why practically everything in R is an assignment.

```r
# creating a vector and testing its class (type of object)
my_vector <- c(1, 5, 2)
print(class(my_vector))
```

```
## [1] "numeric"
```

```r
# testing the class (aka object type) of an object
print(my_vector)
```

```
## [1] 1 5 2
```

```r
# Sorting
sort(my_vector)
```

```
## [1] 1 2 5
```

Syntax style

Every language has a specific expressive style. R is characterized by the following elements…

- Comment symbol #
- Quotations with either "" or ''
- Curly brackets `{}` enclose expressions explicitly
- Parentheses `()` call functions
- Semicolon `;` separates expressions
- <,>,|,&,==, `!=` define logical statements

```
# This is a comment

# Quotes are used to define strings
"a" == 'a'

# Expression and calls
my_fun(x,y){ x  + y }

# two expression in one line
2 + 2 ; 3 + 3

# are these equal/different
2 == 2 ; 2 != 2
```

Help

An facilitator for using R are **help files** and **vignettes**. Help files are required documentations for every R function and package published on **CRAN**. Don't worry if help files may appear cryptical, however, over time you will realise how helpful they are. **Vignettes** are long tutorials sometimes provided by the authors of a package.

```
# To access help files
help("name_of_function")
?name_of_function

# find help files
??name_of_function

# To list and access vignettes
vignette(package="name_of_package")
vignette(package="name_of_vignette")
```

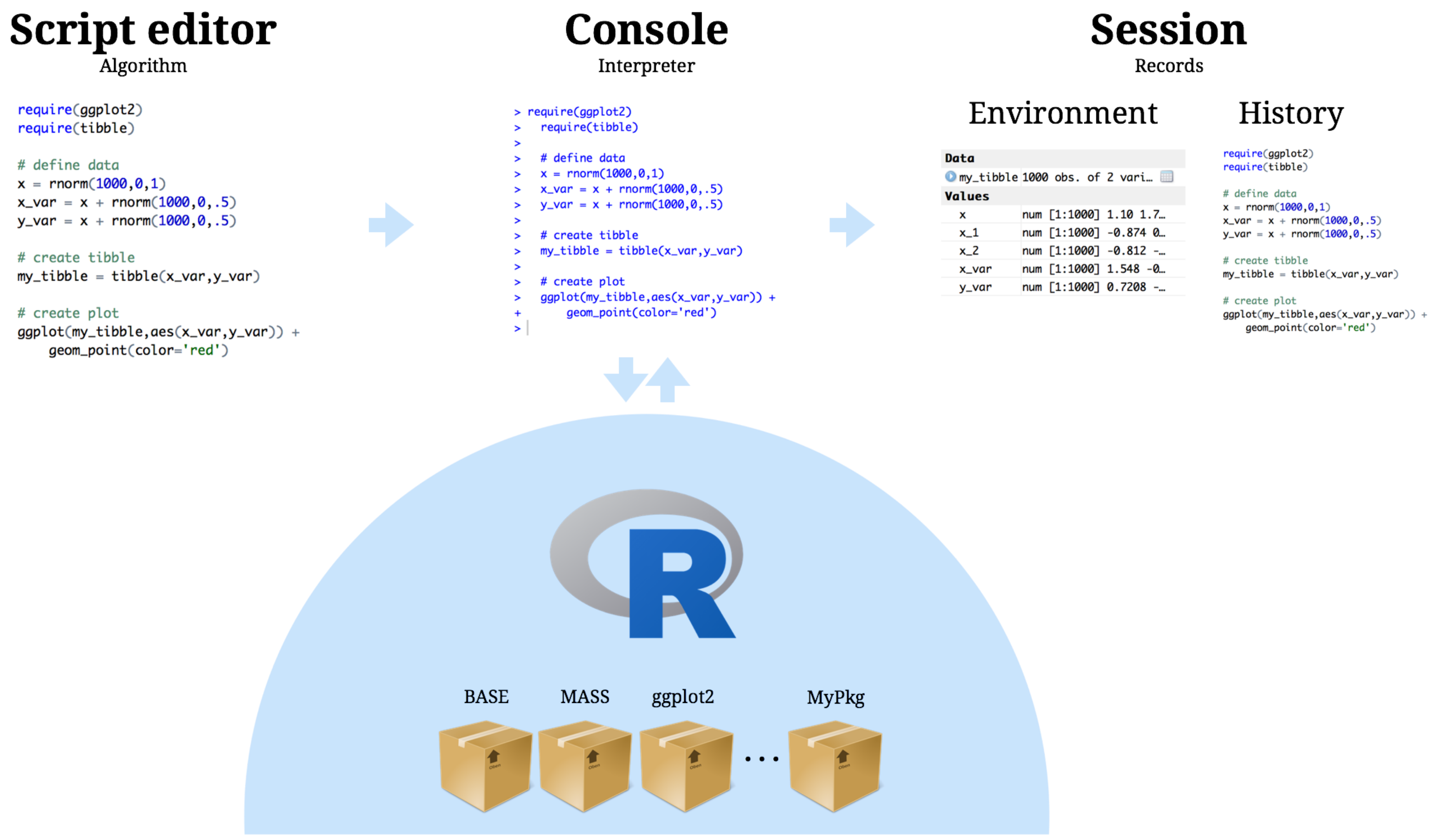
Packages

One of the huge benefits of R is its vast and cutting-edge collection of **packages**. Responsible for this is R's large and active user base, but also the **CRAN**, who examine every package, apply a rigorous quality control, and eventually host the packages on various mirrors throughout the world. Note: when downloading one of the many packages never forget that the package must also be loaded.

```r
# To install a package
install.packages('package_name')

# load a package
library(package_name)
require(package_name)
```

The workflow of R



```
require(ggplot2)
require(tibble)

# define data
x = rnorm(1000,0,1)
x_var = x + rnorm(1000,0,.5)
y_var = x + rnorm(1000,0,.5)

# create tibble
my_tibble = tibble(x_var,y_var)

# create plot
ggplot(my_tibble,aes(x_var,y_var)) +
    geom_point(color='red')
```

```
> require(ggplot2)
>   require(tibble)
>
>   # define data
>   x = rnorm(1000,0,1)
>   x_var = x + rnorm(1000,0,.5)
>   y_var = x + rnorm(1000,0,.5)
>
>   # create tibble
>   my_tibble = tibble(x_var,y_var)
>
>   # create plot
>   ggplot(my_tibble,aes(x_var,y_var)) +
+       geom_point(color='red')
> |
```

```
Data
my_tibble 1000 obs. of 2 vari…
Values
  x       num [1:1000] 1.10 1.7…
  x_1     num [1:1000] -0.874 0…
  x_2     num [1:1000] -0.812 -…
  x_var   num [1:1000] 1.548 -0…
  y_var   num [1:1000] 0.7208 -…
```

```
require(ggplot2)
require(tibble)

# define data
x = rnorm(1000,0,1)
x_var = x + rnorm(1000,0,.5)
y_var = x + rnorm(1000,0,.5)

# create tibble
my_tibble = tibble(x_var,y_var)

# create plot
ggplot(my_tibble,aes(x_var,y_var)) +
    geom_point(color='red')
```

Interactive session

Open up **Rstudio**…

Link to practical