Introduction to R

author: Maria Paniw date: January 25, 2016; UCA

History of R

- R is a dialect of the statistical languages S and S-PLUS, developed in the 1990s
- R is essentially an itegrated environment for data manipulation, statistical analysis, and graphical output
- Most people use R for the graphics
- The R Core Group exists to control the source code
- You can donate to R
- Main difference between R and SAS or SPSS: R saves much of its output as objects that are called by additional functions and do not automatically appear on screen

Some good reason to use R

- **R is extremely poweful**
- **R is free** - in the sense that you can download and distribute it for free
- **R is free** - in the sense that you can manipulate the source code
- **R runs well on many computer platforms** - even on Play Station 3...
- **R is rapidly gaining popularity** - active development with frequent releases
- **R has a powerful graphic interface** - many plotting packages and functions

Some problems with R

- **R depends on command line code** - this by itself is an advantage, as we don't have to point-and-click; However, inefficient code may result in very slow analyses
- **3D graphics** - doing 3D graphic in R is truly painful (although there are ongoing improvements)
- **Memory** - Most results of R functions are stored in physical memory. Working with large data sets, this can become a problem.

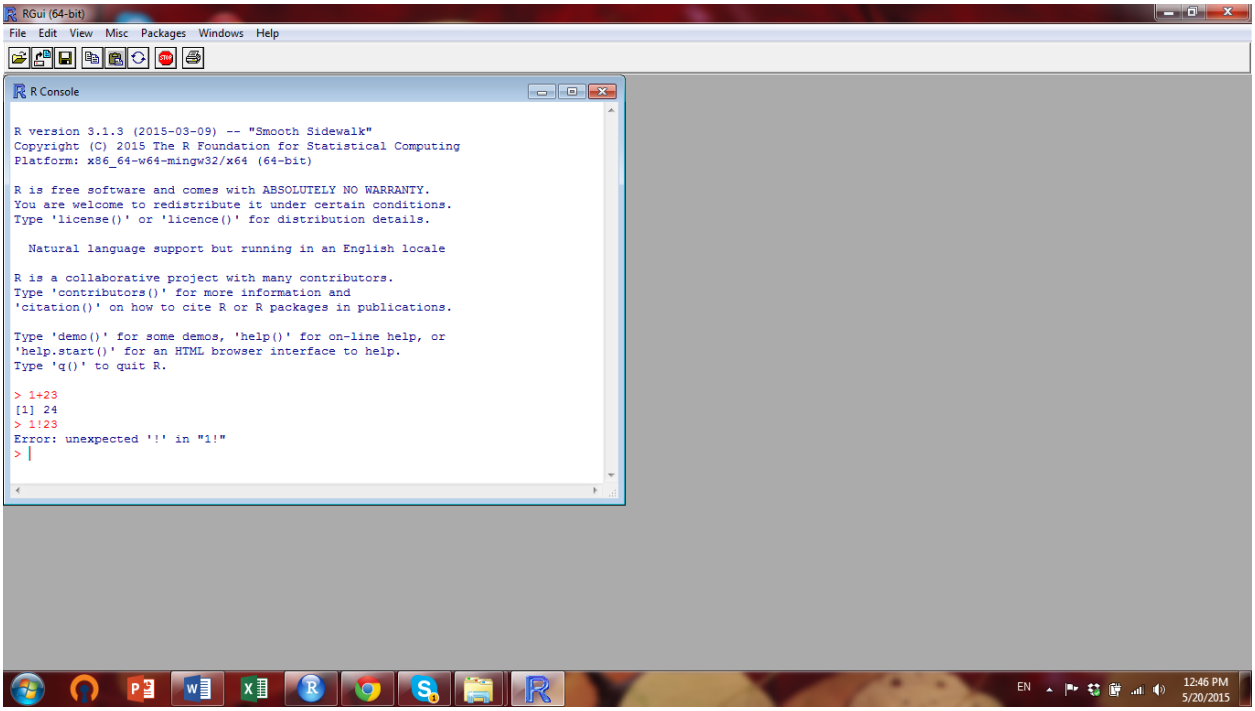
The Essentials of R

R is maintained and archived on CRAN

> CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.

User Interface in R: RStudio

Running R in command line can be burdensome!!!

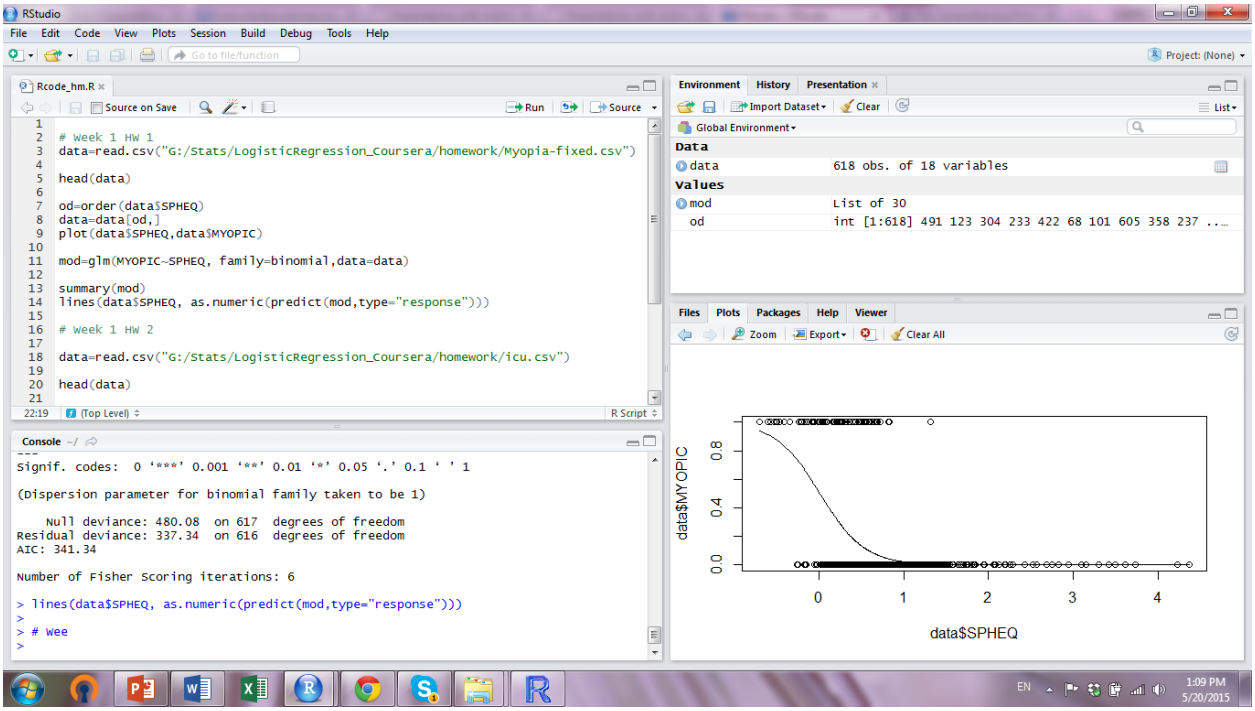


What is RStudio?

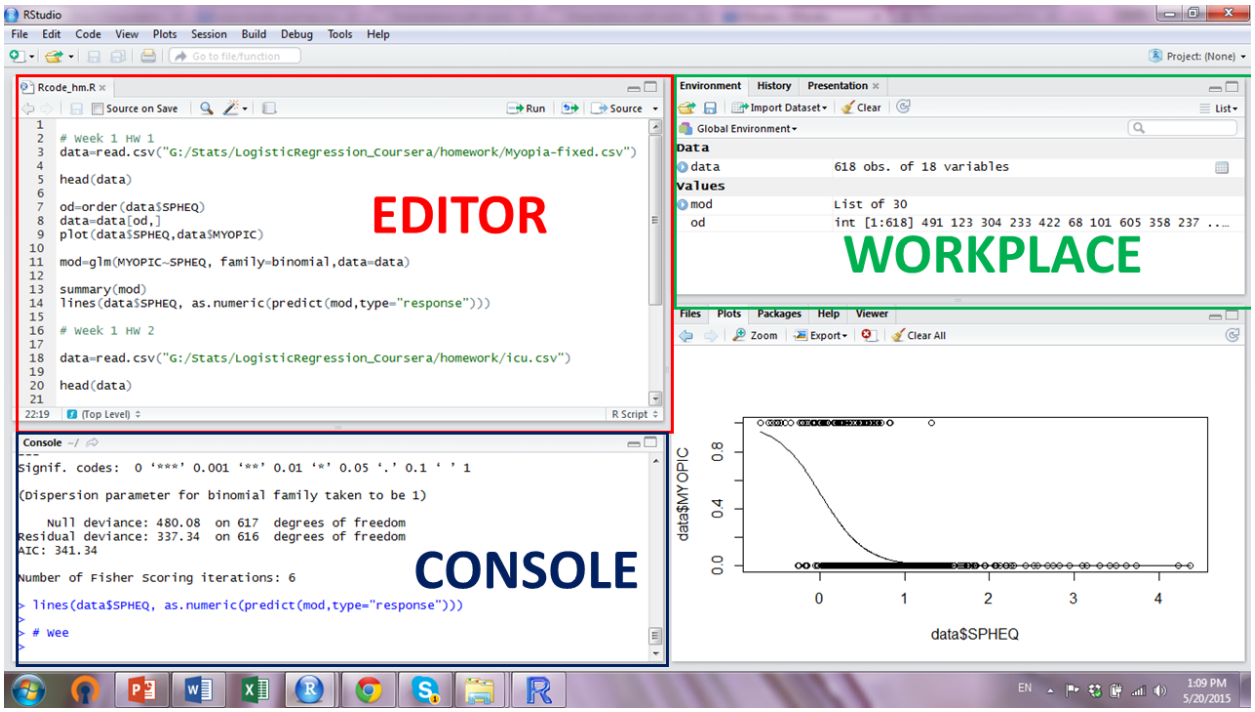
RStudio

> RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

RStudio



RStudio - Sections



RStudio - Sections

create **R scripts** in the editor: *File -> New File -> R Script*

Always comment your script properly!!!

```
# Intro to R
# January 25, 2016
# Basic R commands
```

```
# Write this addition into the editor and run it
# either by moving pointer to the line and clicking "Run" above
# or by highlighting the line and clicking "Run"
# Intead of "Run", you can also click "Ctrl + Enter"

2+5 # this addition will be executed and the result printed in the Console
```

```
[1] 7
```

```
10^6
```

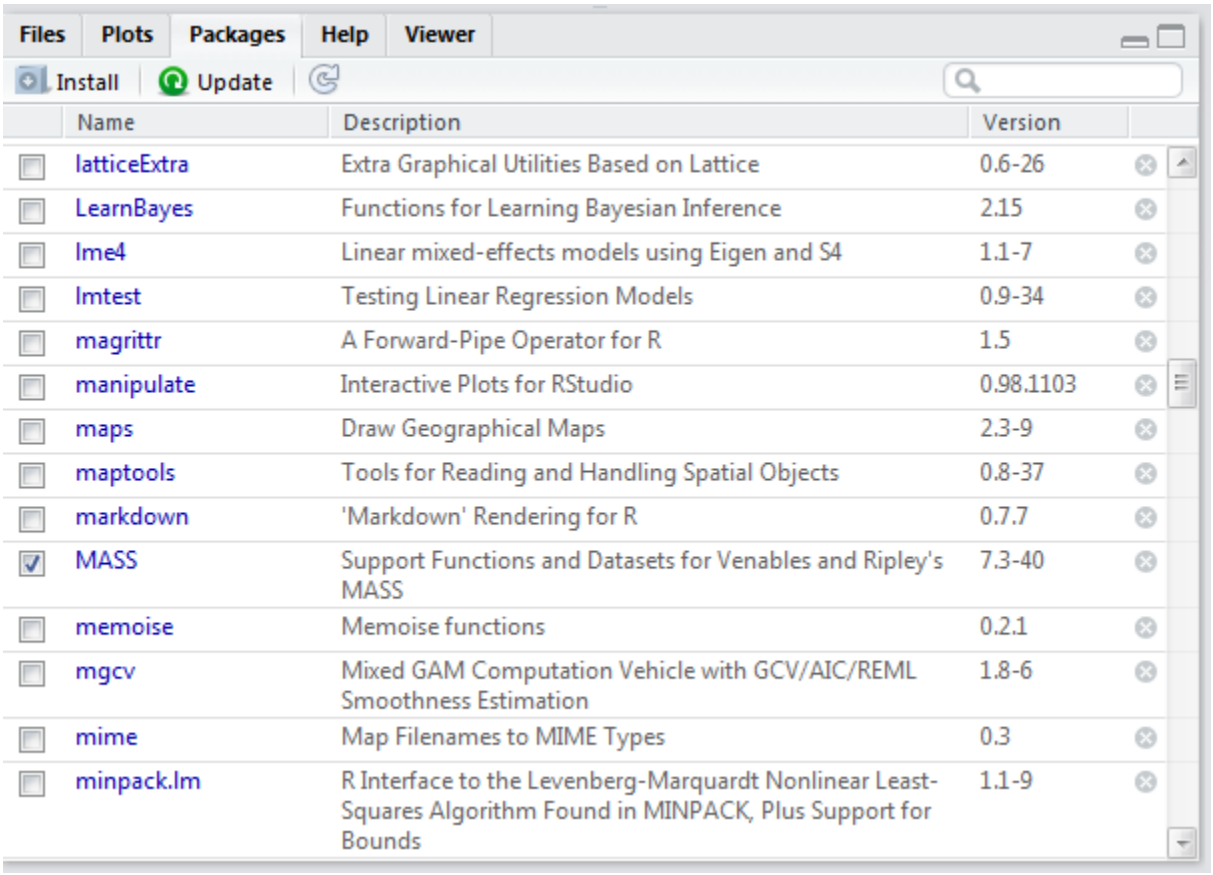
```
[1] 1e+06
```

Flexibility of R Studio

- Manipulate the scripts in the editor
- Save your R scripts and open them in R studio later

Packages in R

Many statistical operations are built-in in R. Others are provided as *packages*. Ca. 25 packages are by default included in R. You can take a look at all your packages by clicking on the *Packages* tab in the lower right corner in R Studio:



Packages are essentially specialized add-ons to perform tasks such as plotting or statistical inference.

You can check the window next to a package to *load* the package into R. Notice that the actual command that is executed to load a package appears in the *console*:

library("MASS", lib.loc="~/R/win-library/3.1")

Obviously, when you use one or several packages a lot, it makes much more sense to include the the command **library("package_name")** into your script.

Packages on CRAN

There are thousands of contributed packages on CRAN: https://cran.r-project.org/. Some packages are also on GitHub and other repositories Bioconductor, but we will focus on CRAN for now.

Using packages is what your project is all about. Once you find a package you would like to work with, the first thing to do is to download it. Again, there are two ways of doing it:

1. point and click in the *Packages* manu: *Packages -> Install -> Install from Repository (CRAN, CRANextra)* -> give the name of package
2. run **install.packages("name")** in the console

Most packages come with a *vignette* (user's guide) or at the very least with a help file that explains all the functions included in the package.

After you load a package into R, you can use this command to get some info on the package:

?name

The package info will appear in the *Help* menu right next to the *Packages* menu. An easier way to go about this is actually Googling the package.

EXERCISE: Install and load the package **ggplot2**. What does this package do?

Some useful resources and (humble) advice

There are a TON of resources online. One of the best overviews of R I have found so far is [Awesome R] (https://github.com/qinwf/awesome-R). Here you will find many packages organized by topic and also many useful links to get R help.

In general, Google is your best friend, BUT, search for R topics **in English**.

Learning R is a steep curve, but it is worth it if you require computational flexibility in your research.

Error messages and warnings

Before we begin the actual coding, be warned: **you will get error messages**, and will also get stuck not knowing how to perform certain functions or fix the errors. When this happens, follow theses steps:

- Google the error, including archives of forums that dealt with a similar problem
- Read the manual of a package (e.g., FAQ sections)
- Experiment with your code to try and find an answer (many errors are syntax erors - did you forget a comma? case-sensitivity?)
- ask someone who knows R
- if you are a programmer, take a look at the source code