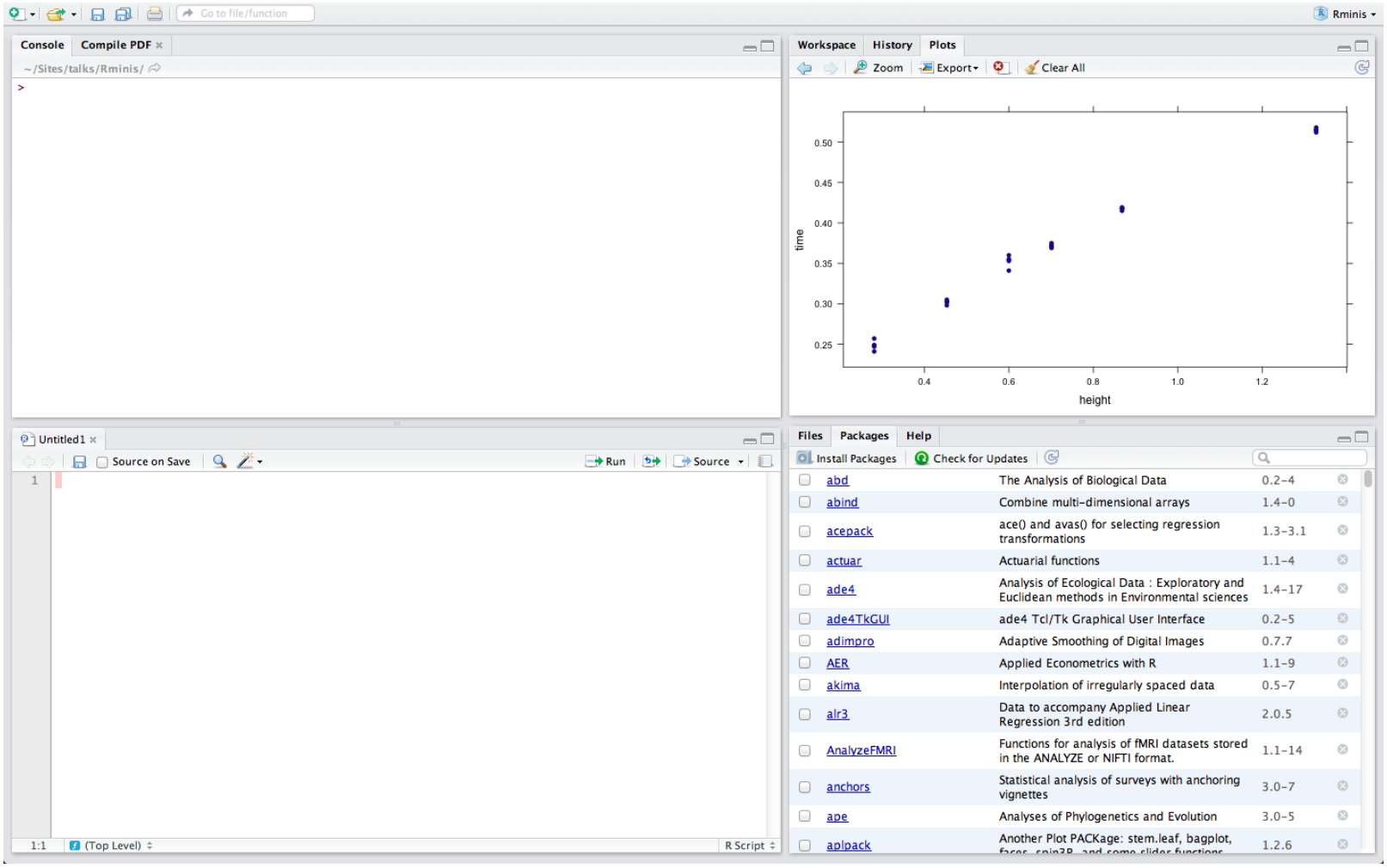
Getting Started with RStudio

Randall Pruim

Calvin College

Welcome to RStudio

RStudiois organized into four panes, some with multiple tabs.



What's Where?

RStudiotries to make the most of the real estate by dividing its window into 4 panels, each of which may have multiple structuretabs. Which tabs are in which panels is configurable.

Some of the important tabs include

- Console: This is where you can execute R commands interactively
- History: A record of past commands (can be saved, reloaded, etc.)
- Workspace: A listing of the objects available in your R session
- Plots: Where plots show up
- Help: Where documentation files appear when you ask for them
- Files: A file manager for locating, loading, moving, renaming, files.
- Packages: Install and load packages here.
- Open Files: Open files have a tab labeled with the file name.

The R Console can be used like a calculator

```
> 3 * 5

[1] 15

> sqrt(100)

[1] 10

> log(100)

[1] 4.605

> log10(100)

[1] 2
```

The default prompt is $>$ and a $+$ prompt is used to indicate that R is waiting for more input. This allows you to break commands over multiple lines.

Getting Around

You can speed up your typing and improve accuracy using these tricks:

- `TAB` completion: Hit tab while typing a command and RStudio will help you complete it.
- $\uparrow$, $\downarrow$: navigate through your history with arrow keys.
- `<ESC>`: If things get messed up, hit the escape key and try again.

Saving Results

You can save results using one of the assignment operators (=, <-, ->)

```
> x <- 3 * 5
> x


[1] 15


> x + 10


[1] 25


> wow <- sqrt(x + 10)
> wow


[1] 5
```

Notice that results that are assigned are not automatically displayed.

Arithmetic with Vectors

R makes it easy to work with more than one number at a time. R calls these ordered lists of numbers vectors.

```
> x <- 1:10
> x

 [1]  1  2  3  4  5  6  7  8  9 10

> x^2

 [1]   1   4   9  16  25  36  49  64  81 100

> log(x)

 [1] 0.0000 0.6931 1.0986 1.3863 1.6094 1.7918 1.9459 2.0794 2.1972
[10] 2.3026
```

Recycling

When doing arithmetic with multiple vectors, the shorter vector is recycled to make it as long as the longer one, then arithmetic happens component-wise.

```
> x <- 1:9
> x


[1] 1 2 3 4 5 6 7 8 9


> y <- 1:3
> y


[1] 1 2 3


> 100 * x + y  # notice recycling!


[1] 101 202 303 401 502 603 701 802 903
```

Data Frames: R's primary way to store data

A data frame is arranged in rows and columns.

- rows: observational units
- columns: variables

```
> dim(KidsFeet)  # how many rows, columns?


[1] 39  8


> head(KidsFeet, 3)  # first 3 rows


   name birthmonth birthyear length width sex biggerfoot domhand
1 David          5        88   24.4   8.4   B          L       R
2  Lars         10        87   25.4   8.8   B          L       L
3  Zach         12        87   24.5   9.7   B          R       R
```

```
> KidsFeet$sex  # one variable


 [1] B B B B B B B G G B B B B B G G G G G G B B G G G B G B B B G G G
[34] B B G G G G
Levels: B G
```

Adding new variables

The `transform()` function provides a simple way to add new variables to a data frame (often by transforming other variables).

```
> # these do the same thing
> KidsFeet$llength1 <- log(KidsFeet$length)
> KidsFeet <- transform(KidsFeet, llength2 = log(length))
```

```
> KidsFeet <- transform(KidsFeet, v = width * length)  # crude area estimate
> head(KidsFeet, 3)

   name birthmonth birthyear length width sex biggerfoot domhand
1 David          5        88   24.4   8.4   B          L       R
2  Lars         10        87   25.4   8.8   B          L       L
3  Zach         12        87   24.5   9.7   B          R       R
  llength1 llength2     v
1    3.195    3.195 205.0
2    3.235    3.235 223.5
3    3.199    3.199 237.6
```

Grabbing a subset

The `subset()` function culls a subset out of a data frame.

```
> data(KidsFeet)  # reload the original data set
> Girls <- subset(KidsFeet, sex == "G")
> dim(Girls)


[1] 19  8


> Girls$sex


 [1] G G G G G G G G G G G G G G G G G G G
Levels: B G
```

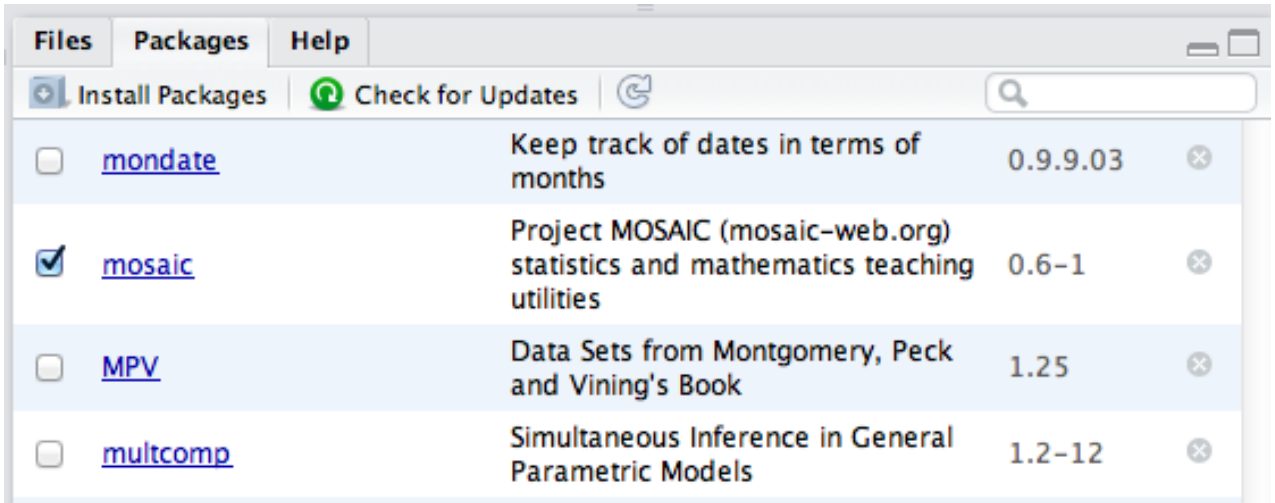
Notes:

- == is used for comparing to see if things are equal (to distinguish it from assignment)
- We need quotation marks around `G`, else R will look for a variable named `G`.
- R remembers that `B` is a possible value for `sex`.

Packages Tab

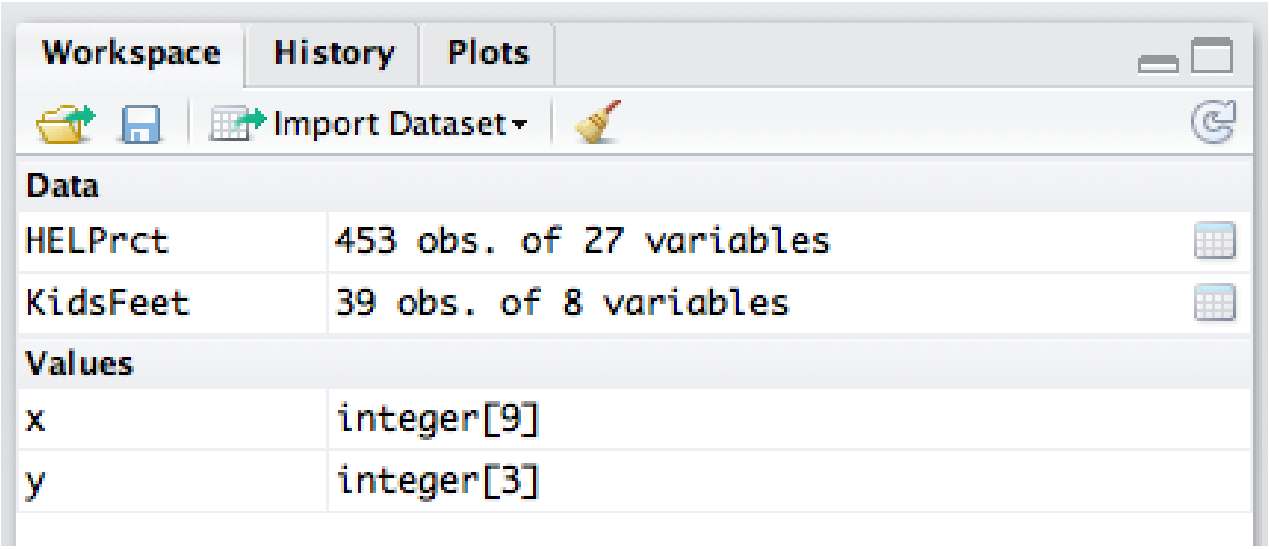
Much of the power of R comes from the 1000s of R packages containing code and data for specialized situations. You can inspect the installed packages in the `Packages` tab.

- Check marks indicate that the package is loaded (i.e., usable).
- Click on the install packages icon to search for packages and install them.



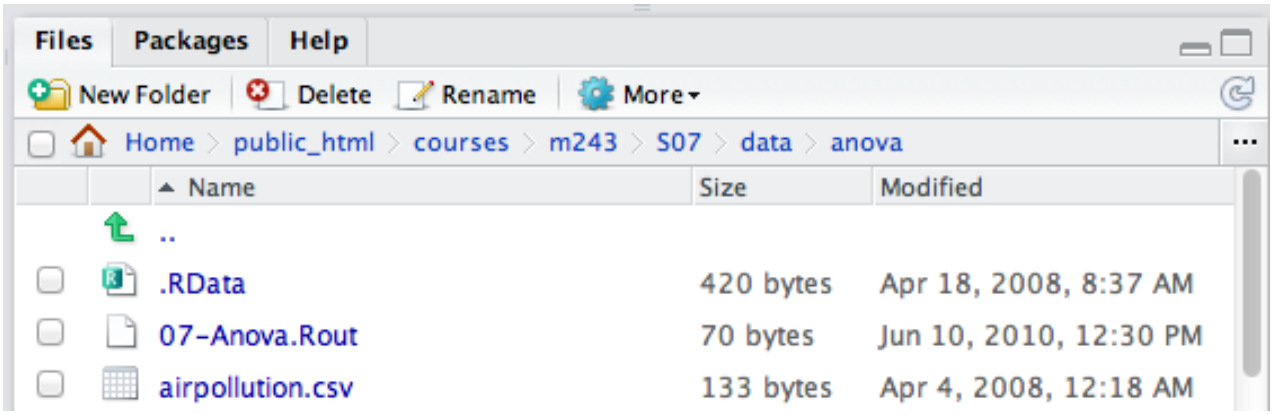
Workspace Tab

In addition to a listing of objects in your workspace, this is where you find tools to help you import data. (If you are working in the web version, you will first want to upload the data in the `Files` tab to get the data onto the server.)



Files Tab

The `Files` tab provides a simple interface for finding, opening, moving, renaming files. If you are working on the server, these are the files in your account on the server, not those on your local machine. (But you can upload files to get them from your machine to the server.)



File Tabs

Eventually you will want to graduate from working in the console to working in a file, since this makes it easier to edit an save code.

In the file menu, you can create new files of various types (R Script, R markdown, Sweave, etc.)

- R Script: R Code
- R Markdown: easy way to create documents with text, code, analysis, plots all in one document. File gets converted to markdown and then to HTML. (`pandoc` can be used to convert to other file types.)
- R Sweave: Similar to R Markdown, but working with LaTeX-based `knitr`. For those who know LaTeX this gives much more control of format, but it isn't as easy to use.

RStudio includes integrated tools for compiling these files, executing individual lines and chunks of R code, debugging, etc.