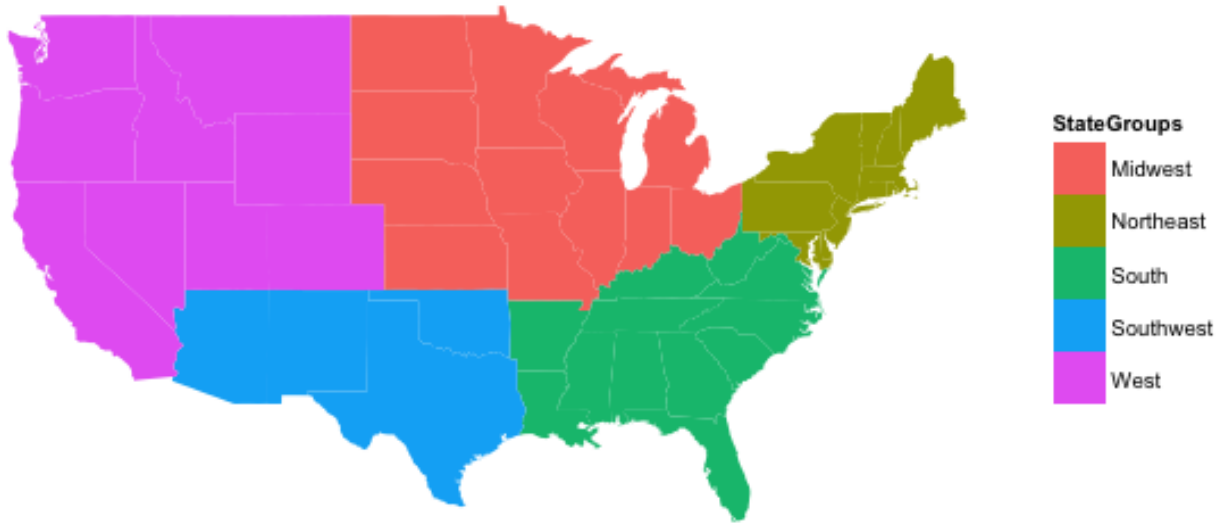
Working with R to Analyze and Plot Data

Di Cook, Eric Hare

May 14, 2015



California Dreaming - ASA Travelling Workshop

INTRODUCTION TO R

This material is taken from Garret Grolemund's book "Hands-On Programming with R", and it serves as a quick start on learning to use the software of data analysis. We would encourage you to get a copy of this book if you want to get started on R independently.

The versions of the software that we will use are:

- RStudio 0.98.1103
- R 3.2.0 "Full of Ingredients"

California Dreaming - ASA Travelling Workshop

WHY R?

These three reasons make it an essential tool for analyzing data. R is a programming language. The language was initially research in the 1970s, and it has evolved to be the best (some may argue) language of data analysis. Data analysis needs to be performed on the computer, but you have a choice of using a graphical user interface or a programming language. Don't be afaid of programming! It will equip you to be a data scientist, which is very handy when you come to apply for jobs today. Using the R language:

- encourages reproducibility because it allows you and others to re-create a past analysis, essential for good data science.

- enables automation, so that an analysis can be re-run easily if the data changes
- enables communication because the code is just text, so can be emailed to others to read, can be googled for tracking down errors and help.

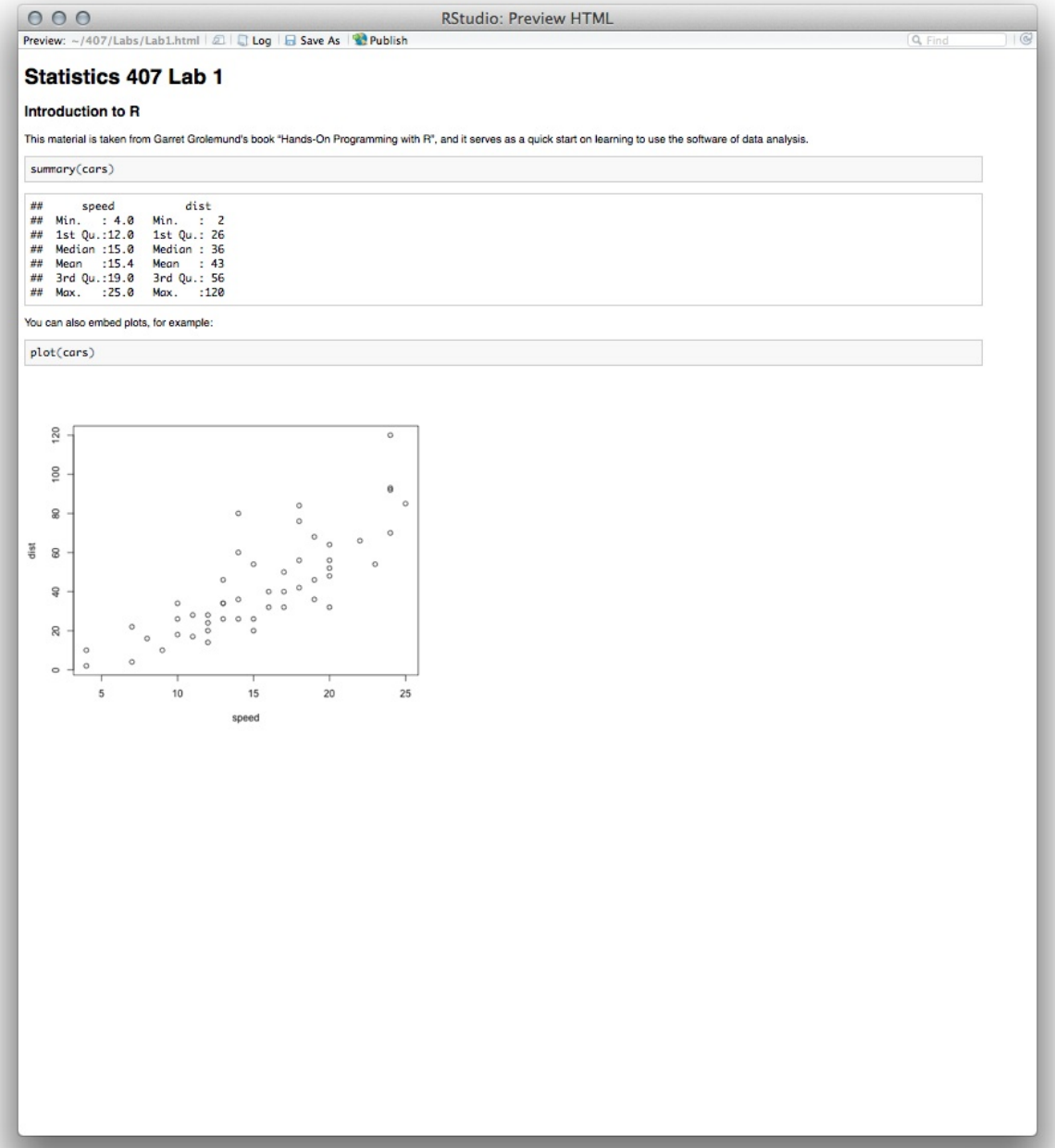
Learning a new language can be frustrating. It will take time to be fluent. This is normal. It is worth it!

CREATE A PROJECT

The top left pane has R code embedded in a markdown document, so that you can write code, and write explanations, simultaneously. It is a good way to keep your analysis organized: write code, document the code and the results, and output it to a communicable format.

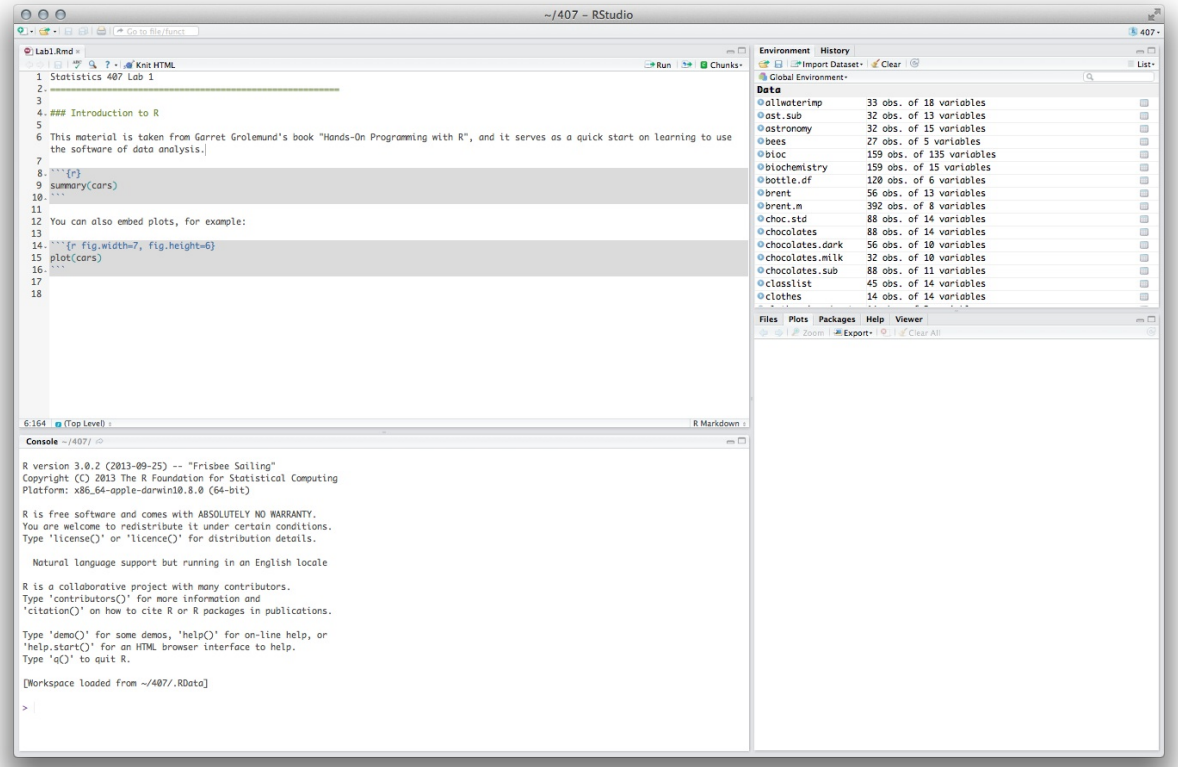
Using the "Knit HTML" button we can process the document, get the R code to run and produce an html page, that looks like this

From the html, you can convert to a pdf document or to a Microsoft Word document.



Using the RStudio Interface

Lines of code in the text editor window can be run one-by-one by setting the cursor to point to the line, and clicking the "Run" button. Blocks of code can be run by highlighting the lines and clicking "Run". Once you are sure that the code runs properly, you can embed it in the markdown and use "Knit HTML" to generate your document. My recommendation is to work this way, incrementally, in small chunks and build up to a full final document.



Your turn

- Download the "try.Rmd" file,
- Open in the R Studio,
- Knit as html, and knit as Word

For the advanced user, you can also write in latex (instead of markdown) and

R as a calculator

```
1+1
2*5
12/4
20-3
100:110 # All integers from 100 to 110
```

```
## [1] 2
## [1] 10
## [1] 3
## [1] 17
##  [1] 100 101 102 103 104 105 106 107 108 109 110
```

If you type a command that R doesn't recognize you will get an error, eg.

```
> 5 % 3
```

```
Error: unexpected input in "5 % 3"
```

Objects

R objects store data. The assignment operator, `<-` is used to name the object. It is also possible to use `=` to mean the same thing.

```
mynumbers <- 5:12
mynumbers
mynumbers + 2
```

```
## [1]  5  6  7  8  9 10 11 12
## [1]  7  8  9 10 11 12 13 14
```

Names can be almost anything, except for special characters `^, !, $, @, +, -, /, *`. It is good practice to name your objects with some meaning for what they contain, be reasonably short (less typiing). They should from common R functions, for example, don't use `data` because it is also used to load stored data from packages, or `c` because this is an R function that allows you to collect a bunch of objects together. You won't get errors by using these names but you may get confused when you come back and look at your code later.

Types of objects

Objects can be of different types. The object `mynumbers` is a vector of numbers. Numbers can be various types also: integer or double.

```
typeof(mynumbers)
is.numeric(mynumbers)
is.vector(mynumbers)
length(mynumbers)
```

```
## [1] "integer"
## [1] TRUE
## [1] TRUE
## [1] 8
```

Types of objects

Other common types of objects for data analysis are characters, logicals, factors, dates. Factors store categorical data. Dates have a special format that enables it to be treated similarly to how we use dates in real life.

```
mytext <- c("hello", "class")
length(mytext)
mylogic <- c(TRUE, FALSE, TRUE, TRUE)
gender <- factor(c("male", "female", "female", "female", "male"))
levels(gender)
summary(gender)
```

```
## [1] 2
## [1] "female" "male"
## female   male
##      3      2
```

Working with temporal data

There are basic routines for handling dates specially, but the best package for operating on temporal data is lubridate, we'll talking about this later.

```
now <- Sys.time()
now
typeof(now)
class(now)
today <- as.Date("08/26/2014", format="%m/%d/%Y")
today
```

```
## [1] "2015-05-14 16:22:55 CDT"
## [1] "double"
## [1] "POSIXct" "POSIXt"
## [1] "2014-08-26"
```

Functions

One of the powerful aspects of R is to build on the reproducibility. If you are going to do the same analysis over and over again, compile these operations into a function that you can then apply to different data sets.

```
mymode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
mymode(c(rep("A", 5), rep("B", 3)))
```

```
## [1] "A"
```

Function structure

`mymode <- function(x, labelit="m") ....`

- `x` is an input to the function that is essential to provide each time.
- `label` is an optional addition that mostly will be the default value, but the user could change it to something else, simply by changing it in the function on the fly.

```
mymode(x=c(rep("A", 5), rep("B", 3)))
mymode(x=c(rep("A", 5), rep("B", 3)), labelit="mode")
```

```
##   m
## "A"
## mode
##  "A"
```

Next

Let's make some plots of data....