ETC 2420/5242 Lab 1 2016

Di Cook

Week 1

Getting up and running with the computer

- R and RStudio
- Projects
- RMarkdown
- Basic functions

R is ...

- **Free** to use
- **Extensible**
  - Over 8500 user contributed add-on packages currently on CRAN!
- **Powerful**
  - With the right tools, get more work done, faster.
- **Flexible**
  - Not a question of *can*, but *how*.
- **Frustrating**
  - Flexibility comes at a cost (easy to shoot yourself in the foot!).

RStudio is ...

From Julie Lowndes:

If R were an airplane, RStudio would be the airport, providing many, many supporting services that make it easier for you, the pilot, to take off and go to awesome places. Sure, you can fly an airplane without an airport, but having those runways and supporting infrastructure is a game-changer.

The RStudio IDE

1. Source editor:

- Docking station for multiple files,
- Useful shortcuts ("Knit"),
- Highlighting/Tab-completion,
- Code-checking (R, HTML, JS),
- Debugging features

2. Console window:

- Highlighting/Tab-completion,

- Search recent commands

3. Other tabs/panes:

- Graphics,
- R documentation,
- Environment pane,
- File system navigation/access,
- Tools for package development, git, etc

Projects

Creating a project helps organise work. For this unit, I have created a project on my laptop called `ETC2420`.

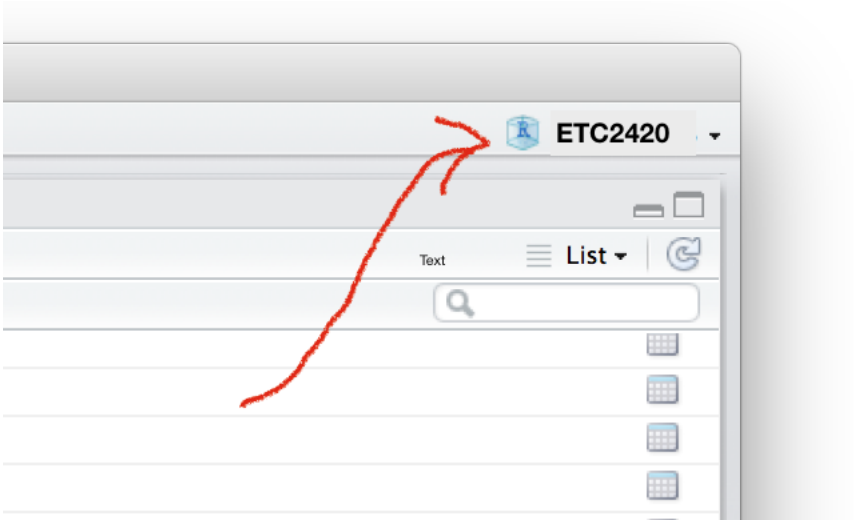

Figure 1: Using projects to organise your work

Exercise 1

Create a project for this unit, in the directory. (Be sure that you open this at the start of each lab. Generally it is a good idea NOT TO SAVE THE WORKSPACE when you close a project for the day.)

- File -> New Project -> Existing Directory -> Empty Project

Exercise 2

Open a new Rmarkdown document. You are going to want to call it `MYLab1` (it will automatically get the file extension `.Rmd`) when you save it.

- File -> New File -> R Markdown -> OK -> Knit HTML

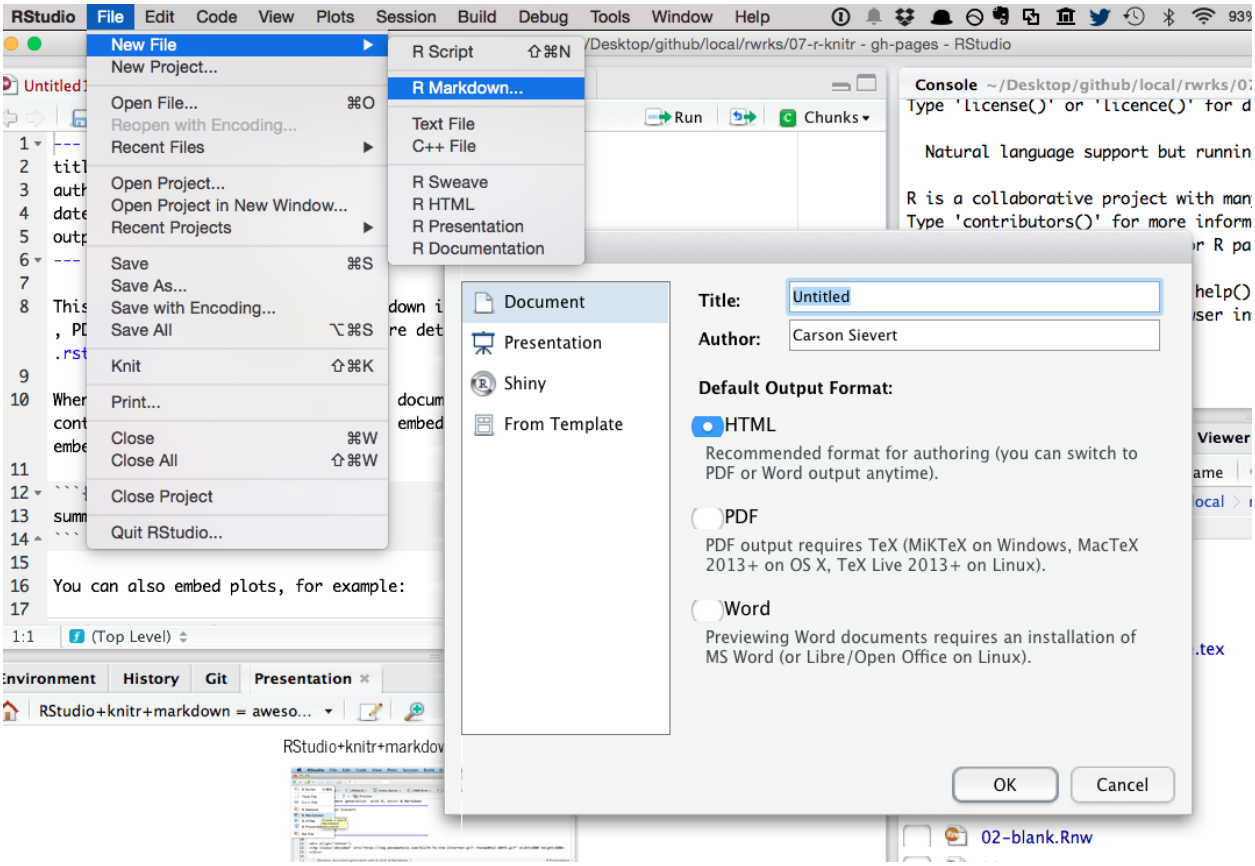

Figure 2: Writing and computing with the one document

What is R Markdown?

- From the R Markdown home page:
- R Markdown is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R.
- It combines the core syntax of **markdown** (an easy-to-write plain text format) **with embedded R code chunks** that are run so their output can be included in the final document.
- R Markdown documents are fully reproducible (they can be automatically regenerated whenever underlying R code or data changes).

Exercise 3

Look at the text in the `MYLab1.Rmd` document.

- What is R code?
- How does `knitr` know that this is code to be run?
- Using the RStudio IDE, work out how to run a chunk of code. Run this chunk, and then run the next chunk.
- Using the RStudio IDE, how do you run just one line of R code?
- Using the RStudio IDE, how do you highlight and run multiple lines of code?
- What happens if you try to run a line that starts with "```{r}"? Or try to run a line of regular text from the document?
- Using the RStudio IDE, `knit` the document into a Word document.

Getting data

Data can be found in R packages

```
library(dplyr)
data(economics, package = "ggplot2")
# data frames are essentially a list of vectors
glimpse(economics)
# Observations: 574
# Variables: 6
# $ date     <date> 1967-07-01, 1967-08-01, 1967-09-01, 1967-10-01, 1967...
# $ pce      <dbl> 507.4, 510.5, 516.3, 512.9, 518.1, 525.8, 531.5, 534....
# $ pop      <int> 198712, 198911, 199113, 199311, 199498, 199657, 19980...
# $ psavert  <dbl> 12.5, 12.5, 11.7, 12.5, 12.5, 12.1, 11.7, 12.2, 11.6,...
# $ uempmed  <dbl> 4.5, 4.7, 4.6, 4.9, 4.7, 4.8, 5.1, 4.5, 4.1, 4.6, 4.4...
# $ unemploy <int> 2944, 2945, 2958, 3143, 3066, 3018, 2878, 3001, 2877,...
```

These are not usually kept up to date but are good for practicing your analysis skills on.

Or in their own packages

```
library(gapminder)
glimpse(gapminder)
# Observations: 1,704
# Variables: 6
# $ country   <fctr> Afghanistan, Afghanistan, Afghanistan, Afghanistan,...
```

```
# $ continent <fctr> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asi...
# $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992...
# $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.8...
# $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 1488...
# $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 78...
```

I primarily use the `readr` package for reading data now. It mimics the base R reading functions but is implemented in `C` so reads large files quickly, and it also attempts to identify the types of variables.

```
library(readr)
ped <- read_csv("http://dicook.github.io/Statistical_Thinking/data/Pedestrian_Counts.csv")
glimpse(ped)
# Observations: 1,392,618
# Variables: 4
# $ Date_Time     <chr> "01-MAY-2009 00:00", "01-MAY-2009 00:00", "01-MA...
# $ Sensor_ID     <int> 4, 17, 18, 16, 2, 1, 13, 15, 9, 10, 12, 11, 5, 6...
# $ Sensor_Name   <chr> "Town Hall (West)", "Collins Place (South)", "Co...
# $ Hourly_Counts <int> 209, 28, 36, 22, 52, 53, 17, 124, 5, 8, 2, 5, 15...
```

You can pull data together yourself, or look at data compiled by someone else.

Question 1

- Look at the `economics` data in the `ggplot2` package. Can you think of two questions you could answer using these variables?
- Write these into your `.Rmd` file.

Question 2

- Read the documentation for `gapminder` data. Can you think of two questions you could answer using these variables?
- Write these into your `.Rmd` file.

Question 3

- Read the documentation for `pedestrian sensor` data. Can you think of two questions you could answer using these variables?
- Write these into your `.Rmd` file.

Exercise 4

- What is a `package`?
- How does the `library()` function relate to a `package`?
- How often do you install a package, using `install.package()`?
- How often do to load a `library`?

Some R Basics

- *Assign* values to a name with `<-` is called *gets*
- `n_max=50` option to the `read_csv` function reads just the first 50 lines
- `dim` reports the dimensions of the data matrix
- `colnames` shows the column names (you can see these by looking at the object in the RStudio environment window, too)
- `$` specify the column to use
- `typeof` indicates the information format in the column, what R thinks
- complex variable names containing spaces, etc, can be used, as long as they are wrapped in single quotes

Data Types

- `list`'s are heterogeneous (elements can have different types)
- `data.frame`'s are heterogeneous but elements have same length
- `vector`'s and `matrix`'s are homogeneous (elements have the same type), which would be why `c(1, "2")` ends up being a character string.
- `function`'s can be written to save repeating code again and again
- If you'd like to know more, see Hadley Wickham's online chapters on data structures and subsetting

Operations

- Use built-in *vectorized* functions to avoid loops

```
set.seed(1000)
x <- rnorm(6)
x
# [1] -0.44577826 -1.20585657  0.04112631  0.63938841 -0.78655436 -0.38548930
sum(x + 10)
# [1] 57.85684
```

- `R` has rich support for documentation, see `?sum`

- Use `[` to extract elements of a vector.

```
x[1]
# [1] -0.4457783
x[c(T, F, T, T, F, F)]
# [1] -0.44577826  0.04112631  0.63938841
```

- Extract *named* elements with `$`, `[[`, and/or `[`

```r
x <- list(
  a = 10,
  b = c(1, "2")
)
x$a
# [1] 10
x[["a"]]
# [1] 10
x["a"]
# $a
# [1] 10
```

Examining 'structure'

- `str()` is a very useful `R` function. It shows you the "structure" of (almost) *any* R object (and *everything* in R is an object!!!)

```r
str(x)
# List of 2
#  $ a: num 10
#  $ b: chr [1:2] "1" "2"
```

Missing Values

- `NA` is the indicator of a missing value in R
- Most functions have options for handling missings

```r
x <- c(50, 12, NA, 20)
mean(x)
# [1] NA
mean(x, na.rm=TRUE)
# [1] 27.33333
```

Counting Categories

- the `table` function can be used to tabulate numbers

```r
head(table(ped$Sensor_Name), 5)
#
#                Alfred Place        Australia on Collins
#                       12365                       48310
#              Birrarung Marr  Bourke St-Russel St (West)
#                       44904                       18573
# Bourke St-Russell St (West)
#                        2208
```

Some Oddities

- Yes, `+` is a function (which calls compiled C code)

```
`+`
# function (e1, e2)  .Primitive("+")
```

- What's that? You don't like addition? Me neither!

```
"+" <- function(x, y) "I forgot how to add"
1 + 2
# [1] "I forgot how to add"
```

- But seriously, don't "overload operators" unless you know what you're doing

```
rm("+")
```

Getting Help on the Web

- Reading documentation only gets you so far. What about *finding* function(s) and/or package(s) to help solve a problem???
- Google! (I usually prefix "CRAN" to my search; others might suggest http://www.rseek.org/
- Ask your question on a relevant StackExchange outlet such as http://stackoverflow.com/ or http://stats.stackexchange.com/
- It's becoming more and more popular to bundle "vignettes" with a package (**dplyr** has *awesome* vignettes)

```
browseVignettes("dplyr")
```

Question 4

1. Read in the OECD PISA data
2. Tabulate the countries (CNT)
3. Extract the values for Australia (AUS) and Shanghai (QCN)
4. Compute the average and standard deviation of the reading scores (PV1READ), for each country
5. Write a few sentences explaining what you learn about reading scores in these two countries.

TURN IN

- Your `.Rmd` file
- Your Word (or pdf) file that results from knitting the Rmd.
- Make sure your group members are listed as authors, one person per group will turn in the report
- DUE: Monday after the lab, by noon, loaded into moodle

Resources

- RStudio IDE cheat sheet
- rmarkdown cheat sheet
- Q/A site: http://stackoverflow.com
- Dynamic Documents with R and knitr, Yihui Xie,