Getting started with R/R Studio

Cody Chiuzan, PhD

Department of Biostatistics
Columbia University
June 6, 2019

Outline

- Overview of R and R Studio
- Importing Data
  - Read CSV files
  - Load existing R datasets
- Examining Data Attributes
  - Data structure, type and dimensionality
  - Data subsets
- Manipulating Data
  - Rename, transform, and combine datasets
- R Markdown – reproducibility

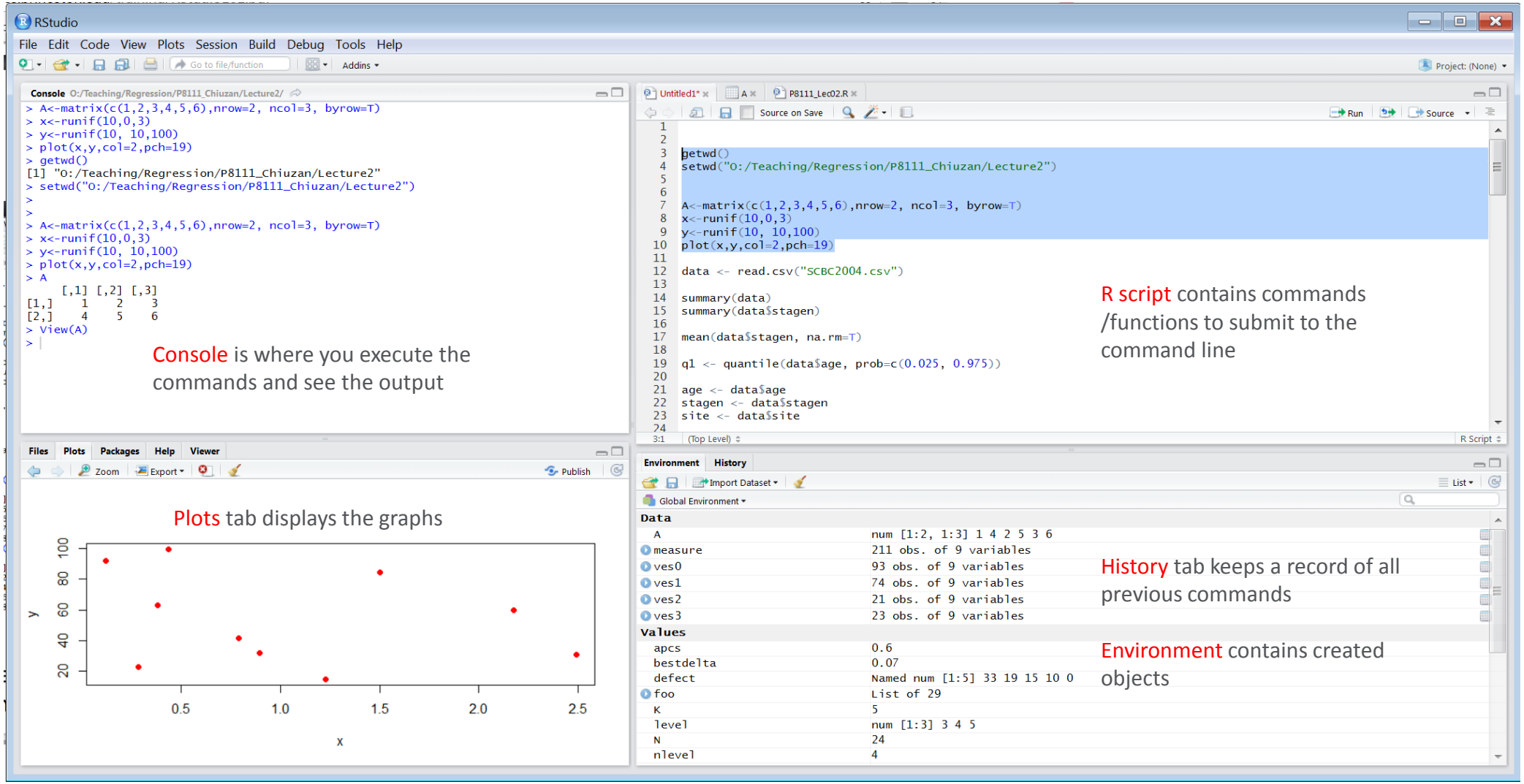
Why R?

- R is a FREE, open-source software used for statistical computing and graphics
- Can be frustrating: a steeper learning curve
  - R is more interactive and also more time consuming
  - There is no 'set-up' procedure for each type of analysis
  - But, you are in control!!
- Installing R: http://cran.r-project.org/ (for Windows, Mac, Linux)
- Some online resources:
  - https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf
  - http://www.mayin.org/ajayshah/KB/R/index.html
  - *R.D., Peng Exploratory Data Analysis With R: https://leanpub.com/exdata*

What is RStudio?

- User-friendly development environment for R
- Installing RStudio: http://www.rstudio.com/
- Some online resources:
  - http://dss.princeton.edu/training/
  - http://libguides.princeton.edu/dss

RStudio: Windows



R Workflow

- For every new project, do the following:
  - Create a directory with a (suggestive) name and path
  - Put an R Project in the directory
  - Create an R Project using File -> New Project -> New or Existing Directory and specifying the directory you just created
- Keep everything – datasets, scripts, reports, output – in there!

Let’s try it!

- Create an R Project:

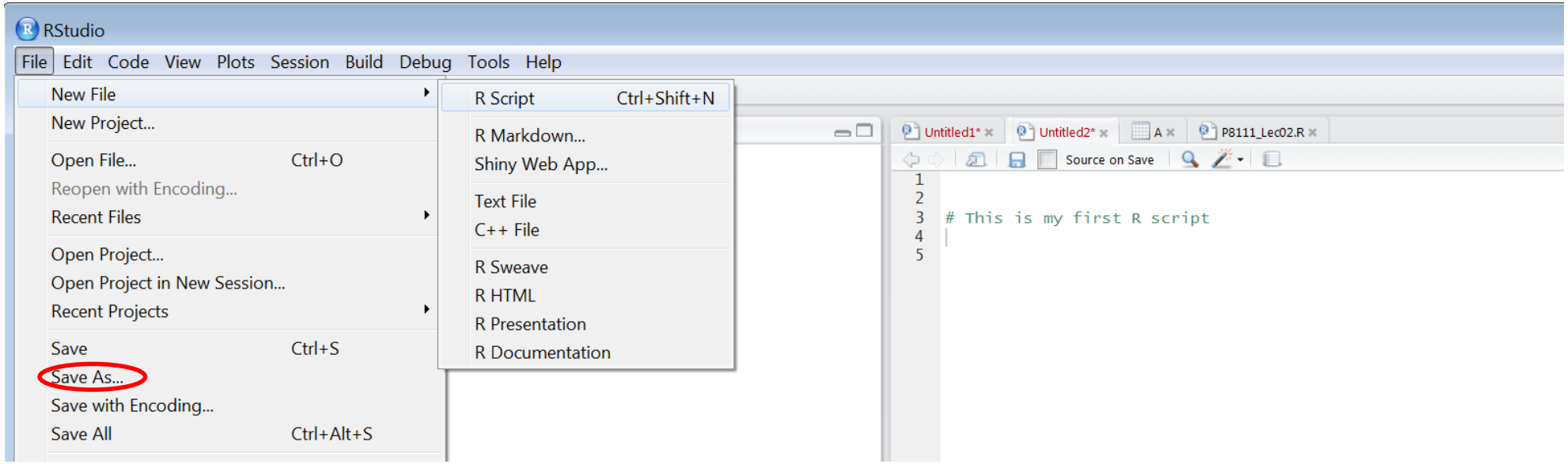
  File -> New Project -> ML_RIntro (for example)

  (specify the directory you just created)

- Move the three datasets and the R code provided for this assignment to that directory

R Script

- How to create and save an R script



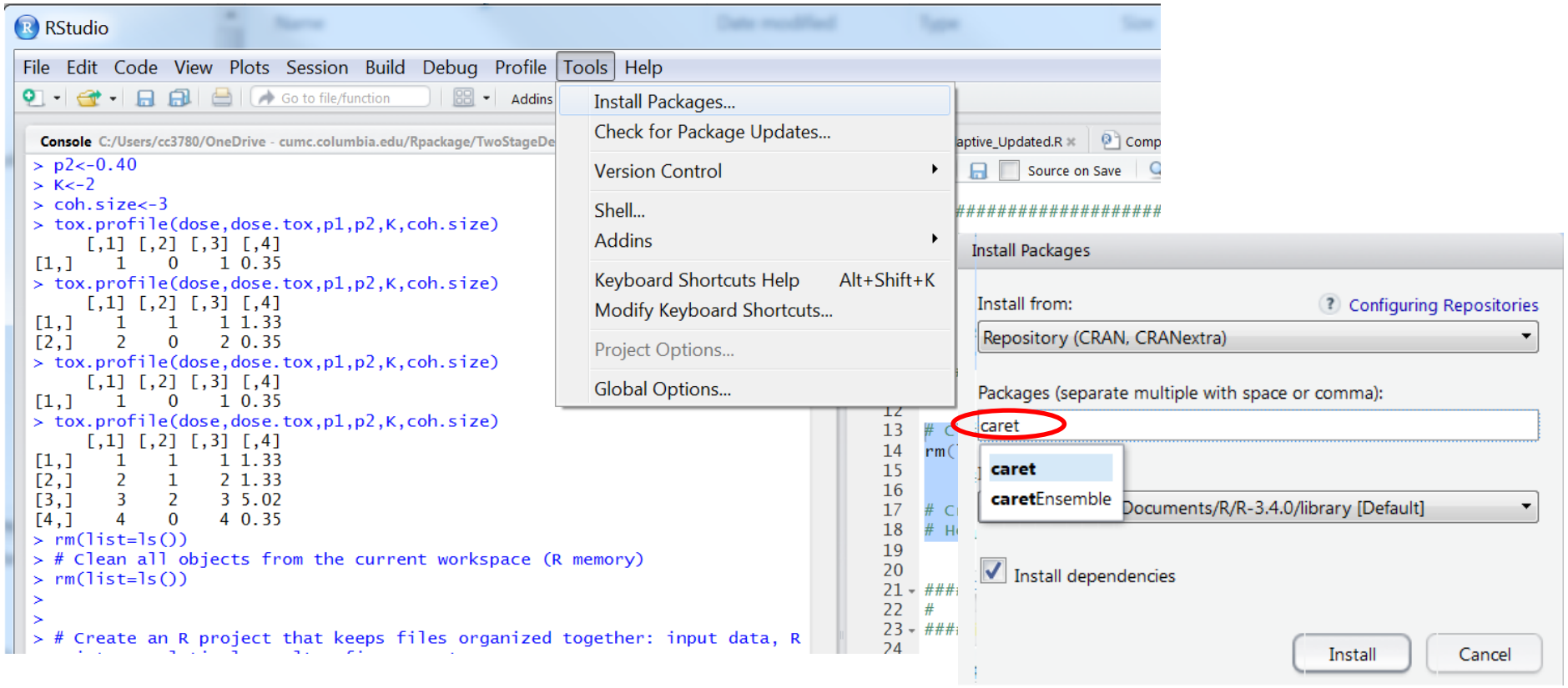
- Type R commands and run them
  - Windows: put the cursor on the line and press CTRL+R (Windows) or Command + Enter (Mac)

R Packages

- Packages are collections of **R** functions, data, and compiled code in a well-defined format
- The directory where packages are stored is called the library
- **R** comes with a standard set of packages
  - Only need to install the package **one time**
  - Others are available for download and installation
- Once installed, they have to be loaded into the session to be used
  - Once installed, you need to load package every time you use R!

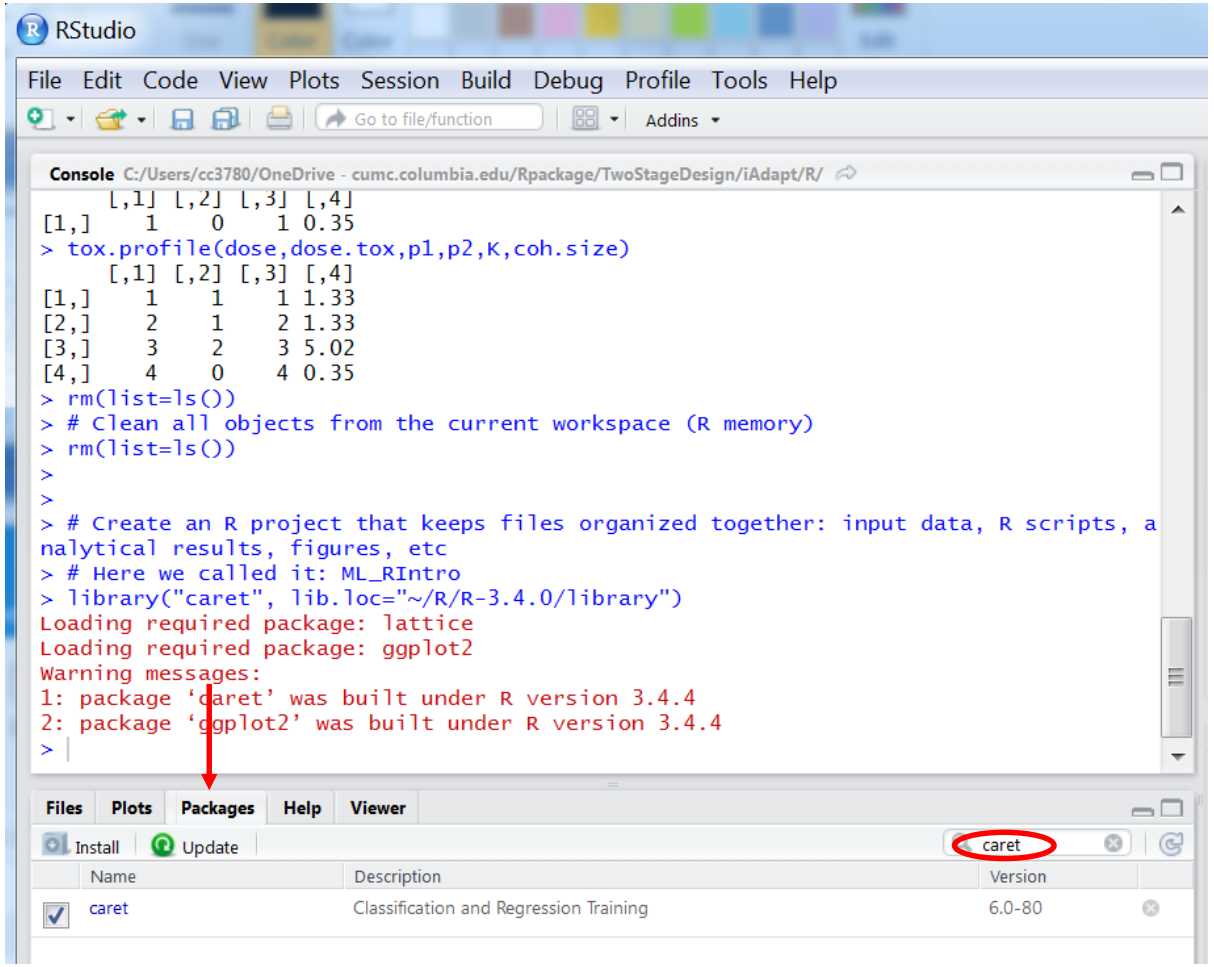
R Packages

- How to install new packages (contain functions)



R Packages

- Use tab ‘Packages’ to:
  - Activate a package
  - Install a new package
  - Update a package
- Or type *library(name)* in the console



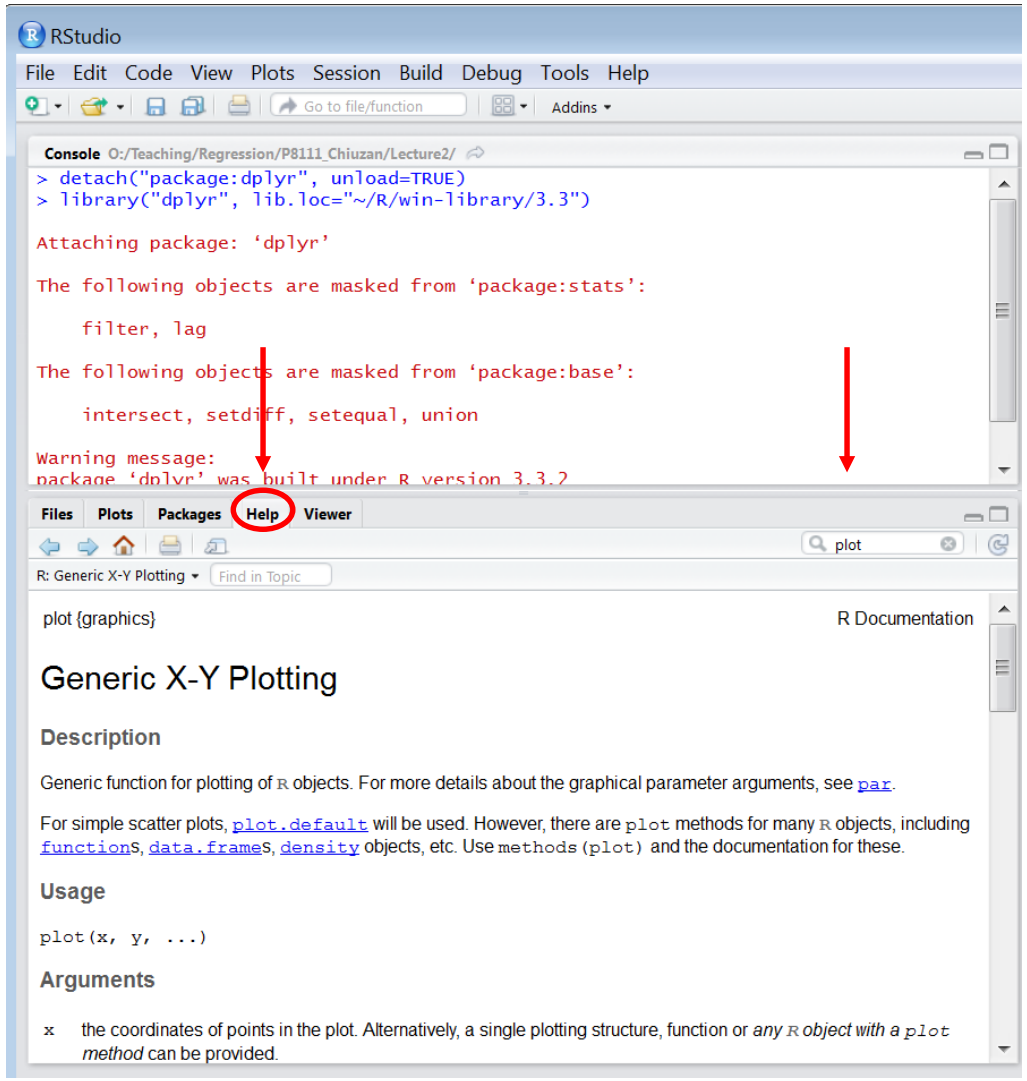
R Help

- Use tab ‘Help’ to look for a topic

  e.g., look for function ‘plot’

- Tab ‘Help’ has a history of the most recent topics you inquired about

- Or just type help(plot) in the console



R Syntax

- R is an object-oriented programming language
  - Users define not only the type of the data structure, but also the type of functions to be applied to the data
- R is *case sensitive*: ‘A’ and ‘a’ are different symbols
- Commands are separated either by (;) or by a new line
  - Commands can be grouped together (in functions) by { and }
- Comments can be inserted almost anywhere
  - Starting with a ‘#’, everything to the end of the line is a comment
  - Use comments to document your code: for YOU and OTHERS!!

Importing Data

Reading Data into R

- First, you need to save your data onto your computer
  - Excel, SPSS, or some other type of file
  - Datasets need to be in the "*PROJECT*" folder we created earlier
- Some useful tips:
  - Reserve the first row for headers (variable/column names)
  - First column is used to identify sampling units
  - Avoid named or fields with blank spaces; put a '_' instead.
  - Delete comments from Excel
  - Missing values should be noted with ' . ' or 'NA'
  - Avoid symbols such as: ' #, ?, *, <, /, -, }'

Reading Data into R

- Set your working directory (e.g., folder where you stored your data):
  - R function: setwd("location of your dataset")
- If you want to know what your working directory is:
  - R function: getwd()
- Careful how you declare the path:
  - Mac ≠ Windows (see R code)!
- Another useful command to remove all objects from the current workspace (R memory):
  - R function: rm(list=ls())

Reading Data into R

- Read CSV files – if you have a '.csv' file (comma separated file)
  - read.csv("location of your dataset/Data.csv", header = FALSE)
  - The header is also set to 'TRUE' by default

- Export data into a CSV file
  - write.csv(mydata, file = "location of your dataset/Data.csv", na= " ")

if you want to omit the NA values

- Read TXT files – if you have a '.txt' or a tab-delimited text file – not so common anymore
  - read.table("location of your dataset/Data.txt", header = FALSE)

Loading Data from R packages

- There are more than 100 datasets supplied with R (in package ‘datasets’)
- To see the list of all datasets currently available use
    - R: data()
- Loading data ‘esoph’ from the R package ‘datasets’
    - R: data(PimaIndiansDiabetes, package = “mlbench")
- If a package has been already been loaded by library(), its datasets are automatically included in the search.
    - Some packages are loaded automatically (e.g., ‘base’)

Data Attributes

Data Description

- Before running any analysis, make sure you examine your data
  - Number of variables and their types, number of observations, missing data, etc.
- Obtain variable names
  - R: names(mydata)
- Obtain data dimensions: (#rows) by (# columns)
  - R: dim(mydata)
  - R: nrow(mydata); R: ncol(mydata)
- Look at the 'top' and 'bottom' of the data
  - R: head(mydata), tail(mydata)
- Check for missing data
  - R: anyNA(mydata)

Data Description: Examples

- Examine the classes of each column:

```
> str(PimaIndiansDiabetes)
'data.frame':   768 obs. of  9 variables:
 $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
 $ pressure: num  72 66 64 66 40 74 50 0 70 96 ...
 $ triceps : num  35 29 0 23 35 0 32 0 45 0 ...
 $ insulin : num  0 0 0 94 168 0 88 0 543 0 ...
 $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
 $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
 $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

- Data has 768 rows (observations) and 9 columns/variables
- All columns contain numeric, continuous values, but variable ‘diabetes’ that is a class variable with two levels: ‘neg’ and ‘pos’

Data Description: Examples

- Tabulate your data
- Symbol '$' is used to select a specific column from your dataset
- <u>Example:</u> tabulate variable 'diabetes' from dataset 'PimaIndiansDiabetes'

```
> table(PimaIndiansDiabetes$diabetes)

neg pos
500 268
```

- There are 268 subjects diagnosed with diabetes and 500 without diabetes

Data Manipulation

Data Subsets

- From this point forward we will use library(dplyr) for data manipulation
  - You need to download and read the library
- Select only a certain column (variable)
- <u>Example:</u> select only column age from dataset 'PimaIndiansDiabetes'

```
> library(dplyr)
> select(PimaIndiansDiabetes, age)
    age
1    50
2    31
3    32
4    21
5    33
6    30
7    26
8    29
9    53
10   54
```

Data Subsets

- Select only certain rows of the data
- <u>Example:</u> select rows 1 to 5 from dataset ‘PimaIndiansDiabetes’

```
> slice(PimaIndiansDiabetes, 1:5)
  pregnant glucose pressure triceps insulin mass pedigree age diabetes
1        6     148       72      35       0 33.6    0.627  50      pos
2        1      85       66      29       0 26.6    0.351  31      neg
3        8     183       64       0       0 23.3    0.672  32      pos
4        1      89       66      23      94 28.1    0.167  21      neg
5        0     137       40      35     168 43.1    2.288  33      pos
```

Operators in R

Logical comparisons

< for less than
> for greater than
<= for less than or equal to
>= for greater than or equal to
== for equal to each other
!= not equal to each other
is.na() is NA
!is.na() is not NA.

Logical operators

value == 2**|**3; value equal 2 or (|) 3
&; means and. For example smoke == “0” & age > 25

Data Subsets

- Remove columns from data:
- <u>Example:</u> remove column ‘age’ from dataset ‘PimaIndiansDiabetes’

```
> dplyr::select(PimaIndiansDiabetes, -age)
    pregnant glucose pressure triceps insulin mass pedigree diabetes
1          6     148       72      35       0 33.6    0.627      pos
2          1      85       66      29       0 26.6    0.351      neg
3          8     183       64       0       0 23.3    0.672      pos
4          1      89       66      23      94 28.1    0.167      neg
5          0     137       40      35     168 43.1    2.288      pos
```

Data Subsets

- Keep only certain rows
- <u>Example:</u> select subjects only under 25 yrs. of age from dataset 'PimaIndiansDiabetes'

```
> filter(PimaIndiansDiabetes, age < 25)
    pregnant glucose pressure triceps insulin mass pedigree age diabetes
1          1      89       66      23      94 28.1    0.167  21      neg
2          1      97       66      15     140 23.2    0.487  22      neg
3          3      88       58      11      54 24.8    0.267  22      neg
4          2      71       70      27       0 28.0    0.586  22      neg
5          7     105        0       0       0  0.0    0.305  24      neg
```

Data Subsets

- Advanced filtering for rows
- <u>Example:</u> select all subjects under 25 yrs. with diabetes

```
> filter(PimaIndiansDiabetes, age < 25 & diabetes=="pos")
   pregnant glucose pressure triceps insulin mass pedigree age diabetes
1         0      95       85      25      36 37.4    0.247  24      pos
2         3     171       72      33     135 33.3    0.199  24      pos
3         0     113       76       0       0 33.3    0.278  23      pos
4         3     107       62      13      48 22.9    0.678  23      pos
5         0     140       65      26     130 42.6    0.431  24      pos
```

Ordering Data

- Again, we use library(dplyr)
- Order rows of a data according to one of the variables
- Example: order dataset 'PimaIndiansDiabetes' by number of pregnancies

```
> arrange(PimaIndiansDiabetes, pregnant)
    pregnant glucose pressure triceps insulin mass pedigree age diabetes
1          0     137       40      35     168 43.1    2.288  33      pos
2          0     118       84      47     230 45.8    0.551  31      pos
3          0     180       66      39       0 42.0    1.893  25      pos
4          0     100       88      60     110 46.8    0.962  31      neg
5          0     146       82       0       0 40.5    1.781  44      neg
```

- Use function desc() to arrange in descending order
- See R code for ordering by multiple variables/columns

Rename Variables

- Renaming a variable in a dataset
- <u>Example:</u> rename variable ‘mass’ from dataset ‘PimaIndiansDiabetes’ and save the new data under ‘new_diab’

```
> new_diab <- rename(PimaIndiansDiabetes, BMI = mass)
> head(new_diab)
  pregnant glucose pressure triceps insulin  BMI pedigree age diabetes
1        6     148       72      35       0 33.6    0.627  50      pos
2        1      85       66      29       0 26.6    0.351  31      neg
3        8     183       64       0       0 23.3    0.672  32      pos
4        1      89       66      23      94 28.1    0.167  21      neg
5        0     137       40      35     168 43.1    2.288  33      pos
6        5     116       74       0       0 25.6    0.201  30      neg
```

Data Transformations

- Sometimes you want to create new variables derived from existing ones
  - E.g., apply a log transformation to skewed variables
- <u>Example:</u> Take the log of ‘age’ from dataset ‘PimaIndiansDiabetes’

```
> PimaIndiansDiabetes <- mutate(PimaIndiansDiabetes, log_age = log(age))
> head(PimaIndiansDiabetes)
  pregnant glucose pressure triceps insulin mass pedigree age diabetes  log_age
1        6     148       72      35       0 33.6    0.627  50      pos 3.912023
2        1      85       66      29       0 26.6    0.351  31      neg 3.433987
3        8     183       64       0       0 23.3    0.672  32      pos 3.465736
4        1      89       66      23      94 28.1    0.167  21      neg 3.044522
5        0     137       40      35     168 43.1    2.288  33      pos 3.496508
6        5     116       74       0       0 25.6    0.201  30      neg 3.401197
```

Data Transformations: R Math

>log() – natural logarithm

>sqrt() – square root

>x^n – exponent

Matrix Operations:

>A%%B* - matrix multiplication

>t(A) – matrix transpose

>det(A) – determinant of A

>diag(A) – diagonal of A

>solve(A) – matrix inverse

IF-ELSE Function

- Very useful when you want the assignment statement to apply to *some* observations and *not all*
- <u>Example:</u> Create three age categories in dataset 'PimaIndiansDiabetes'
- Cat1: Age<29, Cat2: 30≤Age≤45, Cat3: Age>45

```
> PimaIndiansDiabetes$age_cat <- ifelse(PimaIndiansDiabetes$age < 29 , 1,
+                                ifelse((PimaIndiansDiabetes$age >=30 & PimaIndiansDiabetes$age <= 45), 2, 3))
>
> head(dplyr::select(PimaIndiansDiabetes, -log_age))
  pregnant glucose pressure triceps insulin mass pedigree age diabetes age_cat
1        6     148       72      35       0 33.6    0.627  50      pos       3
2        1      85       66      29       0 26.6    0.351  31      neg       2
3        8     183       64       0       0 23.3    0.672  32      pos       2
4        1      89       66      23      94 28.1    0.167  21      neg       1
5        0     137       40      35     168 43.1    2.288  33      pos       2
6        5     116       74       0       0 25.6    0.201  30      neg       2
```

Combining Data Sets

- You can combine data from different sources by ‘row’ (horizontally) or by ‘column (vertically)
- Make sure the dimensions are comparable
- You can also merge two datasets; this requires that both datasets have at least one variable in common (either character or numeric)

Combine by Column

<u>Example:</u> To dataset ‘PimaIndiansDiabetes’ append two additional variables: ‘ID’ and smoking status: 0-never smoked, 1-smoking history

```
> combo_col <- cbind(PimaIndiansDiabetes, ID, Smoking)
> head(combo_col)
  pregnant glucose pressure triceps insulin mass pedigree age diabetes  log_age
1        6     148       72      35       0 33.6    0.627  50      pos 3.912023
2        1      85       66      29       0 26.6    0.351  31      neg 3.433987
3        8     183       64       0       0 23.3    0.672  32      pos 3.465736
4        1      89       66      23      94 28.1    0.167  21      neg 3.044522
5        0     137       40      35     168 43.1    2.288  33      pos 3.496508
6        5     116       74       0       0 25.6    0.201  30      neg 3.401197
  age_cat ID Smoking
1       3  1       0
2       2  2       1
3       2  3       0
4       1  4       0
5       2  5       1
6       2  6       0
```

R Markdown

R Markdown: Reproducible Research

- R Markdown documents (.RMD files) combine formatted text, code + results, and figures
  - Save and execute code
  - Generate high quality reports
- From RStudio go to File -> New File -> R Markdown
  - Give it a title, and click “OK”
  - RStudio provides a template
- Click the ‘knit’ button to generate

R Markdown: Layout

YAML

Section Header

Regular text

Code chunk

Untitled1

Knit | Insert | Run

```
---
title: "Sample"
author: "Lizzy Gibson"
date: "5/24/2019"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

2:1 Sample R Markdown

R Markdown: Parts

- YAML header surrounded by '---' specifies
  - Title, Author, Date
  - Output type (HTML, PDF, Word)
  - Table of contents (optional)
- Text can be included regularly
- Code must be inside 'chunks'
  - Opt + Cmd + I for Mac
  - Ctrl + Alt + I for Windows
- # specifies section headings and subsections headings

R Markdown: Text Formatting

syntax

```
Plain text
End a line with two spaces to start a new paragraph.
*italics* and _italics_
**bold** and __bold__
superscript^2^
~~strikethrough~~
[link](www.rstudio.com)

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6

endash: --
emdash: ---
ellipsis: ...
inline equation: $A = \pi*r^{2}$
image: ![](path/to/smallorb.png)

horizontal rule (or slide break):

***
```

becomes

Plain text
End a line with two spaces to start a new paragraph.
italics and *italics*
bold and **bold**
superscript^2
~~strikethrough~~
link

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

endash: –
emdash: —
ellipsis: …
inline equation: $A = \pi * r^2$
image: 

horizontal rule (or slide break):

R Markdown: Text Formatting

- Name your chunks!
  - {r chunk_name}
- eval = FALSE: code will be displayed but not executed; results are not included.
- echo = FALSE: code will be executed but not displayed; results are included.
- include = FALSE: code will be executed but not displayed; results are not included.
- results = hide and fig.show = hide: prevents results and figures from being shown, respectively.
- collapse = TRUE: output will be collapsed into a single block at shown at the end of the chunk.
- message = FALSE and warning = FALSE: prevents messages and warnings from being displayed.

Quick Practice:

Use the R code provided for this intro to create your first R Markdown PDF!

THANK YOU