BGGN 213
Data analysis with R
Lecture 4

Barry Grant
UC San Diego

http://thegrantlab.org/bggn213

Recap From Last Time:

- Substitution matrices: Where our alignment match and mis-match scores typically come from
- Comparing methods: The trade-off between *sensitivity*, *selectivity* and *performance*
- Sequence motifs and patterns: Finding functional cues from conservation patterns
- Sequence profiles and position specific scoring matrices (PSSMs), Building and searching with profiles, Their advantages and limitations
- PSI-BLAST algorithm: Application of iterative PSSM searching to improve BLAST sensitivity
- Hidden Markov models (HMMs): More versatile probabilistic model for detection of remote similarities

Feedback

Today's Learning Goals

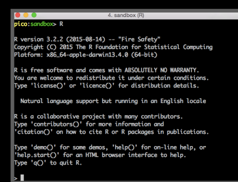
- Familiarity with R's basic syntax.
- Familiarity with major R data structures.
- Understand the basics of using functions.
- Be able to use R to read and parse comma-separated (.csv) formatted files ready for subsequent analysis.
- Appreciate how you can use R scripts to aid with reproducibility.

What is R?

R is a freely distributed and widely used programing **language** and **environment** for statistical computing, data analysis and graphics.

R provides an unparalleled interactive environment for data analysis.

It is script-based (*i.e.* driven by computer code) and not GUI-based (point and click with menus).



4. sandbox (R)

```
pico:sandbox> R

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

4. sandbox (R)

```
pico:sandbox> R

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Type “R” in your terminal

4. sandbox (R)

```
pico:sandbox> R

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Type “R” in your terminal

This is the R prompt

```
pico:sandbox> R

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

Type "R" in your terminal

This is the R prompt: Type **q()** to quit!

What R is **NOT**

A performance optimized software library for incorporation into your own C/C++ etc. programs.

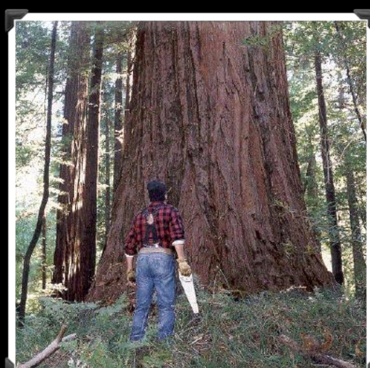
A molecular graphics program with a slick GUI.

Backed by a commercial guarantee or license.

Microsoft Excel!

What about Excel?

- Data manipulation is easy
- Can see what is happening
- **But**: graphics are poor
- Looping is hard
- Limited statistical capabilities
- Inflexible and irreproducible
- There are many many things Excel just cannot do!



Use the right tool!

Christie Bahlai @cbahlai · 2h
Weekly plug for scripted analyses:

Coauthor: "Can you change x,y,z about the analysis?"
Me [not crying]: "Yes." [changes 2 lines of code]

RETWEETS 11 FAVORITES 7

Rule of thumb: Every analysis you do on a dataset will have to be redone 10–15 times before publication. Plan accordingly!

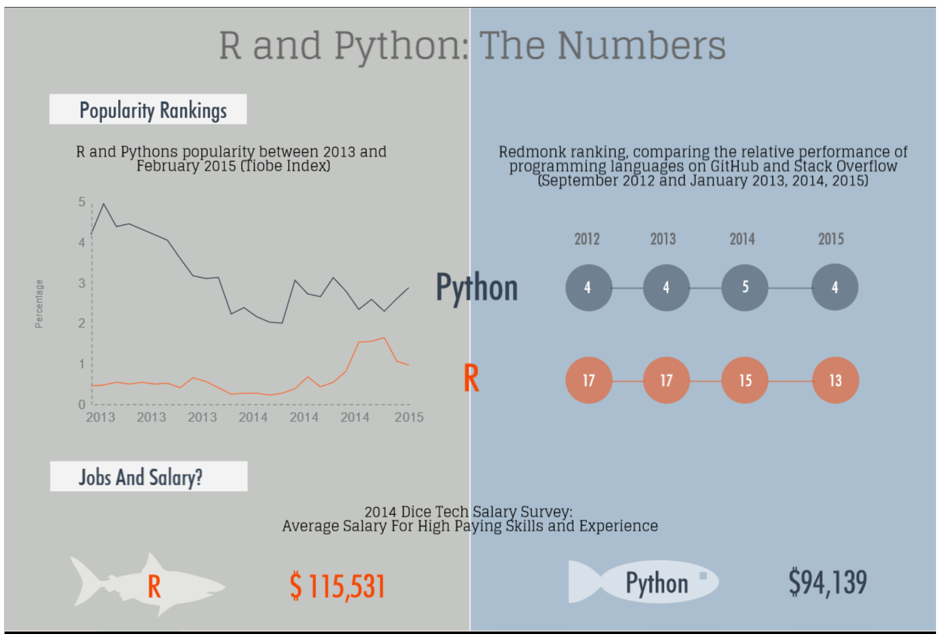
Why use R?

Productivity
Flexibility
Designed for data analysis

IEEE 2016 Top Programming Languages

Language Rank	Types	Spectrum Ranking
1. C		100.0
2. Java		98.1
3. Python		98.0
4. C++		95.9
5. R		87.9
6. C#		86.7
7. PHP		82.8
8. JavaScript		82.2
9. Ruby		74.5
10. Go		71.9

http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages



http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html?utm_medium=email&utm_source=flipboard

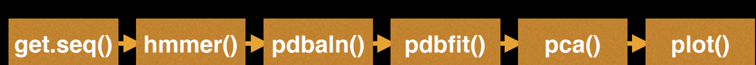
- R is the "lingua franca" of data science in industry and academia.
- Large user and developer community.
  - As of April 13th 2018 there are 12,481 add on **R packages** on **CRAN** and 1,473 on **Bioconductor** - more on these later!
- Virtually every statistical technique is either already built into R, or available as a free package.
- Unparalleled **exploratory data analysis** environment.

Modularity	Core R functions are modular and work well with others
Interactivity	R offers an unparalleled exploratory data analysis environment
Infrastructure	Access to existing tools and cutting-edge statistical and graphical methods
Support	Extensive documentation and tutorials available online for R
R Philosophy	Encourages open standards and reproducibility

Modularity	Core R functions are modular and work well with others
Interactivity	R offers an unparalleled exploratory data analysis environment
Infrastructure	Access to existing tools and cutting-edge statistical and graphical methods
Support	Extensive documentation and tutorials available online for R
R Philosophy	Encourages open standards and reproducibility

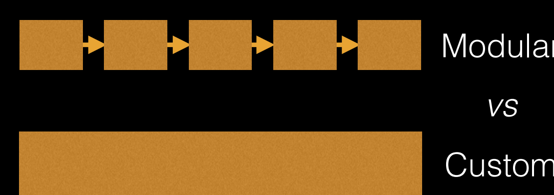
Modularity

R was designed to allow users to interactively build complex workflows by interfacing smaller ‘**modular’ functions** together.



An alternative approach is to write a **single complex program** that takes raw data as input, and after hours of data processing, outputs publication figures and a final table of results.

Which would you prefer and why?



Advantages/Disadvantages

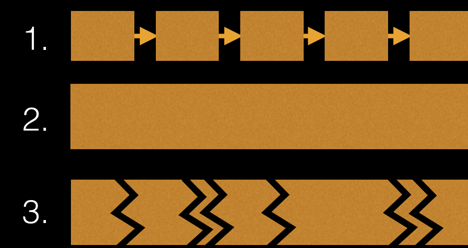
The 'monster approach' is customized to a particular project but results in massive, fragile and difficult to modify (therefore inflexible, untransferable, and error prone) code.

With **modular workflows**, it's easier to:

- Spot errors and figure out where they're occurring by inspecting intermediate results.
- Experiment with alternative methods by swapping out components.
- Tackle novel problems by remixing existing modular tools.

'Scripting' approach

Another common approach to bioinformatics data analysis is to write individual scripts in Perl/ Python/Awk/C etc. to carry out each subsequent step of an analysis



This can offer many advantages but can be challenging to make robustly modular and interactive.

Interactivity & exploratory data analysis

Learning R will give you the freedom to explore and experiment with your data.

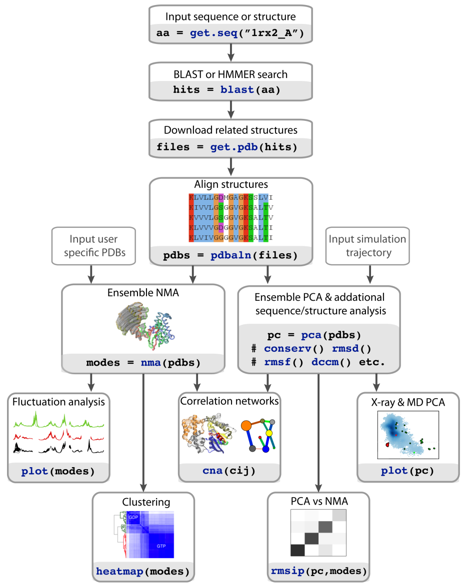
"*Data analysis, like experimentation, must be considered as a highly interactive, iterative process, whose actual steps are selected segments of a stubbily branching, tree-like pattern of possible actions*". [**J. W. Tukey**]

Interactivity & exploratory data analysis

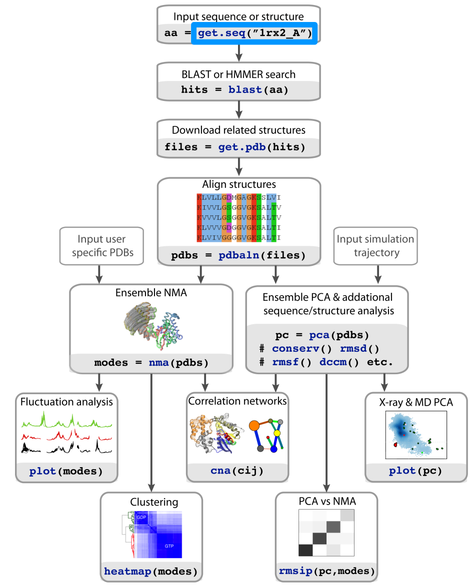
Learning R will give you the freedom to explore and experiment with your data.

"*Data analysis, like experimentation, must be considered as a highly interactive, iterative process, whose actual steps are selected segments of a stubbily branching, tree-like pattern of possible actions*". [**J. W. Tukey**]

Bioinformatics data is intrinsically **high dimensional** and frequently 'messy' requiring **exploratory data analysis** to find patterns - both those that indicate interesting biological signals or suggest potential problems.

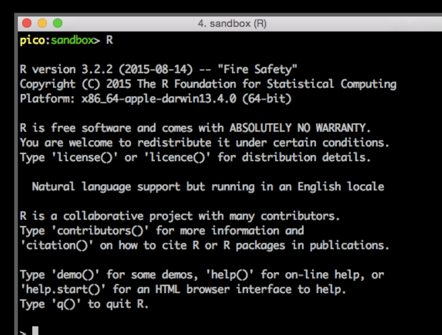


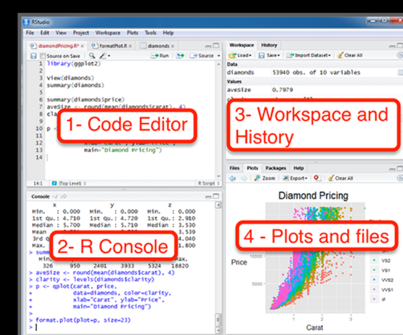
R Features = **functions()**



How do we use R?

Two main ways to use R

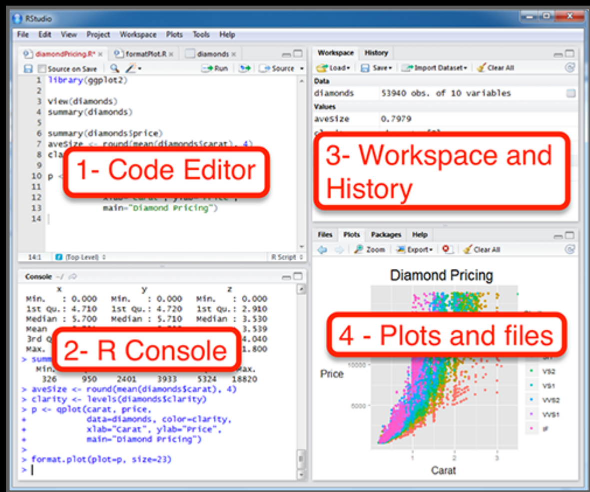




1. Terminal

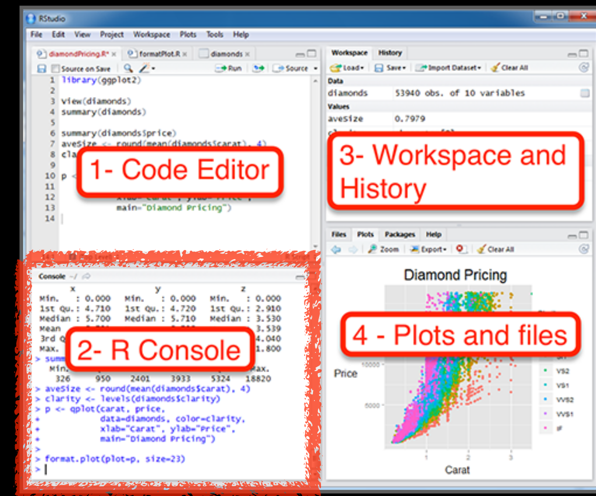
2. RStudio

We will use RStudio today



Lets get started…

Do it Yourself!



Some simple R commands

Do it Yourself!

R prompt!

1.
```
> 2+2
[1] 4
```
Result of the command

2.
```
> 3^2
[1] 9
```

3.
```
> sqrt(25)
[1] 5
```

4.
```
> 2*(1+1)
[1] 4
```

5.
```
> 2*1+1
[1] 3
```
Order of precedence

6.
```
> exp(1)
[1] 2.718282
```

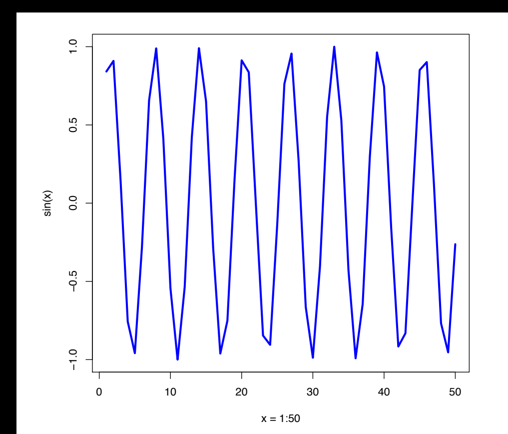
7.
```
> log(2.718282)
[1] 1
```

8.
```
> log(10, base=10)
[1] 1
```
Optional argument

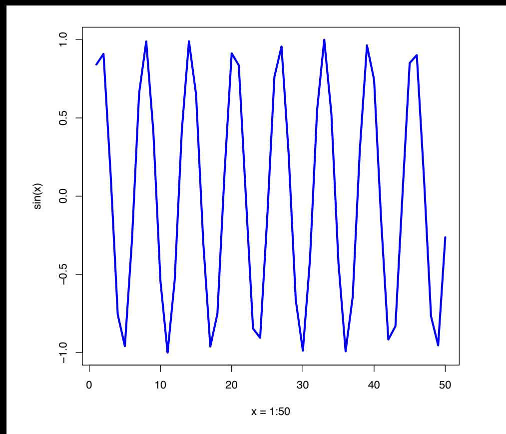
9.
```
> log(10
+      , base = 10)
[1] 1
```
Incomplete command

10.
```
> x=1:50
> plot(x, sin(x))
```

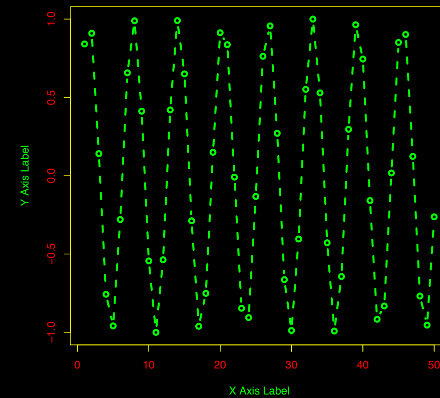
Does your plot look like this?



plot(x, sin(x), typ="l", col="blue", lwd=3, xlab="x = 1:50")



Options: ?plot ?plot.default



X Mean: 54.2659224
Y Mean: 47.8313999
X SD : 16.7649829
Y SD : 26.9342120
Corr. : -0.0642526

Key point: You need to visualize your data!

Error Messages

Sometimes the commands you enter will generate errors.
Common beginner examples include:

- Incomplete brackets or quotes *e.g.*

  ((4+8)*20 <enter>
  +
  This eturns a + here, which means you need to enter the remaining bracket - R is waiting for you to finish your input.
  Press <ESC> to abandon this line if you don't want to fix it.

- Not separating arguments by commas *e.g.*

  plot(1:10 col="red")

- Typos including miss-spelling functions and using wrong type of brackets *e.g.*

  exp{4}

Do it Yourself!

Your turn!

https://bioboot.github.io/bggn213_S18/class-material/04_rintro/

If you have done the introductory DataCamp course then feel free to jump to section #3 *Object Assignment*

Use when finished

Topics Covered:

Calling Functions
Getting help in R
Vectors and vectorization
Workspace and working directory
RStudio projects

Topics Covered:

Calling Functions
Getting help in R
Vectors and vectorization
Workspace and working directory
RStudio projects

Vectors

- Vectors are the most basic data structure in R
- All elements of a vector must be the same type

```
dbl_var <- c(1, 2.5, 4.5)
log_var <- c(TRUE, FALSE, T, F)
chr_var <- c("these are", "some", "strings")
```

- When you attempt to combine different types they will be coerced to the most flexible type.

```
var <- c(1, "G", "4", 0.05, TRUE)
```

Names

- You can name a vector in several ways:
  - When creating it: `x <- c(a = 1, b = 2, c = 3)`
  - By modifying an existing vector in place:

```
x <- 1:3; names(x) <- c("a", "b", "c")
```

- You can then use the names to access (subset) vector elements:

```
x [ c("b", "a") ]
```

Why is this useful?

- Because if you know the name (i.e. your label) then you don't have to remember which element of a vector the data you are after was stored in. Consider this *fictional* example:

```
> grades <- c(alice=80, barry=99, chandra=60, chris=100)
> grades["barry"]
barry
  99
> which.max(grades)
chris
   4
> sort(grades)
chandra   alice   barry   chris
     60      80      99     100
```

What would happen?

1.
```
> x <- 1:3; names(x) <- c("a", "b", "c", "d")
```
2.
```
> x <-1:3; names(x) <- 3:1;  x[3]
```
3.
```
> x["3"]
```

R has many data structures

These included:

- **vector**
- **data frame**
- list
- matrix
- factors

data.frame

- **data.frame** is the *de facto* data structure for most **tabular data** and what we use for statistics and plotting with **ggplot2** - more on this later!
- Arguably the most important R data structure
- Data frames can have additional attributes such as **rownames()** and **colnames()**, which can be useful for annotating data, with things like subject_id or sample_id

data.frame continued…

Do it Yourself!

- Created with the function **data.frame()**

```
dat <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
```

- Or more commonly when reading delimited files (*i.e.* **importing data**) with the functions **read.csv()**, **read.table()**, **read_xlsx()** *etc…*

```
dep <- read.csv2("http://bio3d.uib.no/data/pdb_deposition2.csv")
```

- R Studio can do this for you via:
  File > Import Dataset > From CSV…

Useful **data.frame** Functions

- **head()** -and **tail()** shows first 6 rows and last 6 rows respectively
- **dim()** - returns the dimensions (i.e. number of rows and columns)
- **nrow()** and **ncol()** returns the number of rows and columns separately.
- **rownames()** and **colnames()**- shows the names attribute for rows and columns
- **str()** - returns the structure including name, type and preview of data in each column

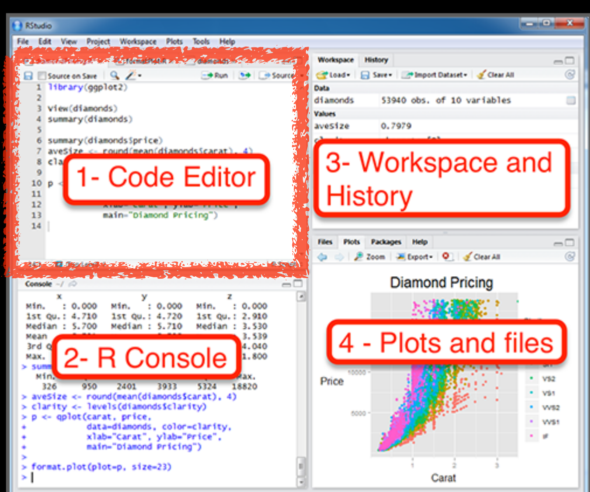
Topics Covered:

Calling Functions
Getting help in R
Vectors and vectorization
Workspace and working directory
RStudio projects

Topics Covered:

Calling Functions
Getting help in R
Vectors and vectorization
Workspace and **working directory**
RStudio **projects**

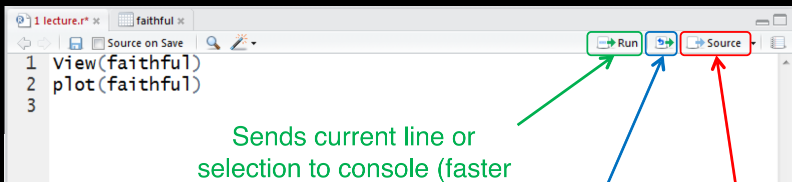
Side-note: Use the code editor for R scripts



R scripts

- A simple text file with your R commands (*e.g.* lecture7.r) that contains your R code for one complete analysis
- **Scientific method**: complete record of your analysis
- **Reproducible**: rerunning your code is easy for you or someone else
- In RStudio, select code and type <ctrl+enter> to run the code in the R console
- **Key point**: Save your R script!

Side-note: RStudio shortcuts



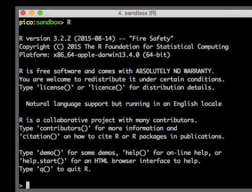
Sends current line or selection to console (faster to type: **command/ctrl+enter**)

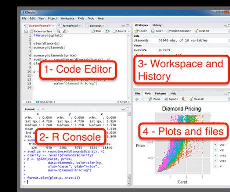
Sends entire file to console

Re-send the lines of code you last ran to the console (useful after edits)

Other RStudio shortcuts!
Up/Down arrows (recall cmds)
Ctrl + 2 (move cursor to console)
Ctrl +1 (move cursor to editor)

Rscript: Third way to use R





> Rscript --vanilla my_analysis.R

1. Terminal **2. RStudio** **3. Rscript**

From the command line!

> Rscript --vanilla **my_analysis.R**

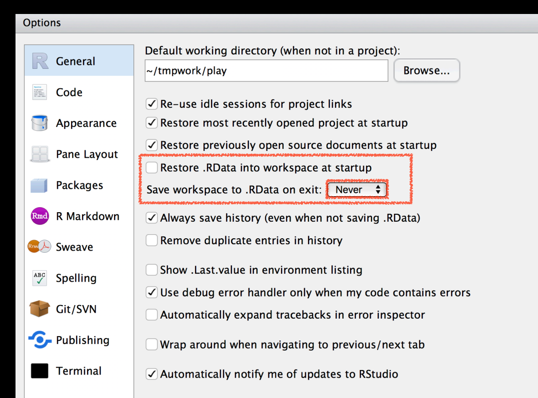
or within R: source(**my_analysis.R**)

R workspaces

- When you close RStudio, SAVE YOUR .R SCRIPT
- You can also save data and variables in an R workspace, but this is generally not recommended
- Exception: working with an enormous dataset
- Better to start with a clean, empty workspace so that past analyses don't interfere with current analyses
- rm(list = ls()) clears out your workspace
- You should be able to reproduce everything from your R script, so save your R script, don't save your workspace!

R workspaces

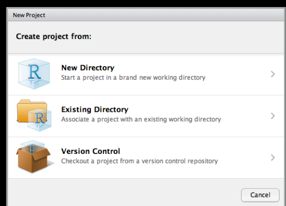
- Set **Tools** > **Global Options**

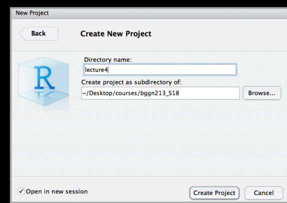


RStudio Projects

- We will use a new RStudio **project** for each new class going forward.

  File > **New Project** > **New Directory** > **New Project**…





- These projects will help keep us **organized** and divide our work into multiple contexts, each with their own working directory, workspace, history, and source documents.

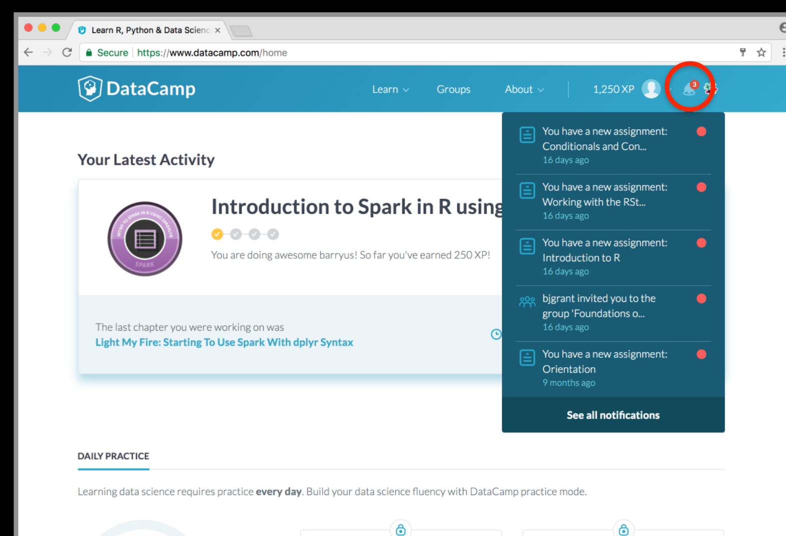
Learning Resources

- **TryR**. An excellent interactive online R tutorial for beginners.
  < http://tryr.codeschool.com/ >
- **RStudio**. A well designed reference card for RStudio.
  < https://help.github.com/categories/bootcamp/ >
- **DataCamp**. Online tutorials using R in your browser.
  < https://www.datacamp.com/ >
- **R for Data Science**. A new O'Reilly book that will teach you how to do data science with R, by Garrett Grolemund and Hadley Wickham.
  < http://r4ds.had.co.nz/ >

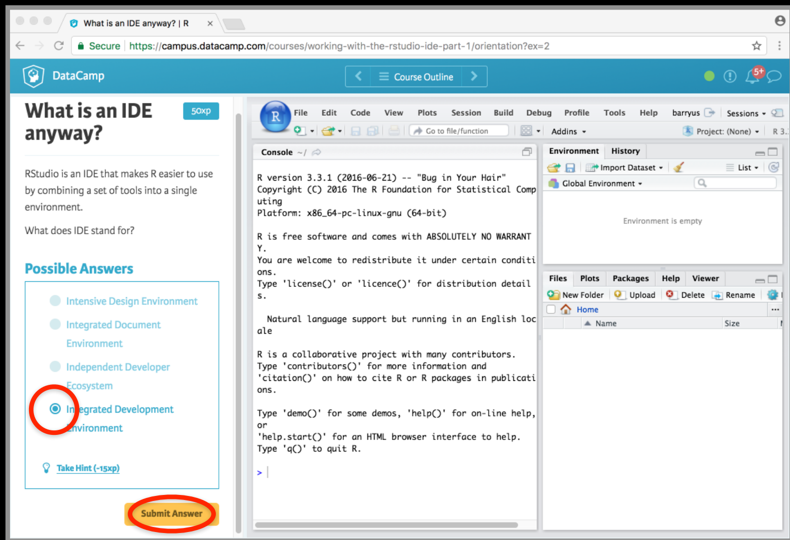
Learning Resources

- **TryR**. An excellent interactive online R tutorial for beginners.
  < http://tryr.codeschool.com/ >
- **RStudio**. A well designed reference card for RStudio.
  < https://help.github.com/categories/bootcamp/ >
- **DataCamp**. Online tutorials using R in your browser.
  < https://www.datacamp.com/ >
- **R for Data Science**. A new O'Reilly book that will teach you how to do data science with R, by Garrett Grolemund and Hadley Wickham.
  < http://r4ds.had.co.nz/ >

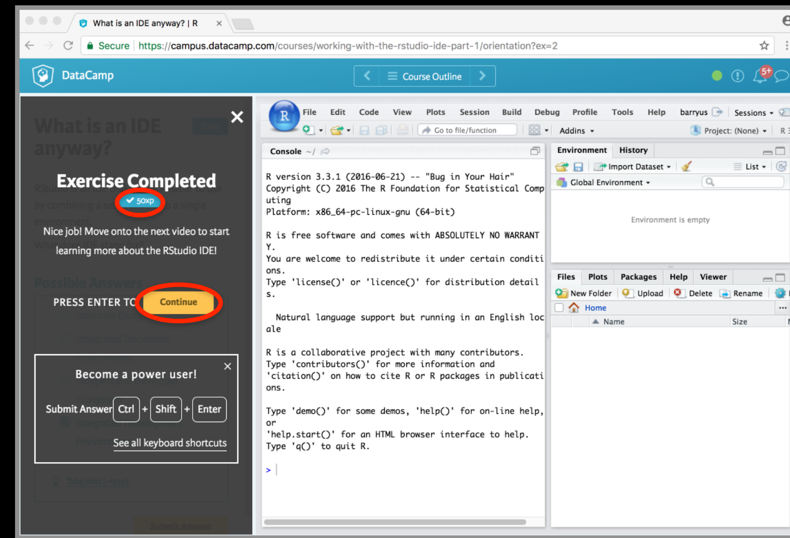
< https://www.datacamp.com/ >



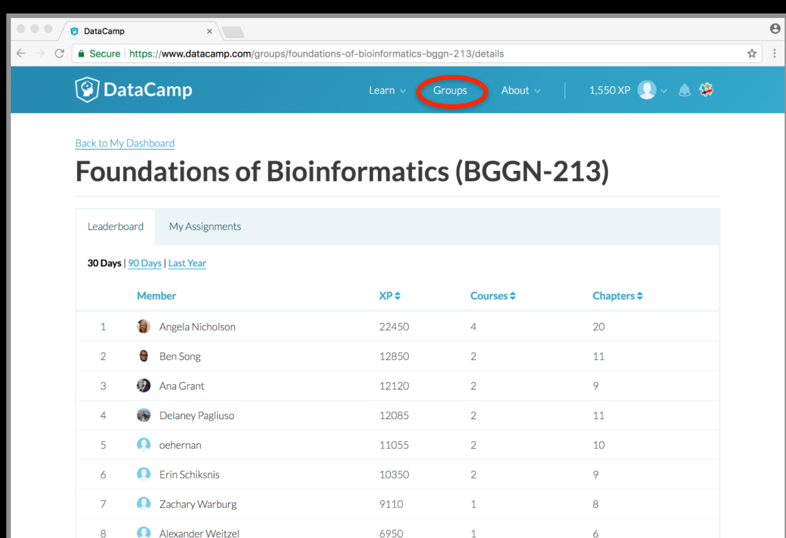
< https://www.datacamp.com/ >



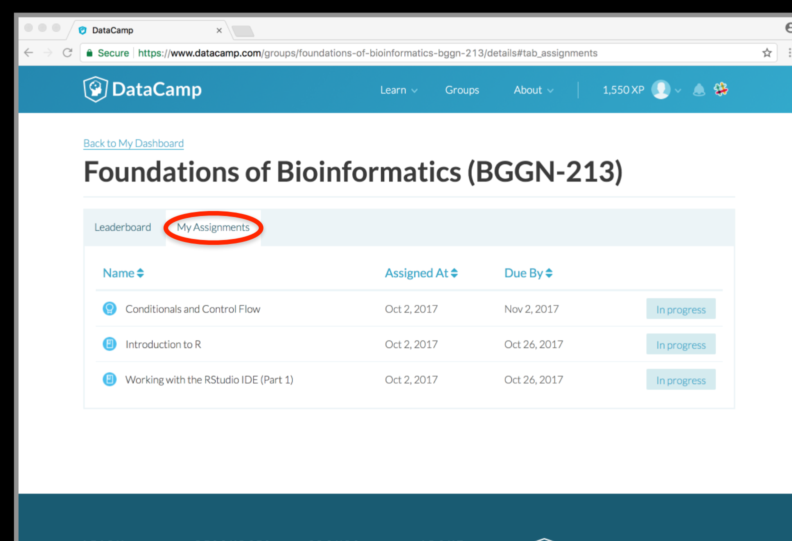
< https://www.datacamp.com/ >



< https://www.datacamp.com/ >



< https://www.datacamp.com/ >



Key Points

- R's basic data types are **logical**, **character**, **numeric**, integer and complex.
- R's basic data structures include **vectors**, lists, **data frames**, matrices and factors.
- Objects may have attributes, such as **name**, **dim**ension, and **class**.
- **DataCamp**, StackOverflow and **help()** are your friends.

Final Knowledge Check!

- What is R and why should we use it?
- Familiarity with R's basic syntax.
- Familiarity with major R data structures namely *vectors* and *data.frames* (with more on *lists* and *matrices* next day).
- Understand the basics of using functions (arguments, vectorizion and re-cycling).
- Be able to use R to read and parse comma-separated (.csv) formatted files ready for subsequent analysis.
- Appreciate how you can use R scripts to aid with reproducibility.

Link: Muddy point assessment

Optional!

http://swcarpentry.github.io/r-novice-inflammation/

Sections: 1, 11 & 12 only!

Help from within R

- Getting help for a function

```
> help("log")
> ?log
```

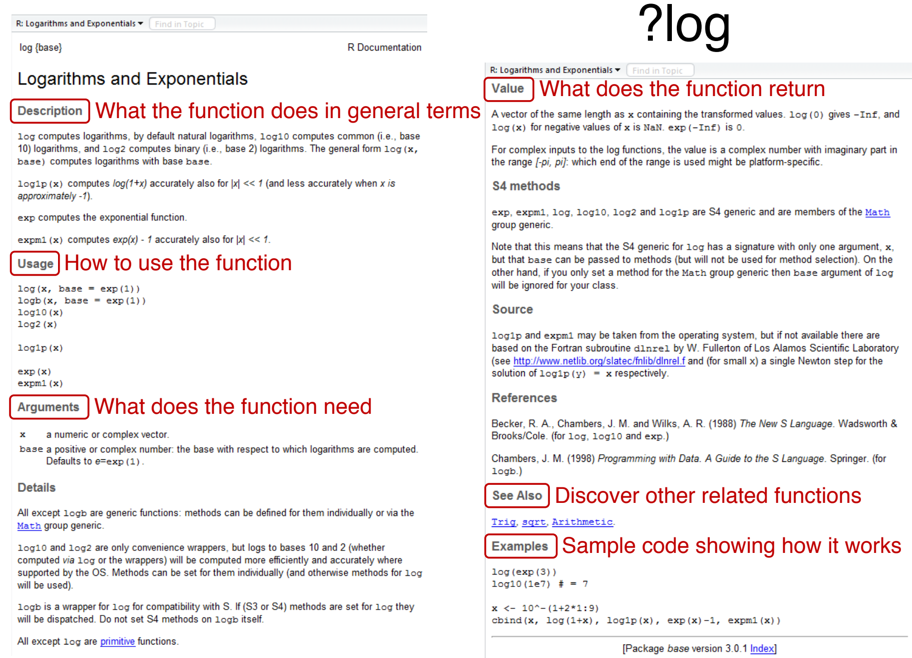
- Searching across packages

```
> help.search("logarithm")
```

- Finding all functions of a particular type

```
> apropos("log")
[7] "SSlogis" "as.data.frame.logical" "as.logical"
   "as.logical.factor" "dlogis" "is.logical"
[13] "log" "log10" "log1p" "log2" "logLik" "logb"
[19] "logical" "loglin" "plogis" "print.logLik" "qlogis"
   "rlogis"
```

?log



Optional Exercise

Use R to do the following. Create a new script to save your work and code up the following four equations:

$$1 + 2(3 + 4)$$

$$\ln(4^3+3^{2+1})$$

$$\sqrt{(4+3)(2+1)}$$

$$\left(\frac{1+2}{3+4}\right)^2$$