



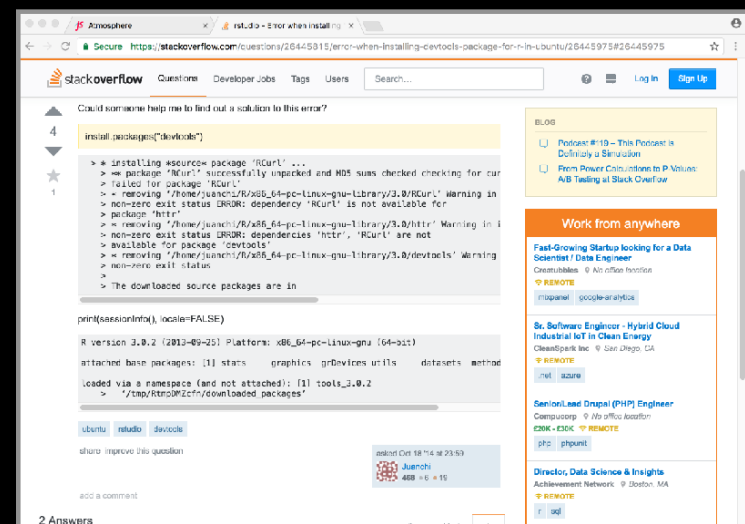
## Recap From Last Time:

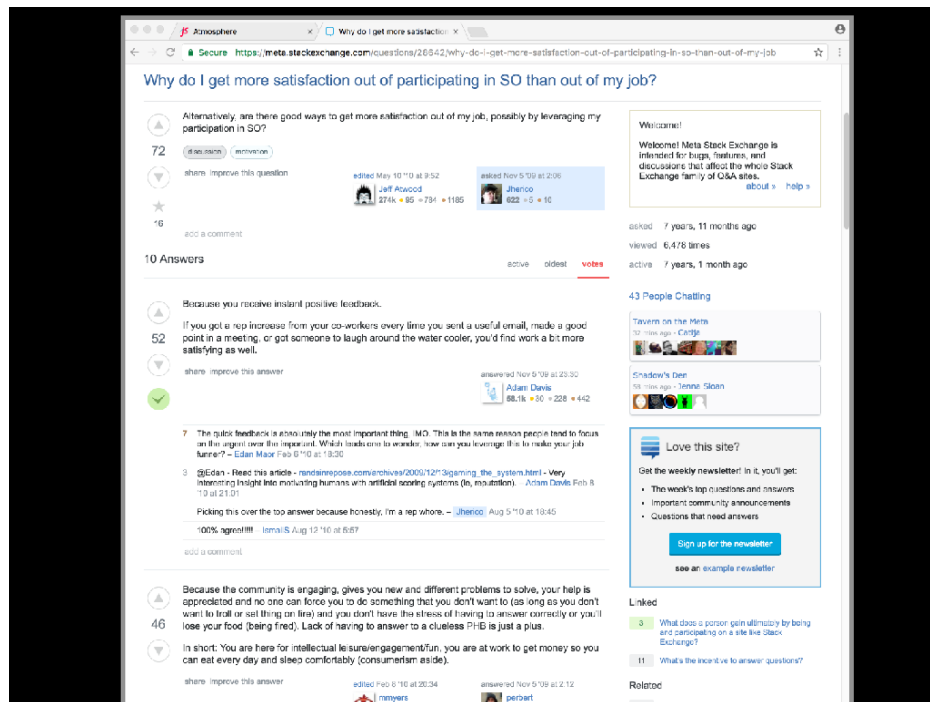
- **UNIX pipes and redirects:** How UNIX commands can be combined to generate flexible solutions to data manipulation tasks.
- **UNIX commands:** Further exploration of the 22 key UNIX commands that you will use during ~95% of your future UNIX work
- **Jetstream:** Starting up instances; ssh access from your Terminal application; *Demoed* installing and running bioinformatics software for a genome scale annotation.
- **Cloud computing:** Many bioinformatic tasks require large amounts of computing power and can't realistically be run on your own machine. These tasks are best performed using remote computers or cloud computing, which can only be accessed through a shell.

## 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.

## Side-Note: StackOverflow is your friend!




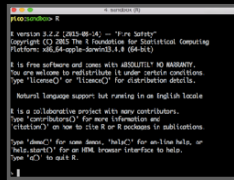


# What is R?

R is a freely distributed and widely used programming **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).

The screenshot shows the R console window titled "4. sandbox (R)". The prompt is "pico:sandbox> R". The output displays the R version (3.2.2, 2015-08-14), copyright (© 2015 The R Foundation for Statistical Computing), and platform (x86\_64-apple-darwin13.4.0 (64-bit)). It also includes a disclaimer about the lack of warranty and the freedom to redistribute under certain conditions. The text mentions natural language support but running in an English locale. It lists various functions for getting help, such as 'demo()', 'help()', 'help.start()', and 'q()'. The prompt is currently at "> |".

```
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.

> |
```

```
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.

> |
```

## 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!

**54** 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

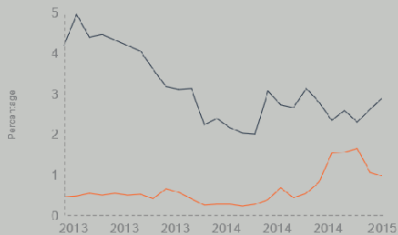
| Language Rank | Types | Spectrum Ranking |
|---------------|-------|------------------|
| 1. C          |       | 100.0            |
| 2. Java       |       | 98.1             |
| 3. Python     |       | 98.0             |
| 4. C++        |       | 95.9             |
| 5. R          |       | 87.9             |
| 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>

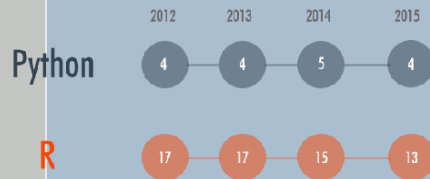
## R and Python: The Numbers

### Popularity Rankings

R and Python's popularity between 2013 and February 2015 (TIOBE Index)



Redmonk ranking, comparing the relative performance of programming languages on GitHub and Stack Overflow (September 2012 and January 2013, 2014, 2015)



### Jobs And Salary?

2014 Dice Tech Salary Survey:  
Average Salary For High Paying Skills and Experience



\$115,531



\$94,139

[http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html?utm\\_medium=email&utm\\_source=flipboard](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 Aug 1st 2016 there are 8811 add on **R packages** on **CRAN** and 1211 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.

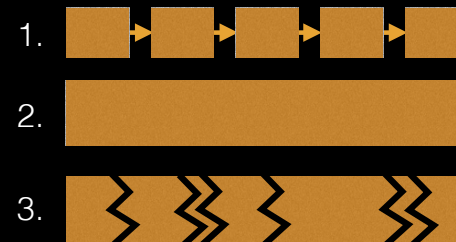
```
get.seq() → hmmer() → pdbaln() → pdbfit() → pca() → plot()
```

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.

All-in-one custom '**Monster**' program

# '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 stabbily branching, tree-like pattern of possible actions". [J. W. Tukey]*

## Interactivity & exploratory data analysis

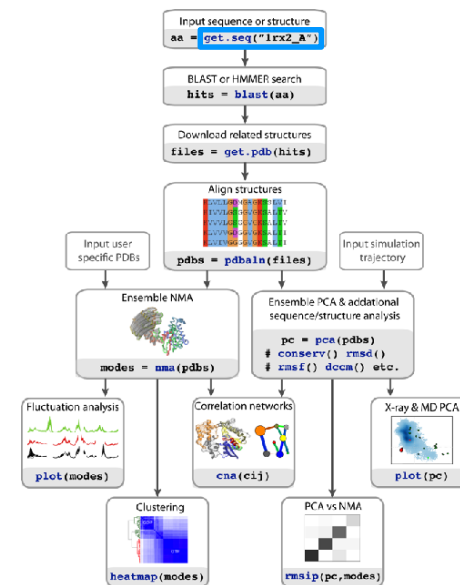
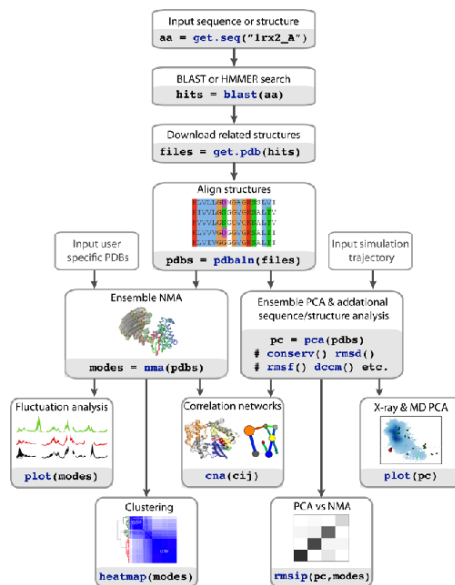
Learning R will give you the freedom to explore and experiment with your data.

*"Data analysis, like experimentation, must be considered as a highly interactive, iterative process, whose actual steps are selected segments of a stabbily 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

```

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.

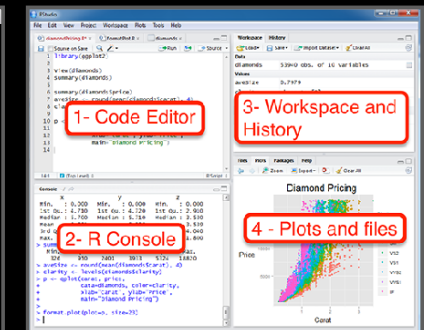
Natural language support but running in an English locale

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.

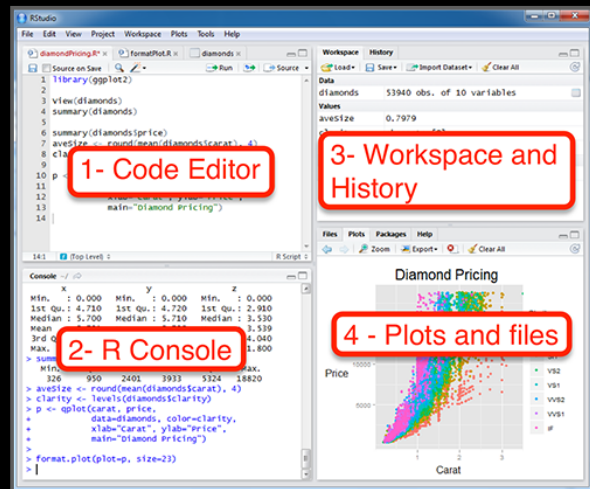
>
    
```

**1. Terminal**



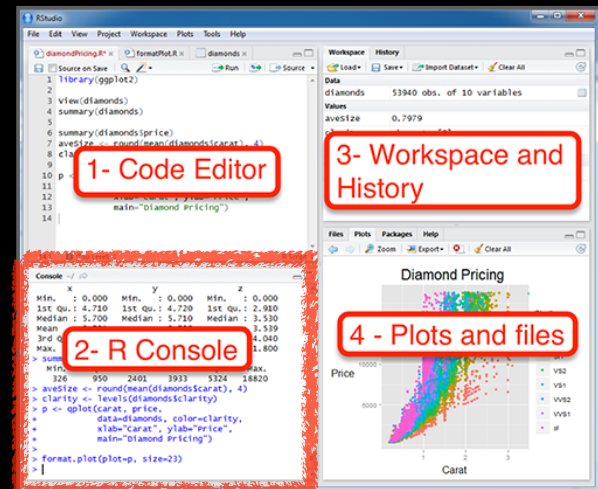
**2. RStudio**

# We will use RStudio today



# Lets get started...

Do it Yourself!

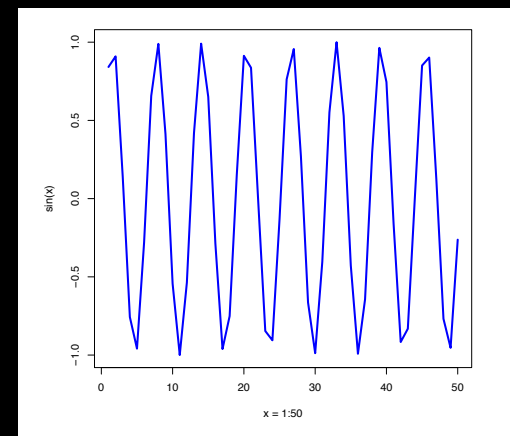


## Some simple R commands

Do it Yourself!

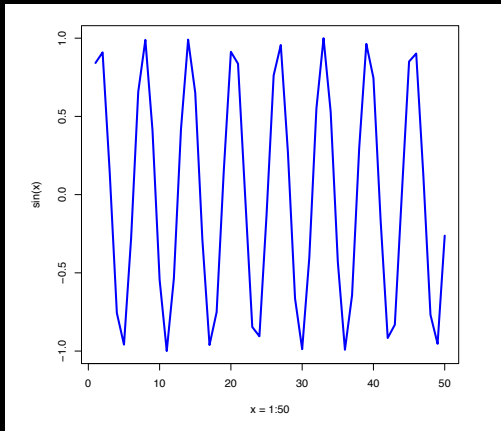
- R prompt!**
- `> 2+2`  
[1] 4 **Result of the command**
  - `> 3^2`  
[1] 9
  - `> sqrt(25)`  
[1] 5
  - `> 2*(1+1)`  
[1] 4
  - `> 2*1+1` **Order of precedence**  
[1] 3
  - `> exp(1)`  
[1] 2.718282
  - `> log(2.718282)`  
[1] 1
  - `> log(10, base=10)` **Optional argument**  
[1] 1
  - `> log(10 + , base = 10)` **Incomplete command**  
[1] 1
  - `> x=1:50`  
`> plot(x, sin(x))`

Does your plot look like this?

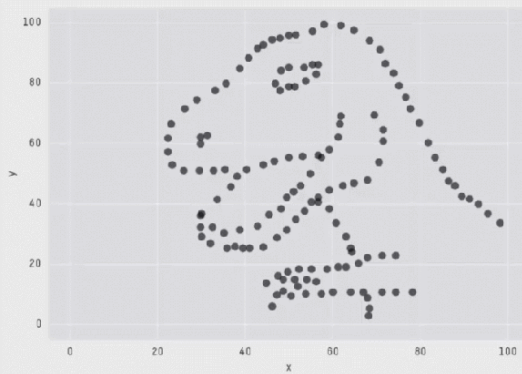
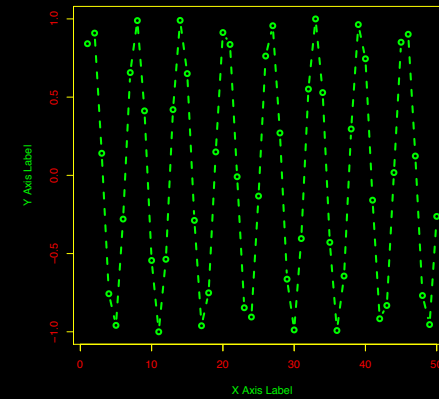




```
plot(x, sin(x), typ="l", col="blue", lwd=3, xlab="x = 1:50")
```



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!



Learning a new  
language is hard!

# 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 returns 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!

<http://tinyurl.com/bgggn213-rintro>

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

### 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 include:

- **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...**

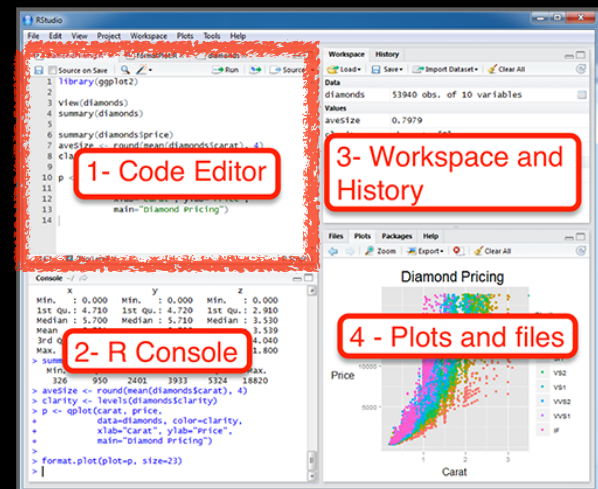
## Useful data.frame Functions

- **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

# 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**, **dimension**, and **class**.

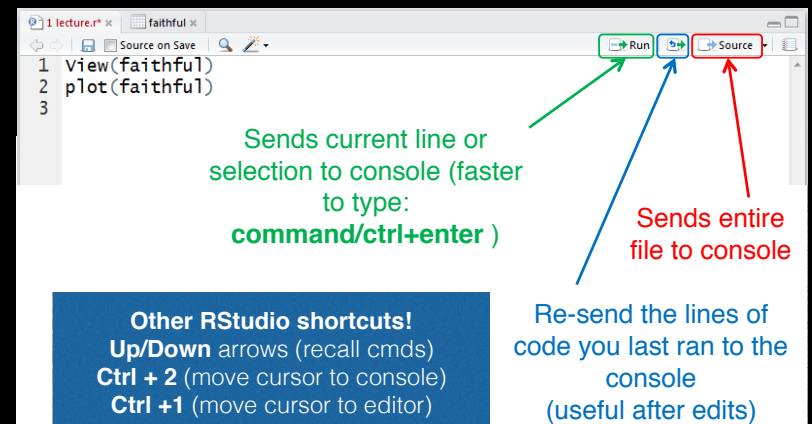
## Side-note: Use the code editor for R scripts



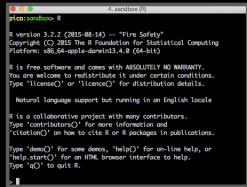
# R scripts

- 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!

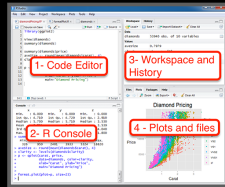
## Side-note: RStudio shortcuts



# Rscript: Third way to use R



1. Terminal



2. RStudio

> Rscript --vanilla my\_analysis.R

3. Rscript

From the command line!

> Rscript --vanilla my\_analysis.R  
# or within R: **source(my\_analysis.R)**

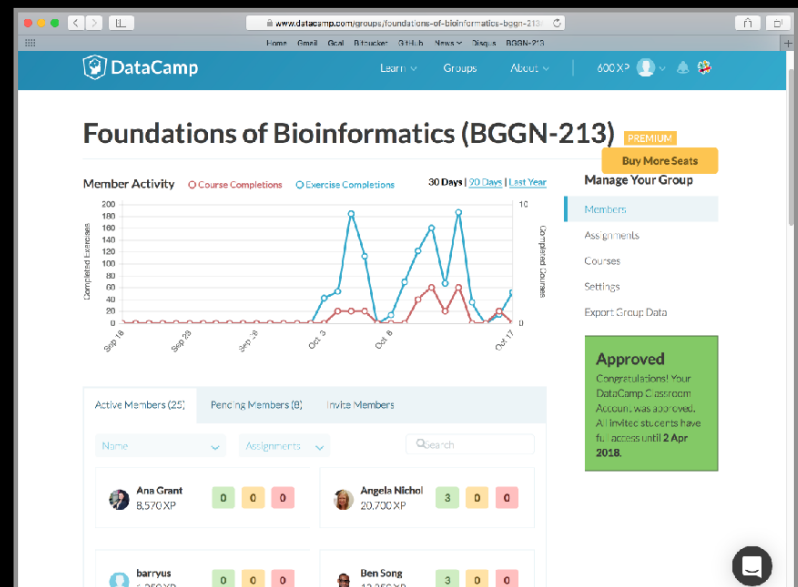
# Side-Note: R workspaces

- 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!**

# 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/> >

**Foundations of Bioinformatics (BGGN-213)** PREMIUM Buy More Seats

**Assignments** Create Assignment

| Name                                  | Assigned At | Due By       | Progress |
|---------------------------------------|-------------|--------------|----------|
| Conditionals and Control Flow         | Oct 2, 2017 | Nov 2, 2017  | 3/10     |
| Introduction to R                     | Oct 2, 2017 | Oct 26, 2017 | 8/10     |
| Working with the RStudio IDE (Part 1) | Oct 2, 2017 | Oct 26, 2017 | 3/10     |

**Manage Your Group**

- Members
- Assignments
- Courses
- Settings
- Export Group Data

**Approved**  
Congratulations! Your DataCamp Classroom Account was approved. All invited students have full access until 2 Apr 2018.

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

**Your Latest Activity**

**Introduction to Spark in R using**  
You are doing awesome barryyus! So far you've earned 250 XP!

The last chapter you were working on was **Light My Fires Starting To Use Spark With dplyr Syntax**

**DAILY PRACTICE**  
Learning data science requires practice every day. Build your data science fluency with DataCamp practice mode.

**Notifications:**

- You have a new assignment: Conditionals and Control Flow (14 days ago)
- You have a new assignment: Working with the RStudio IDE (15 days ago)
- You have a new assignment: Introduction to R (15 days ago)
- Barryyus invited you to the group 'Foundations of Bioinformatics' (15 days ago)
- You have a new assignment: Orientation (9 months ago)

[See all notifications](#)

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

**What is an IDE anyway?** Stop

RStudio is an IDE that makes R easier to use by combining a set of tools into a single environment.

What does IDE stand for?

**Possible Answers**

- Intensive Design Environment
- Integrated Document Environment
- Independent Developer Ecosystem
- Integrated Development Environment**

[Take Hint \(-100%\)](#) [Submit Answer](#)

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

**Exercise Completed** Stop

Nice job! Move onto the next video to start learning more about the RStudio IDE!

[Continue](#)

**Become a power user!**  
Submit Answer Ctrl Shift Enter  
See all keyboard shortcuts

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

The screenshot shows the DataCamp website interface for the 'Foundations of Bioinformatics (BGGN-213)' course. The page features a header with the DataCamp logo and navigation links. Below the header, there's a section for 'Assignments' with a table listing active assignments. The table has columns for 'Name', 'Assigned At', and 'Due By'. There are also buttons for 'Create Assignment' and 'Manage Your Group'. A sidebar on the right contains links for 'Members', 'Assignments', 'Courses', 'Settings', and 'Export Group Data'. A green notification box on the right says 'Approved' and mentions 'Congratulations! Your DataCamp Classroom Account was approved.'

| Name                                  | Assigned At | Due By       | Progress |
|---------------------------------------|-------------|--------------|----------|
| Conditionals and Control Flow         | Oct 2, 2017 | Nov 2, 2017  | 3 0 0    |
| Introduction to R                     | Oct 2, 2017 | Oct 26, 2017 | 8 0 0    |
| Working with the RStudio IDE (Part 1) | Oct 2, 2017 | Oct 26, 2017 | 3 0 0    |

## 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**, **dimension**, 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, vectorization 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.

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

Sections: 1, 11 & 12 only!

Optional!!

# 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"
```

## Optional Exercise

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

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

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

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

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

R: Logarithms and Exponentials [Find in Topics](#)

log [base]

R Documentation

### Logarithms and Exponentials

**Description** What the function does in general terms

log computes logarithms, by default natural logarithms. log10 computes common (i.e., base 10) logarithms, and log2 computes binary (i.e., base 2) logarithms. The general form log(x, base) computes logarithms with base base.

log1p(x) computes log(1+x) accurately also for |x| << 1 (and less accurately when x is approximately -1).

exp computes the exponential function.

expm1(x) computes exp(x) - 1 accurately also for |x| << 1.

**Usage** How to use the function

```
log(x, base = exp(1))
logb(x, base = exp(2))
log10(x)
log2(x)
log1p(x)
exp(x)
expm1(x)
```

**Arguments** What does the function need

x a numeric or complex vector.

base a positive or complex number: the base with respect to which logarithms are computed. Defaults to exp(1).

**Details**

All except logb are generic functions: methods can be defined for them individually or via the [S4](#) group generic.

log1p and log2 are only convenience wrappers, but logs to bases 10 and 2 (whether computed via log or the wrappers) will be computed more efficiently and accurately where supported by the OS. Methods can be set for them individually (and otherwise methods for log will be used).

logp is a wrapper for log for compatibility with S. If (S3 or S4) methods are set for log they will be dispatched. Do not set S4 methods on logb itself.

All except log are [primitive](#) functions.

## ?log

R: Logarithms and Exponentials [Find in Topics](#)

**Value** What does the function return

A vector of the same length as x containing the transformed values. log(0) gives Inf, and log(x) for negative values of x is NaN; exp(-Inf) is 0.

For complex inputs to the log functions, the value is a complex number with imaginary part in the range  $[-\pi, \pi]$ , which end of the range is used might be platform-specific.

**S4 methods**

exp, expm1, log, log10, log2 and log1p are S4 generic and are members of the [stats](#) group generic.

Note that this means that the S4 generic for log uses a signature with only one argument, x, but that base can be passed to methods (but will not be used for method selection). On the other hand, if you only set a method for the stats group generic then base argument of log will be ignored for your class.

**Source**

log1p and expm1 may be taken from the operating system, but if not available there are based on the Fortran subroutine dlnxcl by W. Hutton at Los Alamos Scientific Laboratory (see <http://www.netlib.org/slatec/filib/dlnxcl.f> and for small x) a single Newton step for the solution of log1p(y) = x respectively.

**References**

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole. (for log, log10 and exp.)

Chambers, J. M. (1998) *Programming with Data: A Guide to the S Language*. Springer. (for logb.)

**See Also** Discover other related functions

[log](#), [logb](#), [arithmetic](#)

**Examples** Sample code showing how it works

```
log(exp(3))
log10(1+7) # -7
x <- 10^-(1+2+1+9)
rbind(x, log(1+x), log1p(x), exp(x)-1, expm1(x))
```

[Package base version 3.0.1 [Index](#)]