



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 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).

```
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.

> |
```



```
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.
```

```
> █
```

```
pico:sandbox> R
```

Type "R" in your terminal

```
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.
```

```
> █
```



```
pico:sandbox> R
```

Type "R" in your terminal

```
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.
```

```
> █
```

This is the R prompt

pico:sandbox> R

Type "R" in your terminal

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.

> █

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!



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

R is designed specifically for data analysis

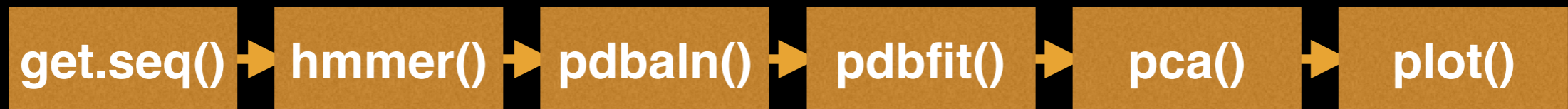
- Large friendly user and developer community.
 - As of Jan 6th 2019 there are 13,645 add on **R packages** on CRAN and 1,649 on Bioconductor - much more on these later!
- Virtually every statistical technique is either already built into R, or available as a free package.
- Unparalleled data analysis environment for **high-throughput genomic data**.

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.



All-in-one custom 'Monster' program

Which would you prefer and why?



Modular

vs



Custom

Advantages/Disadvantages

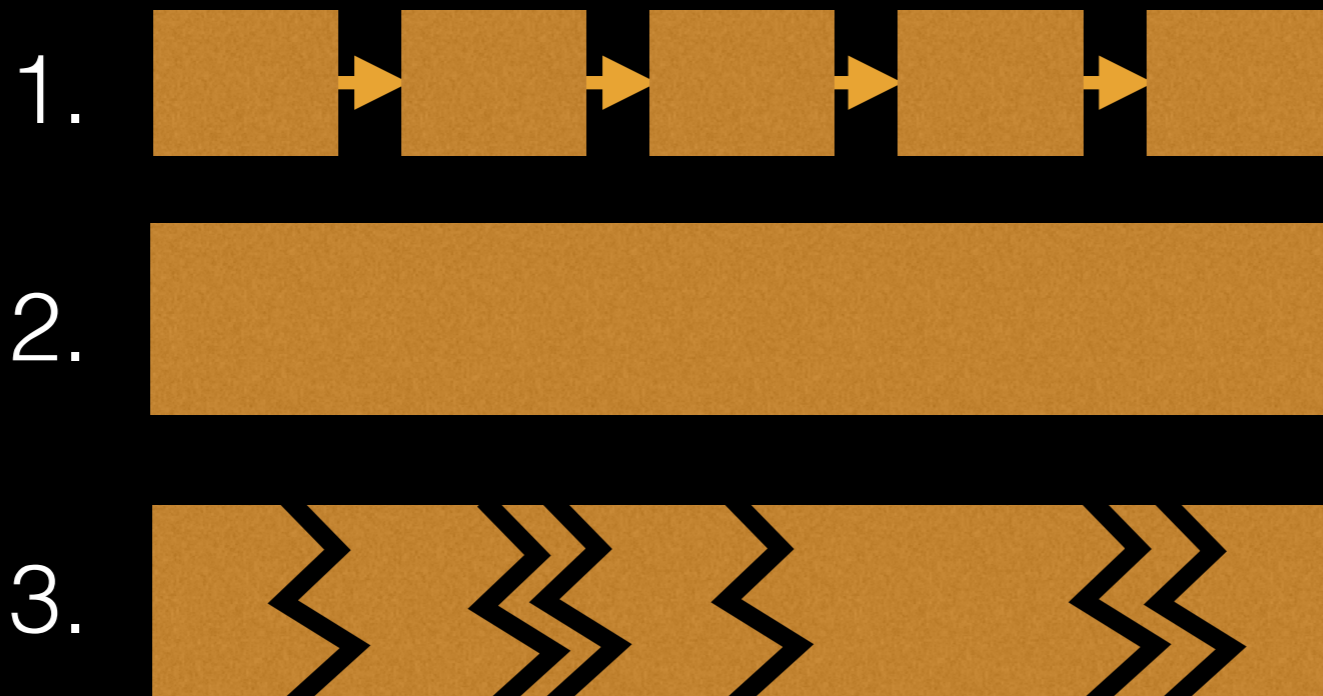
The 'monster approach' is **customized to a particular project** but results in **massive, fragile** and difficult to modify (therefore **inflexible, untransferable, and error prone**) code.

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.

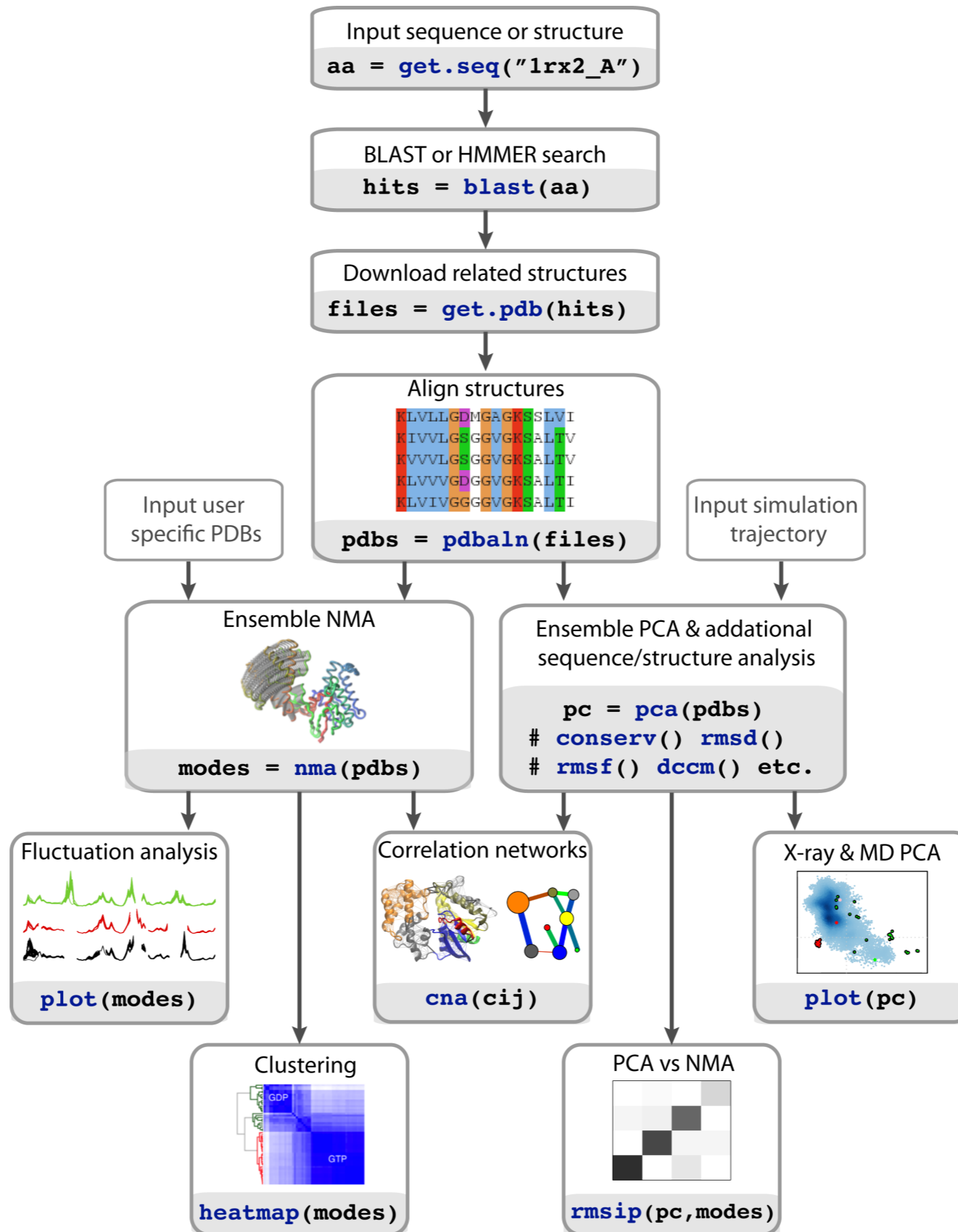
“Data analysis, like experimentation, must be considered as a highly interactive, iterative process, whose actual steps are selected segments of a stubbornly branching, tree-like pattern of possible actions”. [**J. W. Tukey**]

Interactivity & exploratory data analysis

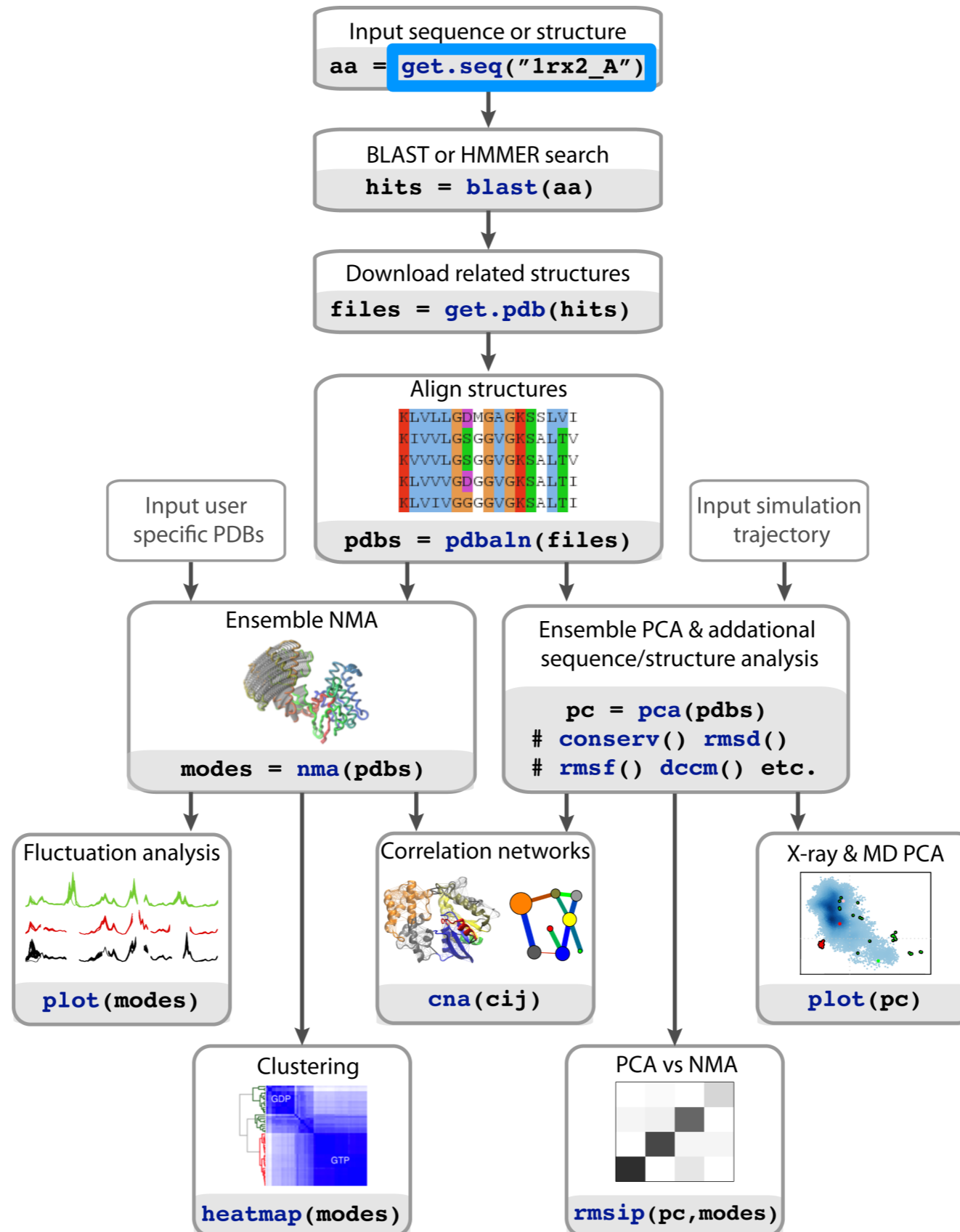
Learning R will give you the freedom to explore and experiment with your data.

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

```
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.

>
```

The screenshot shows the RStudio interface with several components:

- Code Editor:** Contains R code for loading ggplot2, viewing and summarizing the 'diamonds' dataset, and creating a scatter plot of Price vs. Carat, colored by clarity. A red box labeled "1- Code Editor" points to this area.
- R Console:** Shows the output of the code, including summary statistics for the 'diamonds' dataset. A red box labeled "2- R Console" points to this area.
- Workspace and History:** Shows the 'diamonds' dataset loaded into the workspace. A red box labeled "3- Workspace and History" points to this area.
- Plots and Files:** Displays a scatter plot titled "Diamond Pricing" showing Price on the y-axis and Carat on the x-axis, with points colored by clarity. A red box labeled "4 - Plots and files" points to this area.

1. Terminal

2. RStudio

We will use **RStudio** today

The image shows the RStudio interface with four red callout boxes highlighting key features:

- 1- Code Editor:** The top-left pane shows the R script editor with the following code:

```
1 library(ggplot2)
2
3 view(diamonds)
4 summary(diamonds)
5
6 summary(diamonds$price)
7 aveSize <- round(mean(diamonds$carat), 4)
8 cla
9
10 p <-
11
12 xlab="carat", ylab="Price",
13 main="Diamond Pricing")
14
```
- 2- R Console:** The bottom-left pane shows the R console output:

```
Min.   x: 0.000   Min.   y: 0.000   Min.   z: 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean    :      Mean    :      Mean    :
3rd Qu.:      3rd Qu.:      3rd Qu.:
Max.    :      Max.    :      Max.    :
> sum
Min.   Max.
326   950 2401 3933 5324 18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+           data=diamonds, color=clarity,
+           xlab="Carat", ylab="Price",
+           main="Diamond Pricing")
>
> format.plot(plot=p, size=23)
> |
```
- 3- Workspace and History:** The top-right pane shows the Workspace and History sections. The Data section lists 'diamonds' with 53940 observations and 10 variables. The Values section shows 'aveSize' with a value of 0.7979.
- 4- Plots and files:** The bottom-right pane shows a scatter plot titled 'Diamond Pricing'. The x-axis is 'Carat' (ranging from 1 to 3) and the y-axis is 'Price' (ranging from 5000 to 10000). The plot is faceted by clarity (VS2, VS1, VVS2, VVS1, IF) and shows a positive correlation between carat weight and price.

Lets get started...

Do it Yourself!

The image shows a screenshot of the RStudio interface. The top-left pane is the Code Editor, showing R code for loading ggplot2, viewing the diamonds dataset, and creating a plot. The top-right pane is the Workspace and History pane, showing the 'diamonds' dataset with 53940 observations. The bottom-left pane is the R Console, showing the output of the code. The bottom-right pane is the Plots pane, showing a scatter plot titled 'Diamond Pricing' with 'Carat' on the x-axis and 'Price' on the y-axis, colored by clarity.

1- Code Editor

```
1 library(ggplot2)
2
3 view(diamonds)
4 summary(diamonds)
5
6 summary(diamonds$price)
7 aveSize <- round(mean(diamonds$carat), 4)
8 cla
9
10 p <-
11
12 xlab="carat", ylab="Price",
13 main="Diamond Pricing")
14
```

2- R Console

```
> summary(diamonds)
  Min.   x: 0.000   y: 0.000   z: 0.000
  1st Qu.: 4.710  1st Qu.: 4.720  1st Qu.: 2.910
  Median : 5.700  Median : 5.710  Median : 3.530
  Mean   : 3.539
  3rd Qu.: 4.040
  Max.   : 1.800
> summary(diamonds$price)
  Min.   326   Max. 18820
  1st Qu. 950   2nd Qu. 2401   3rd Qu. 3933   Max. 5324
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+           data=diamonds, color=clarity,
+           xlab="Carat", ylab="Price",
+           main="Diamond Pricing")
> format.plot(plot=p, size=23)
```

3- Workspace and History

Data
diamonds 53940 obs. of 10 variables

Values
aveSize 0.7979

4 - Plots and files

Diamond Pricing

Price

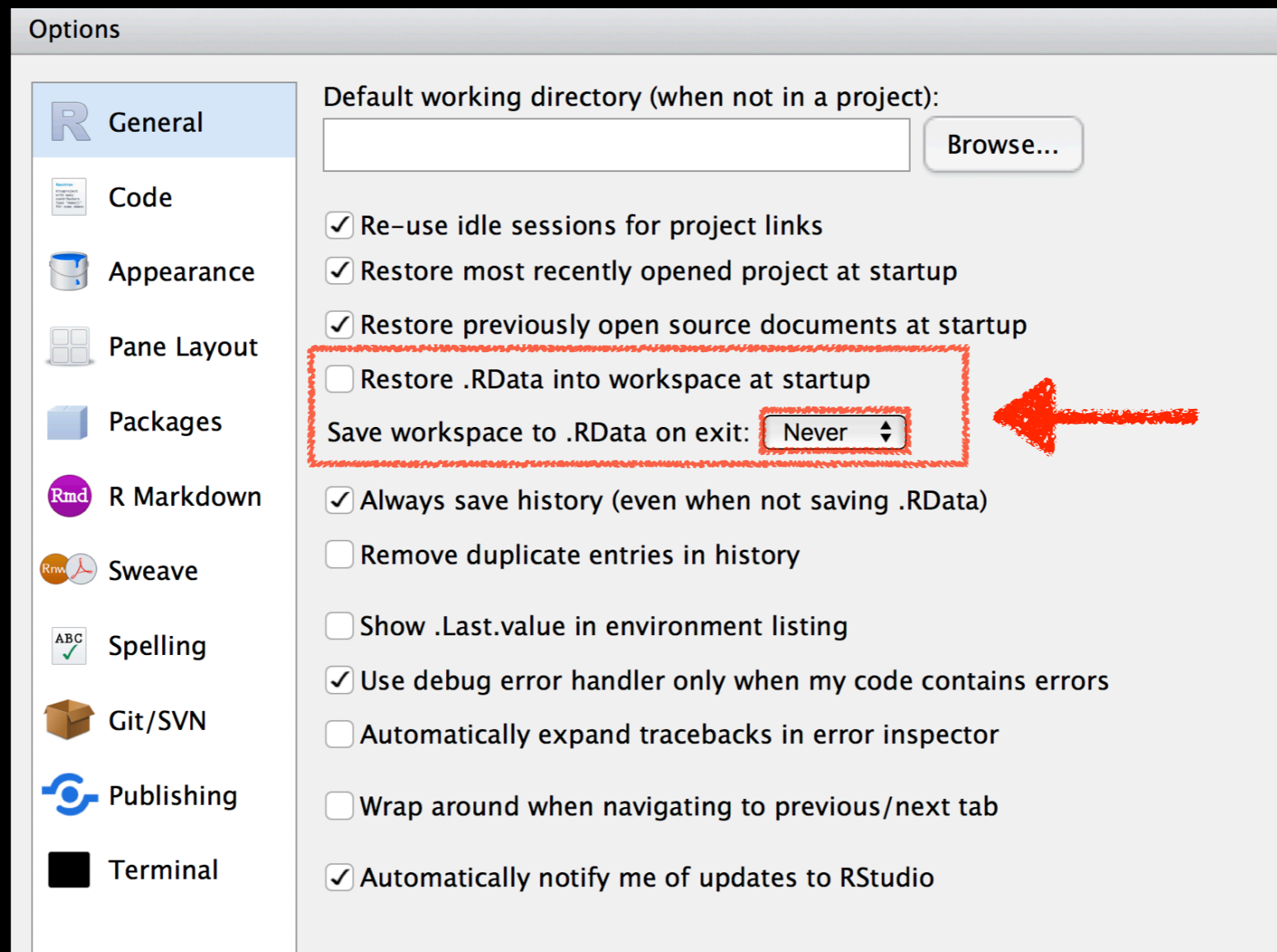
Carat

Legend: VS2, VS1, VVS2, VVS1, IF

Tools > Global Options

Do it Yourself!

- We will **NOT** save our workspace...



Tools > Global Options

- [Optional] Change to dark Appearance and increase font size, etc. ...



Options

General

Code

Appearance

Pane Layout

Packages

R Markdown

Sweave

Spelling

Git/SVN

Publishing

Terminal

RStudio theme: Modern

Zoom: 125%

Editor font: Monaco

Editor font size: 13

Editor theme:

- Ambiance
- Chaos
- Chrome
- Clouds
- Clouds Midnight
- Cobalt
- Crimson Editor
- Dawn
- Dracula
- Dreamweaver
- Eclipse
- Idle Fingers
- Katzenmilch
- Kr Theme
- Material
- Merbivore

Add... Remove

```
# plotting of R objects
plot <- function (x, y, ...)
{
  if (is.function(x) &&
      is.null(attr(x, "class")))
  {
    if (missing(y))
      y <- NULL

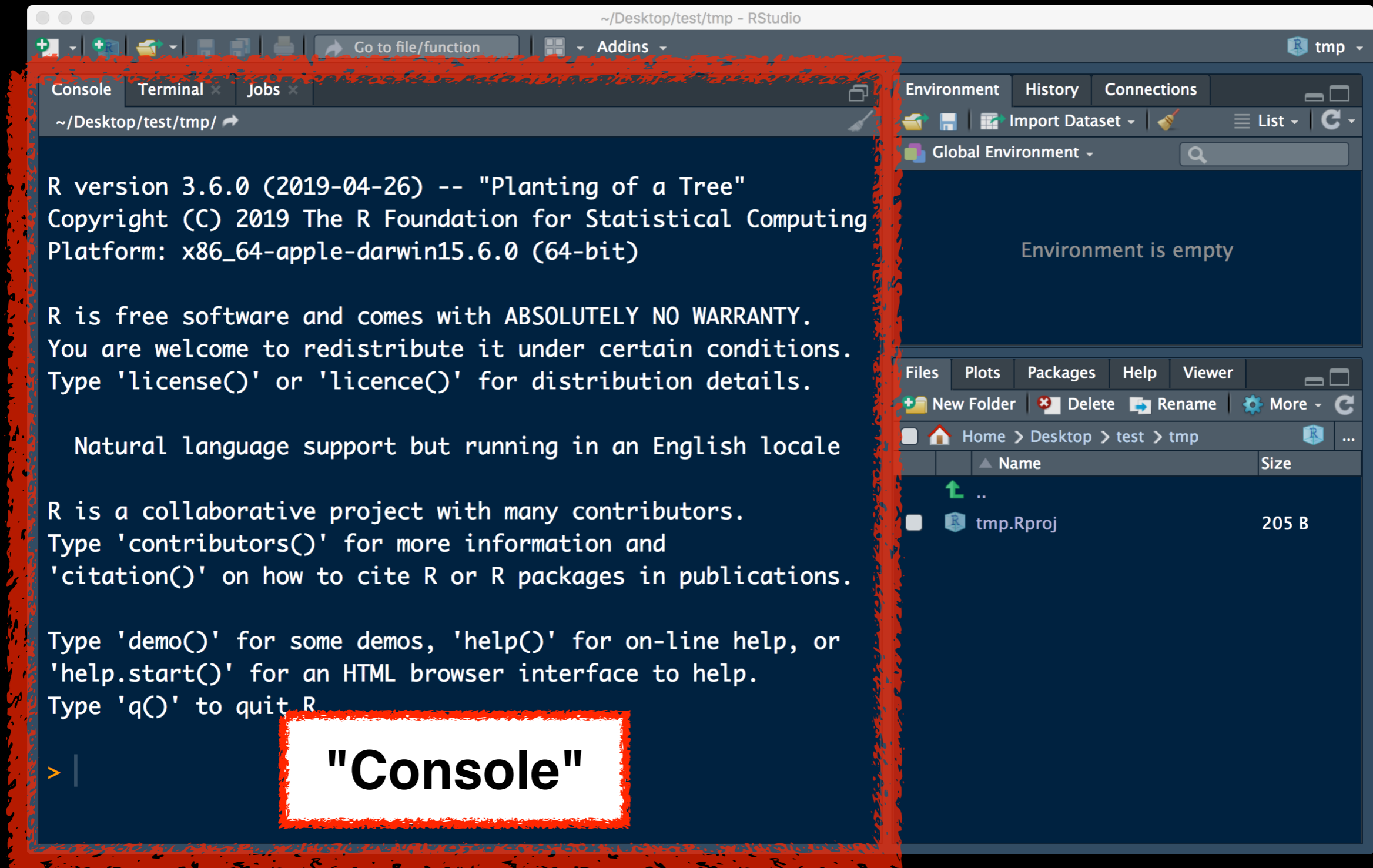
    # check for ylab argument
    hasylab <- function(...)
      !all(is.na(
        pmatch(names(list(...)),
                 "ylab")))

    if (hasylab(...))
      plot.function(x, y, ...)

    else
      plot.function(
        x, y,
        ylab = paste(
          deparse(substitute(x)),
          "(x)",
```

Lets get started...

Do it Yourself!



The image shows a screenshot of the RStudio interface. The console pane on the left displays the R startup message and a list of helpful commands. The environment pane on the right shows an empty global environment. The file browser at the bottom shows the current project directory.

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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

> |
```

Environment is empty

Name	Size
..	
tmp.Rproj	205 B

"Console"

Some simple R commands

Do it Yourself!

R prompt, don't type this!

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`

Order of precedence

`[1] 3`

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)`

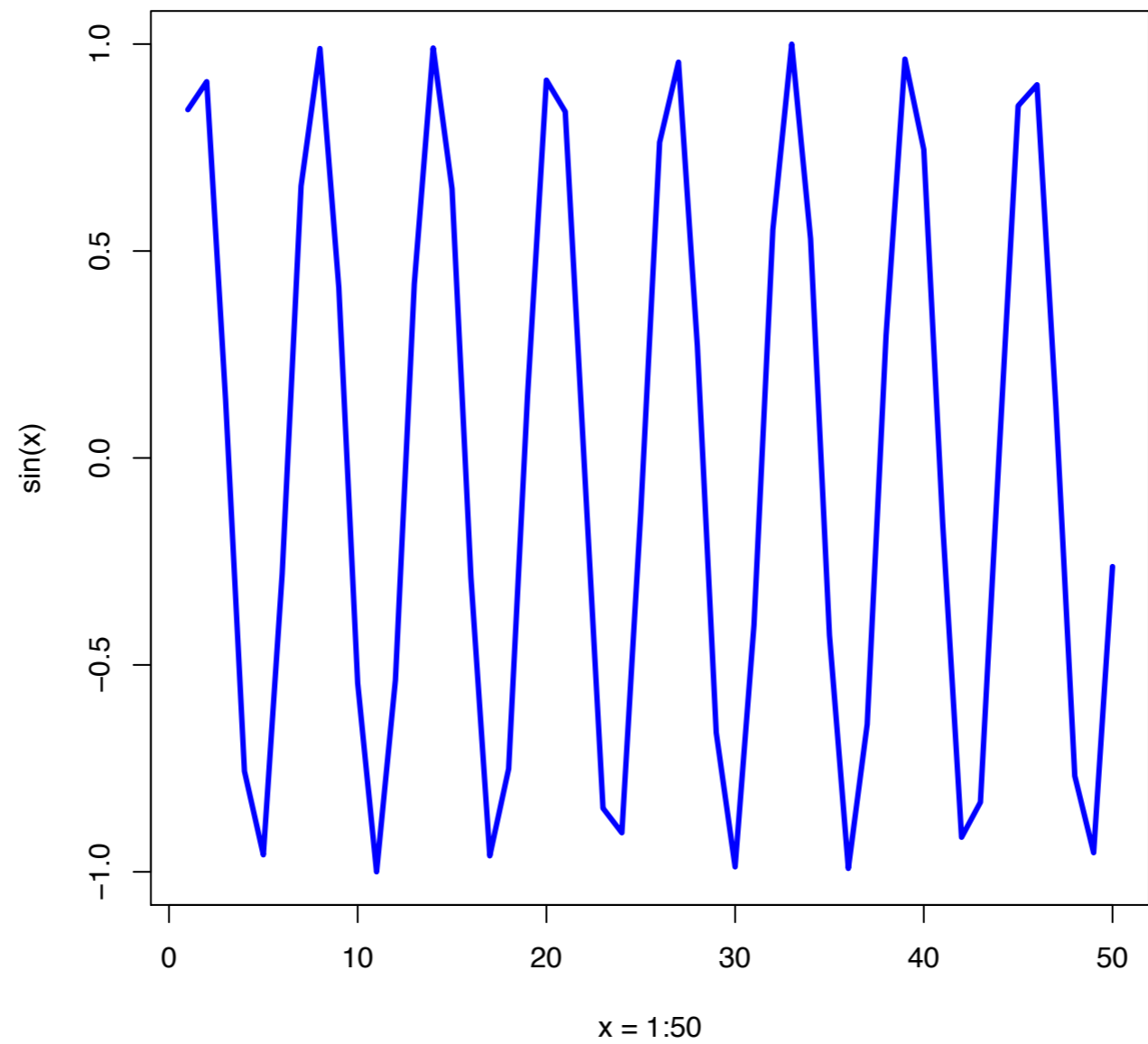
`[1] 1`

Incomplete command

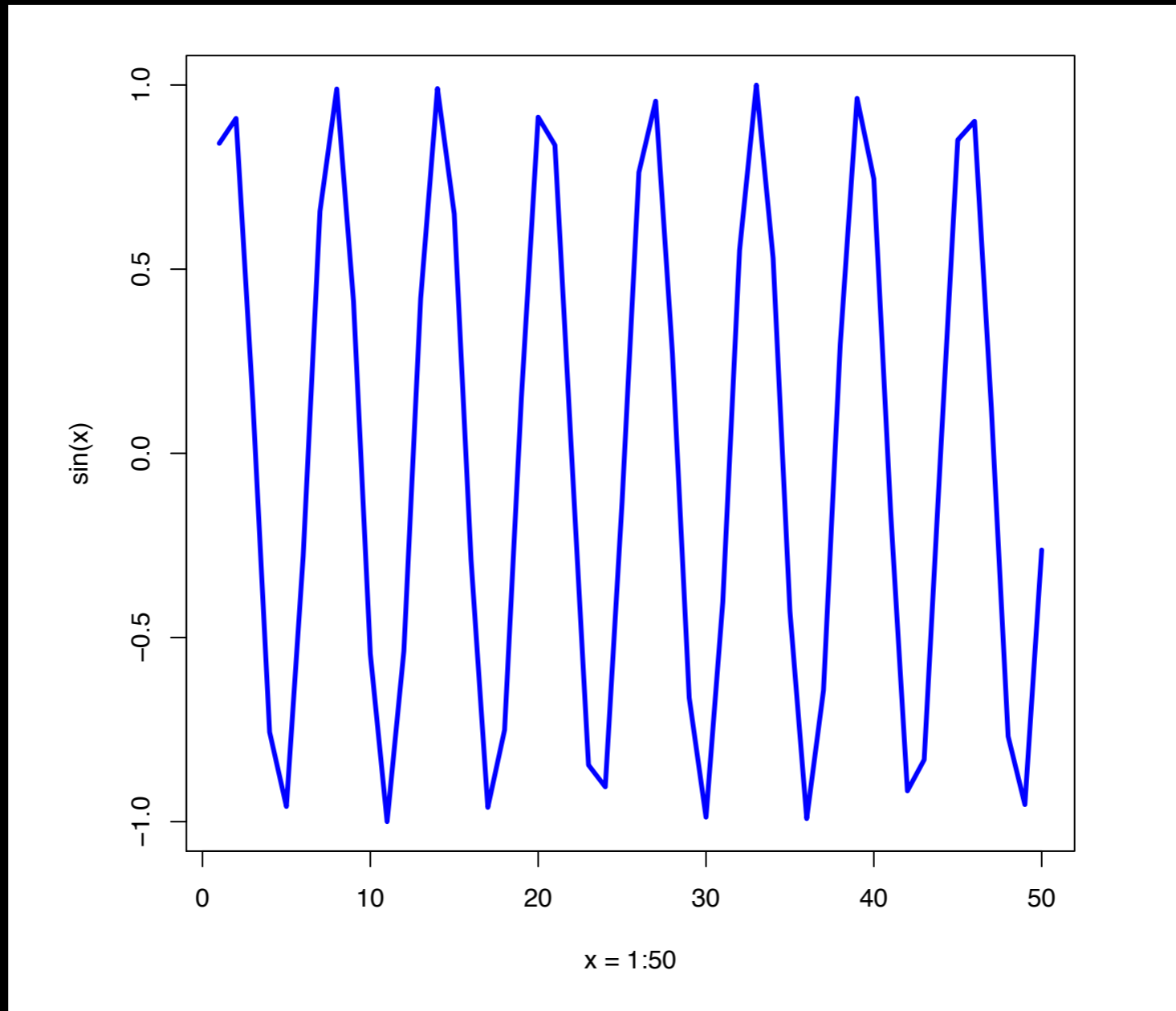
10 `> x=1:50`

`> plot(x, sin(x))`

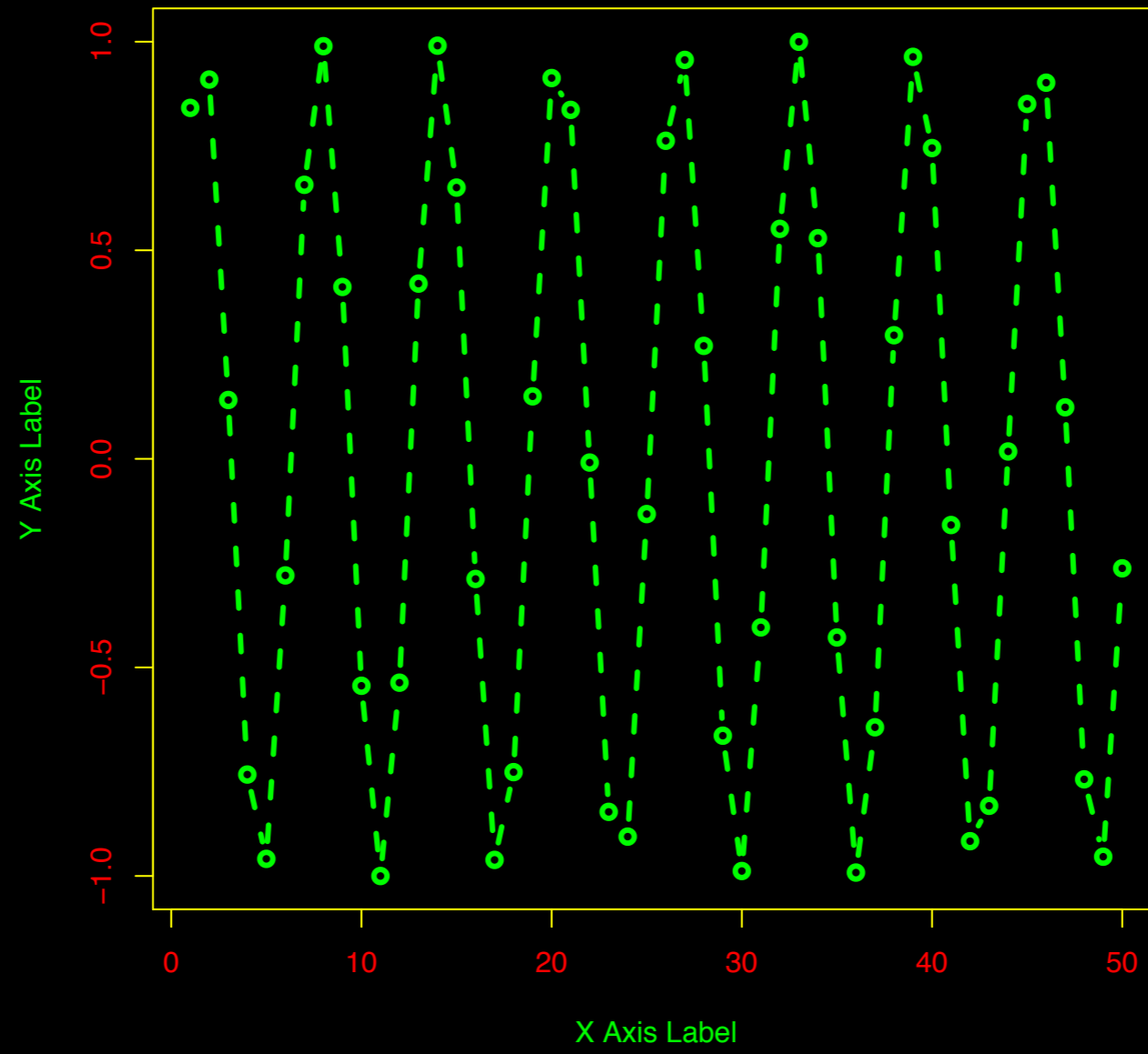
Does your plot look like this?

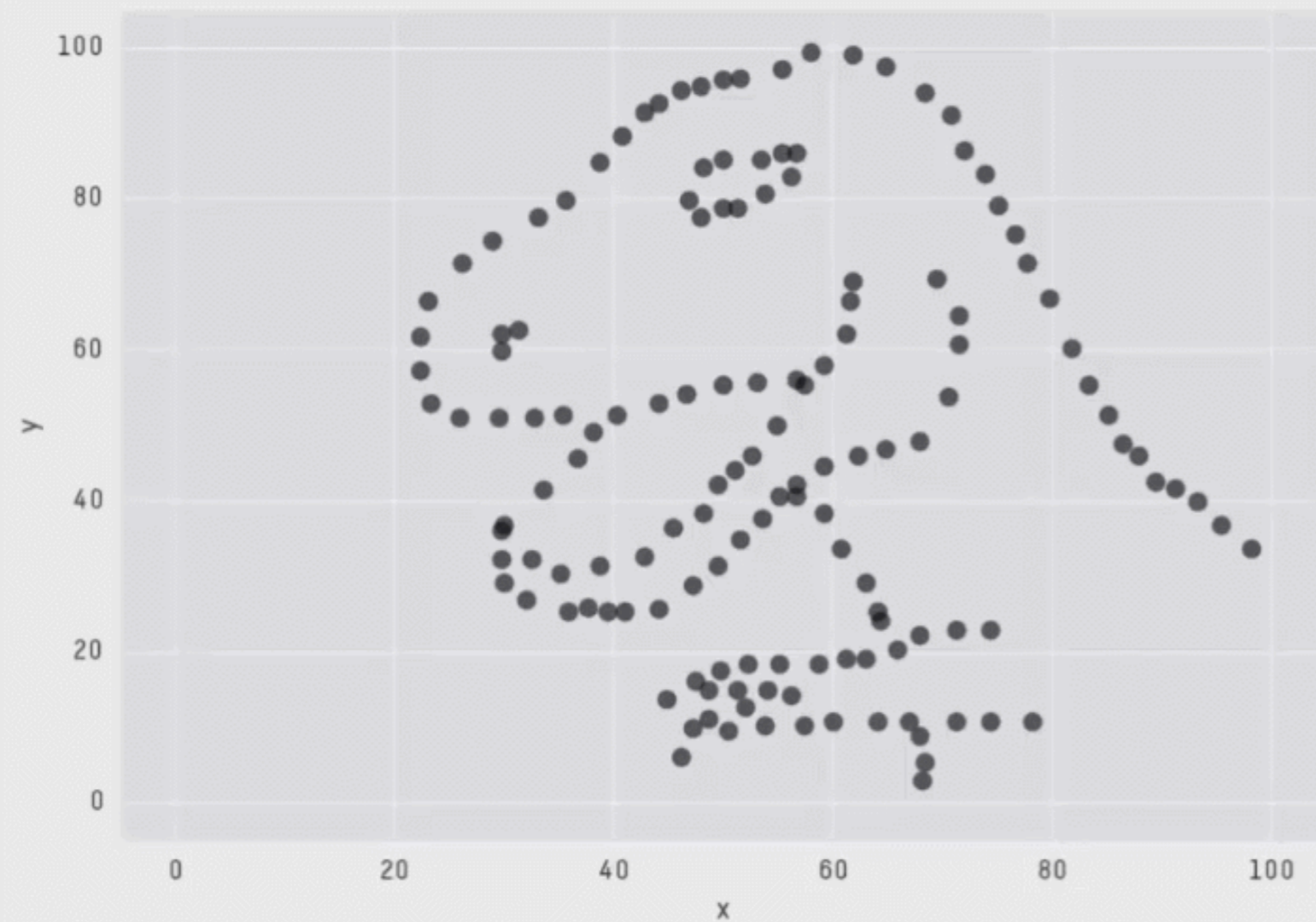


```
plot(x, sin(x), typ="l", col="blue", lwd=3, xlab="x = 1:50")
```



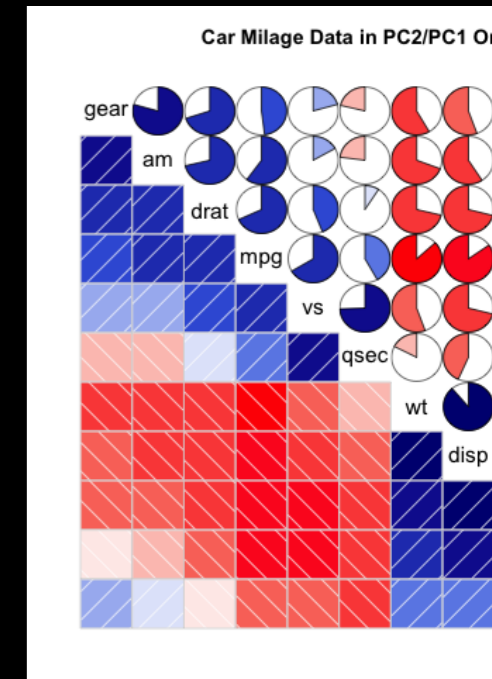
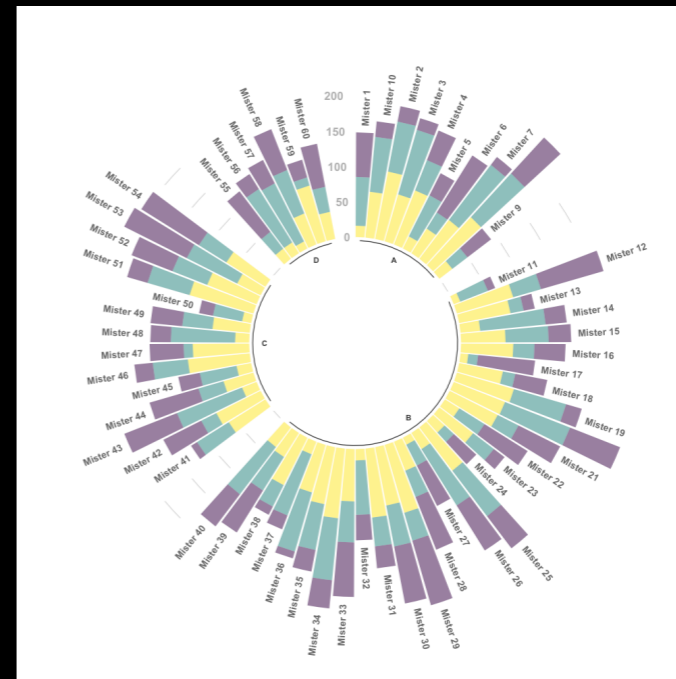
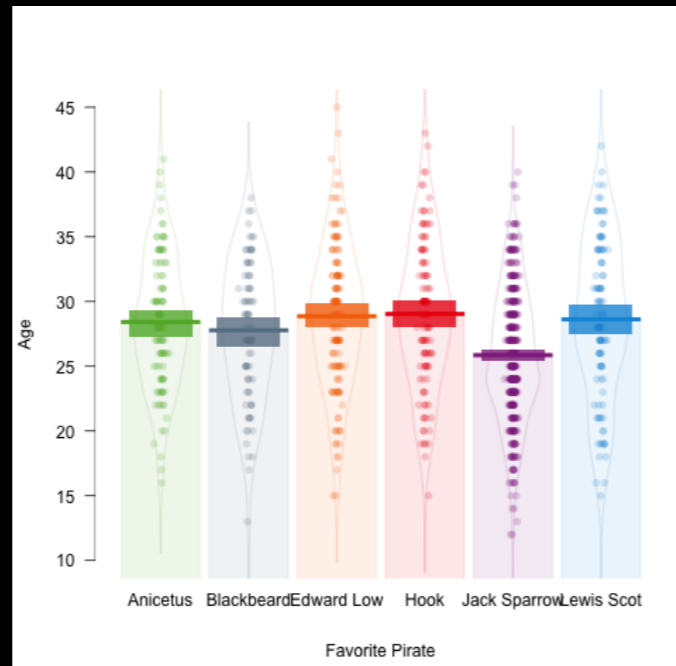
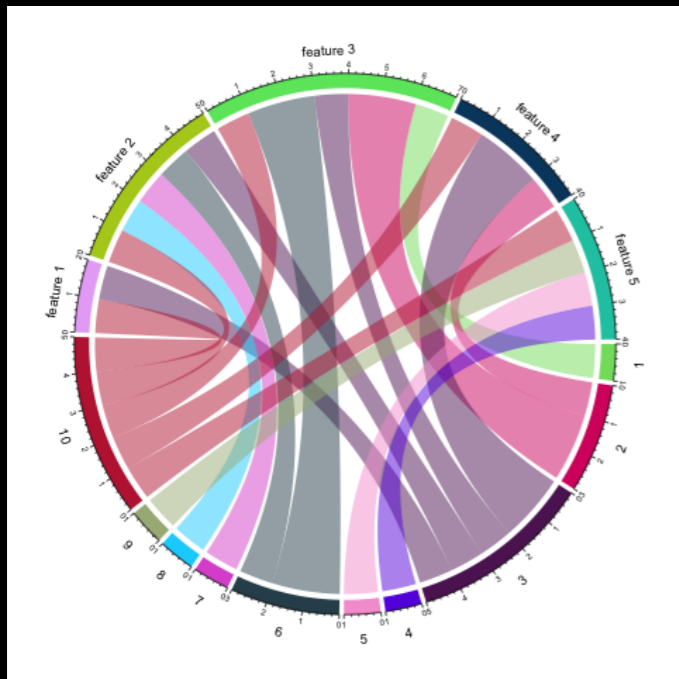
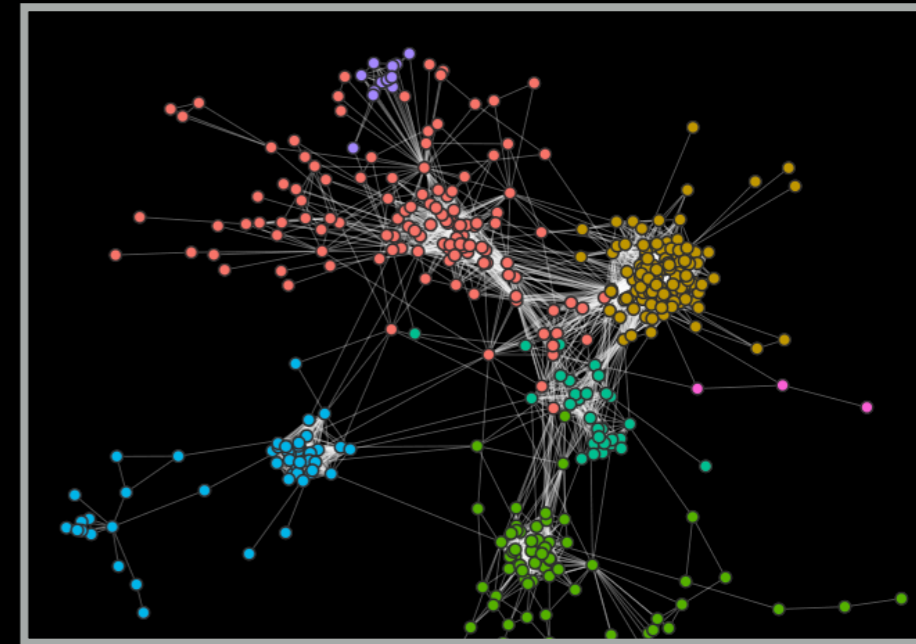
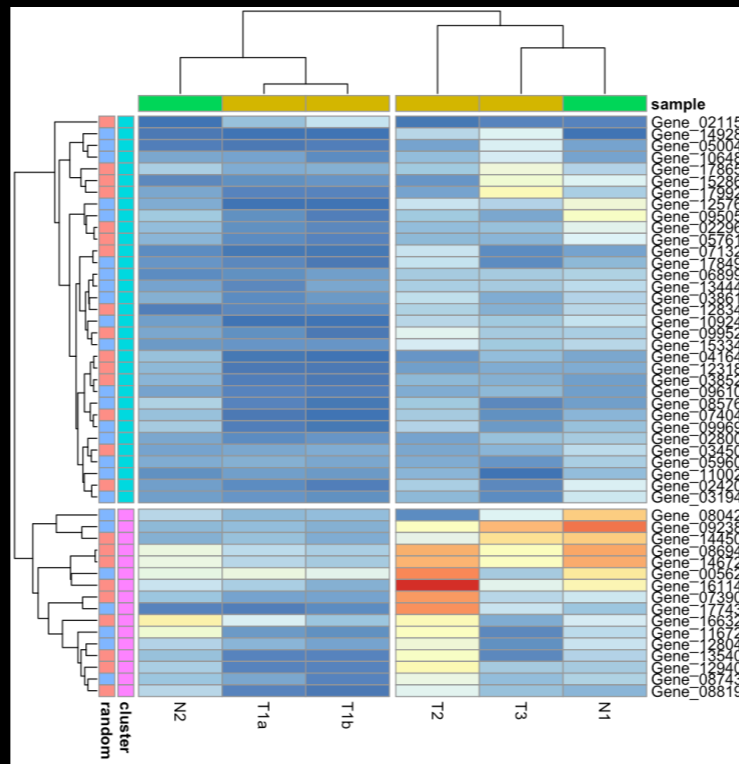
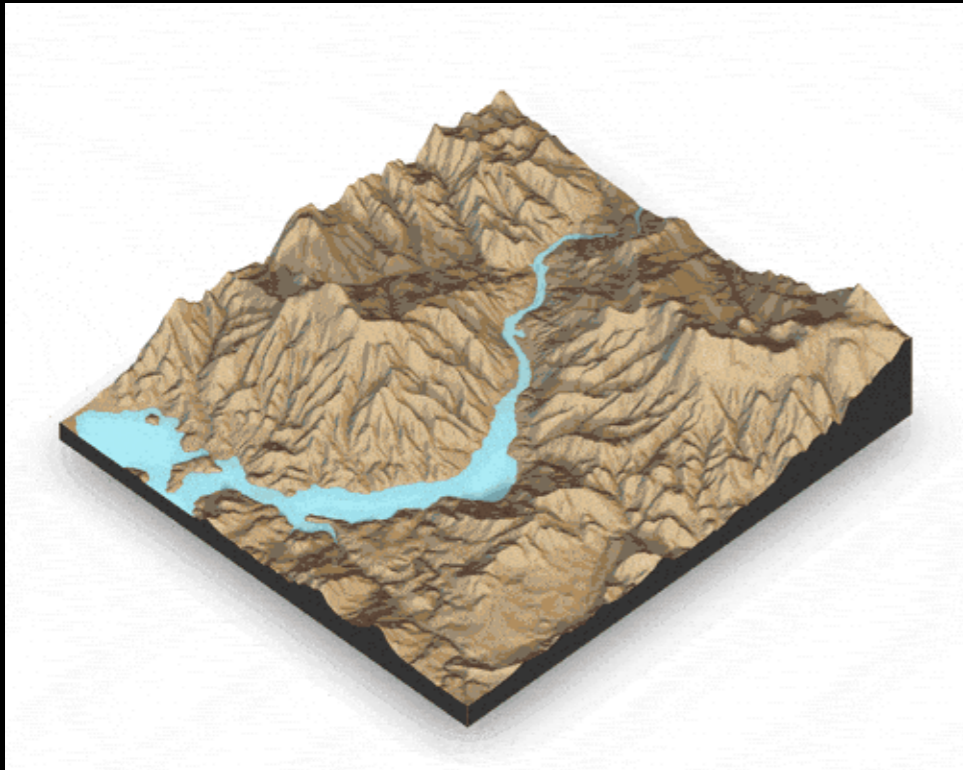
Options: `?plot` `?plot.default`





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!





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!

https://bioboot.github.io/bgggn213_W20/class-material/lab-4-bgggn213/

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

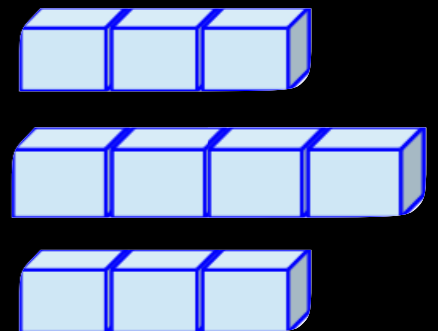
Workspace and working directory

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 a subset of vector elements:

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



Why is this useful?

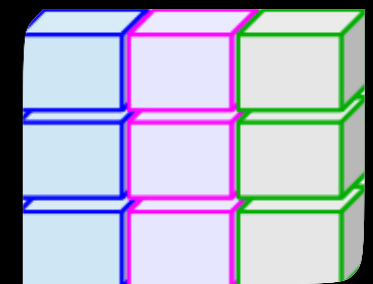
- Because if you know the name (i.e. your label) then you don't have to remember which element of a vector the data you are after was stored in. Consider this *fictional* example:

```
> 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
```

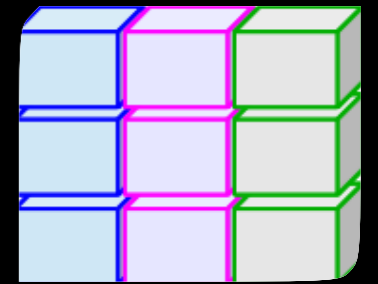
R has many data structures

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")
```

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**.

Topics Covered:

Calling Functions

Getting help in R

Vectors and vectorization

Workspace and working directory

RStudio projects

Topics Covered:

Calling Functions

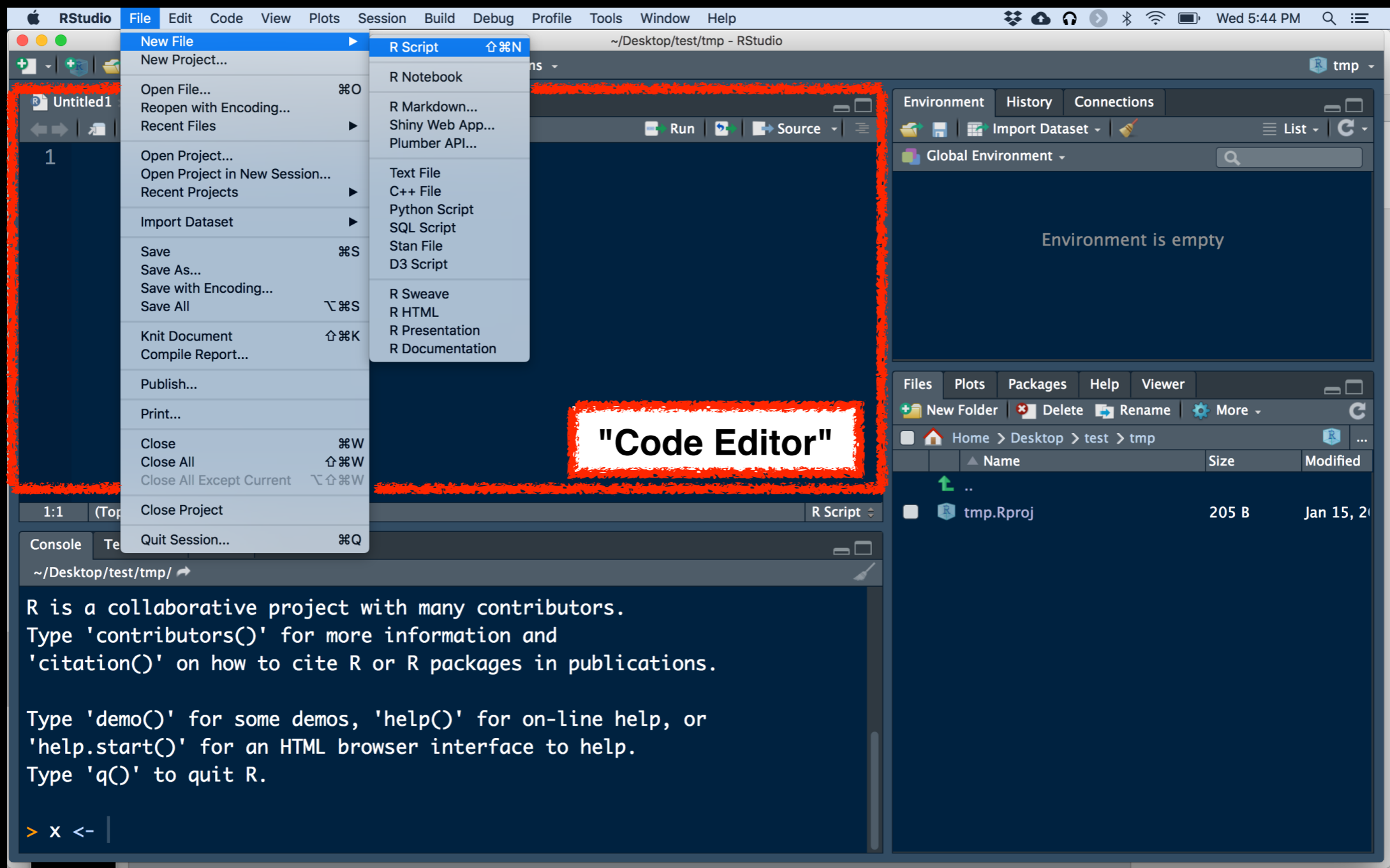
Getting help in R

Vectors and vectorization

Workspace and **working directory**

RStudio **projects**

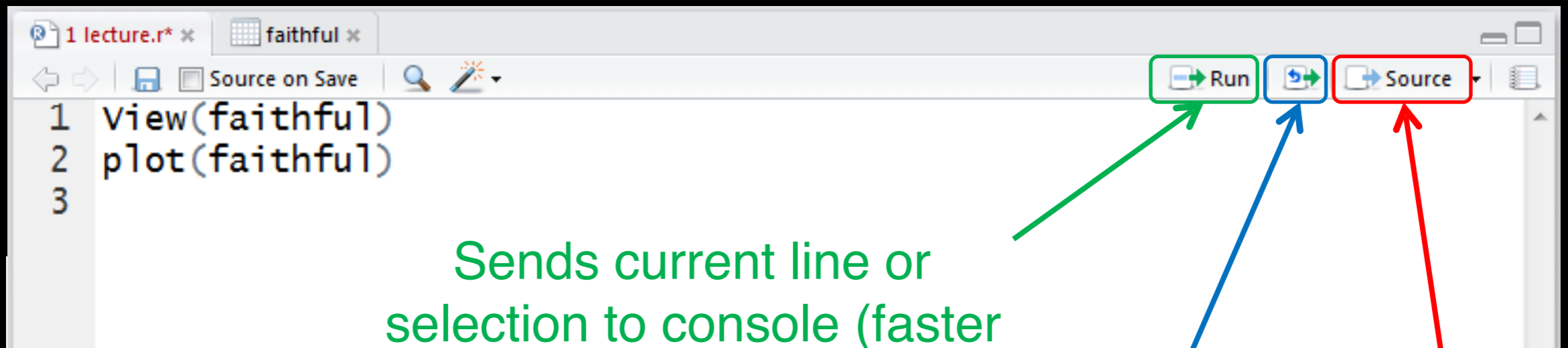
Side-note: Use the code editor for R scripts



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

```
4. sandbox (R)
pico:sandbox> R

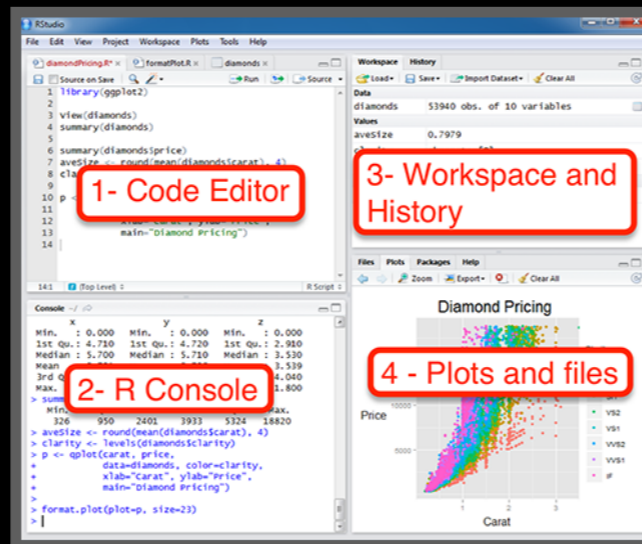
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

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.
>
```



> 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)**

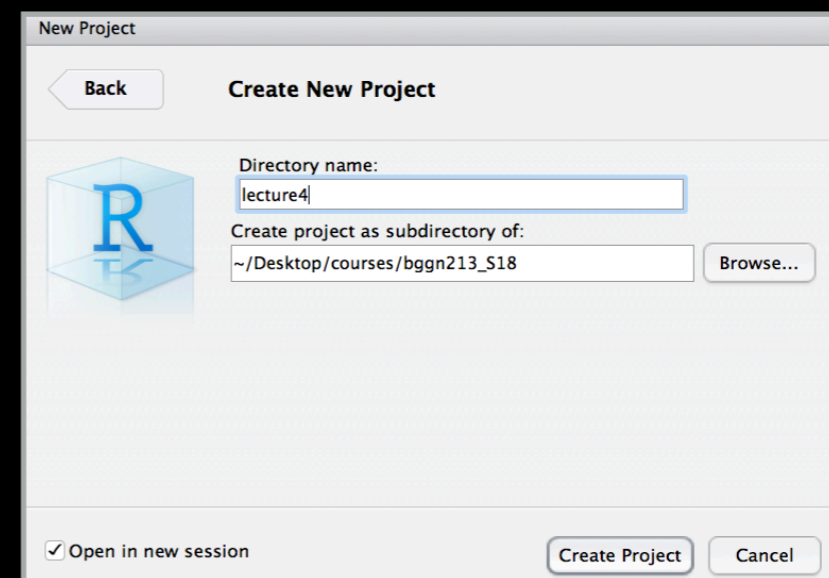
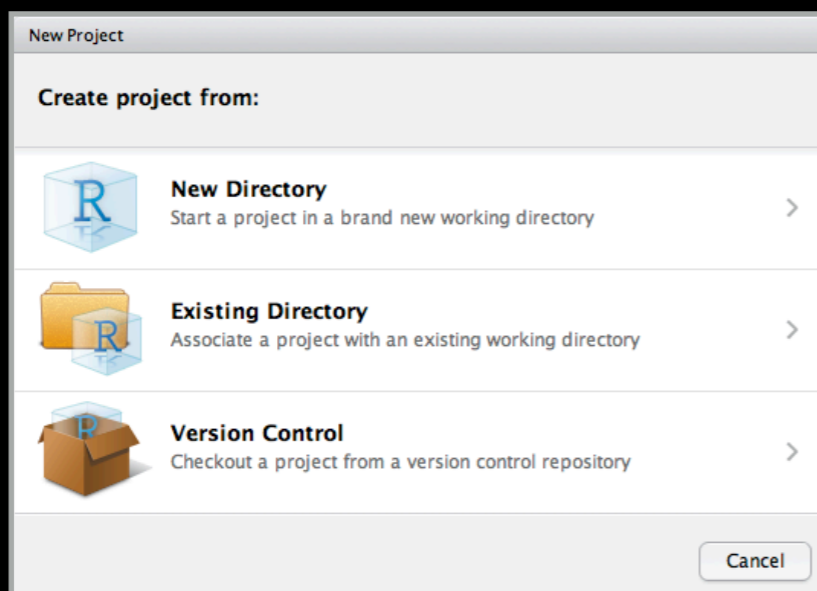
R workspaces

- 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 or use the broom icon
- You should be able to reproduce everything from your R script, so save your R script, don't save your workspace!

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/> >

The screenshot shows the DataCamp website interface. At the top, there is a navigation bar with the DataCamp logo on the left and links for 'Learn', 'Groups', and 'About' in the center. On the right side of the navigation bar, the user's profile is shown with '1,250 XP' and a notification bell icon with a red badge containing the number '3'. A dropdown menu is open from the notification bell, displaying a list of notifications:

- You have a new assignment: Conditionals and Con... 16 days ago
- You have a new assignment: Working with the RSt... 16 days ago
- You have a new assignment: Introduction to R 16 days ago
- bjgrant invited you to the group 'Foundations o... 16 days ago
- You have a new assignment: Orientation 9 months ago

At the bottom of the notification dropdown, there is a button labeled 'See all notifications'. Below the navigation bar, the main content area features a section titled 'Your Latest Activity'. It displays a course card for 'Introduction to Spark in R using...' with a progress indicator (one yellow checkmark and three grey checkmarks) and a message: 'You are doing awesome barryus! So far you've earned 250 XP!'. Below the course card, it states 'The last chapter you were working on was' followed by a link to 'Light My Fire: Starting To Use Spark With dplyr Syntax'. At the bottom of the page, there is a 'DAILY PRACTICE' section with the text: 'Learning data science requires practice **every day**. Build your data science fluency with DataCamp practice mode.'

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.

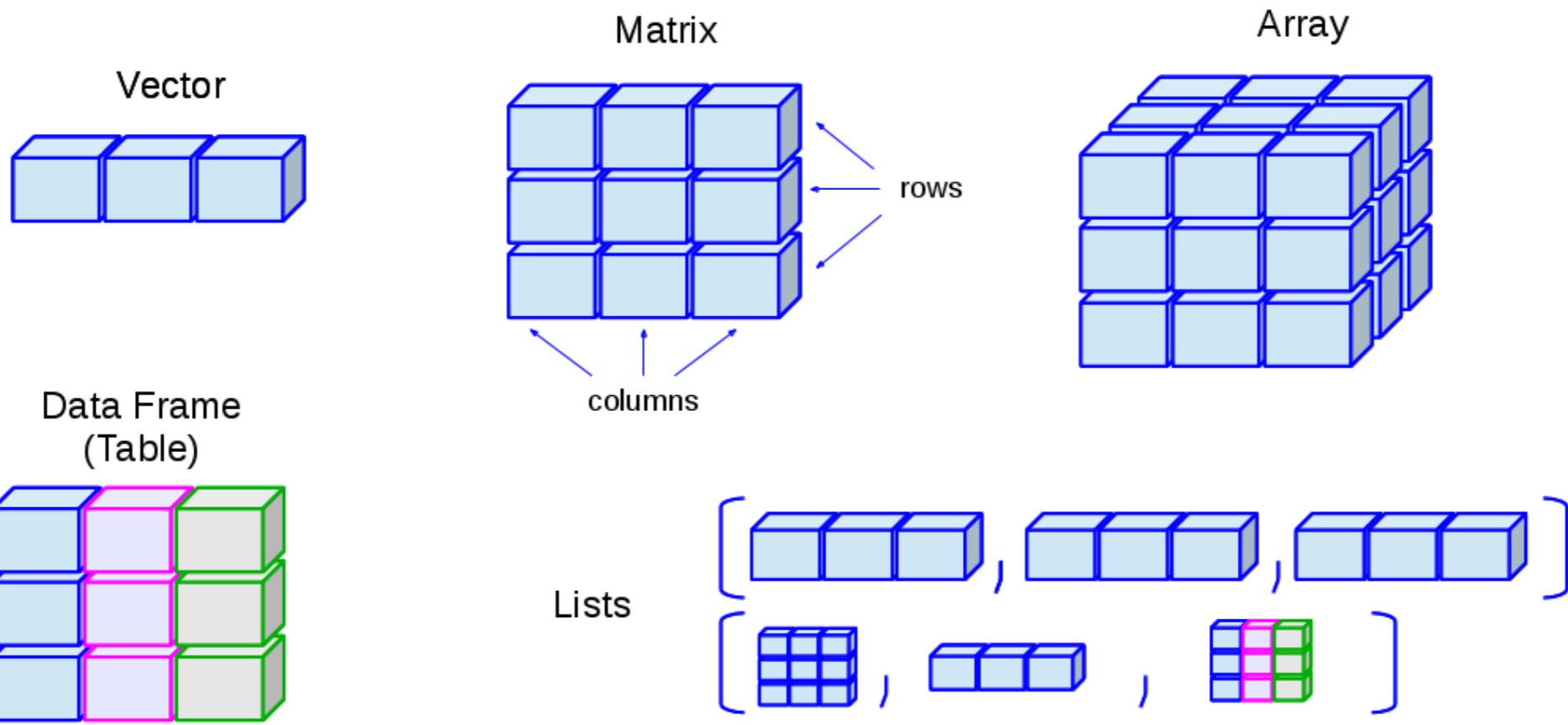
Final Knowledge Check!

- 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.

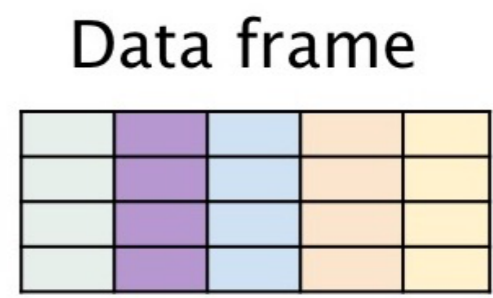
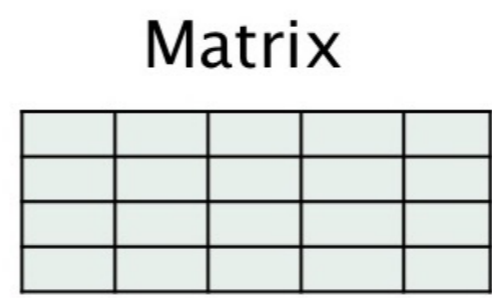
Please complete:

Muddy point assessment

Reference Slides:



Variables	Example
integer	100
numeric	0.05
character	"hello"
logical	TRUE
factor	"Green"



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

R: Logarithms and Exponentials Find in Topic

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 $|x| \ll 1$ (and less accurately when x is approximately -1).

`exp` computes the exponential function.

`expm1(x)` computes $\exp(x) - 1$ accurately also for $|x| \ll 1$.

Usage How to use the function

```
log(x, base = exp(1))
logb(x, base = exp(1))
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 $e = \exp(1)$.

Details

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

`log10` 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).

`logb` 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.

R: Logarithms and Exponentials Find in Topic

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 [Math](#) group generic.

Note that this means that the S4 generic for `log` has 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 `Math` 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 `dlnre1` by W. Fullerton of Los Alamos Scientific Laboratory (see <http://www.netlib.org/slatec/fnlib/dlnrel.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

[Trig](#), [sqrt](#), [Arithmetic](#).

Examples Sample code showing how it works

```
log(exp(3))
log10(1e7) # = 7

x <- 10^-(1+2*1:9)
cbind(x, log(1+x), log1p(x), exp(x)-1, expm1(x))
```

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

Optional Exercise

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

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

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

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

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