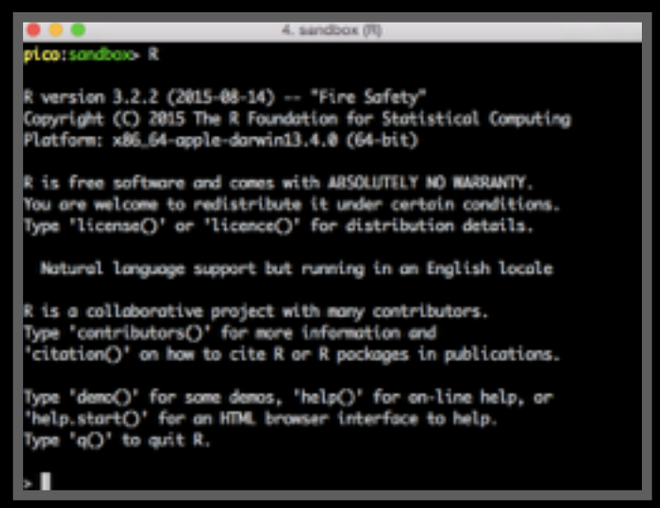
Barry Grant
bjgrant@umich.edu
http://thegrantlab.org

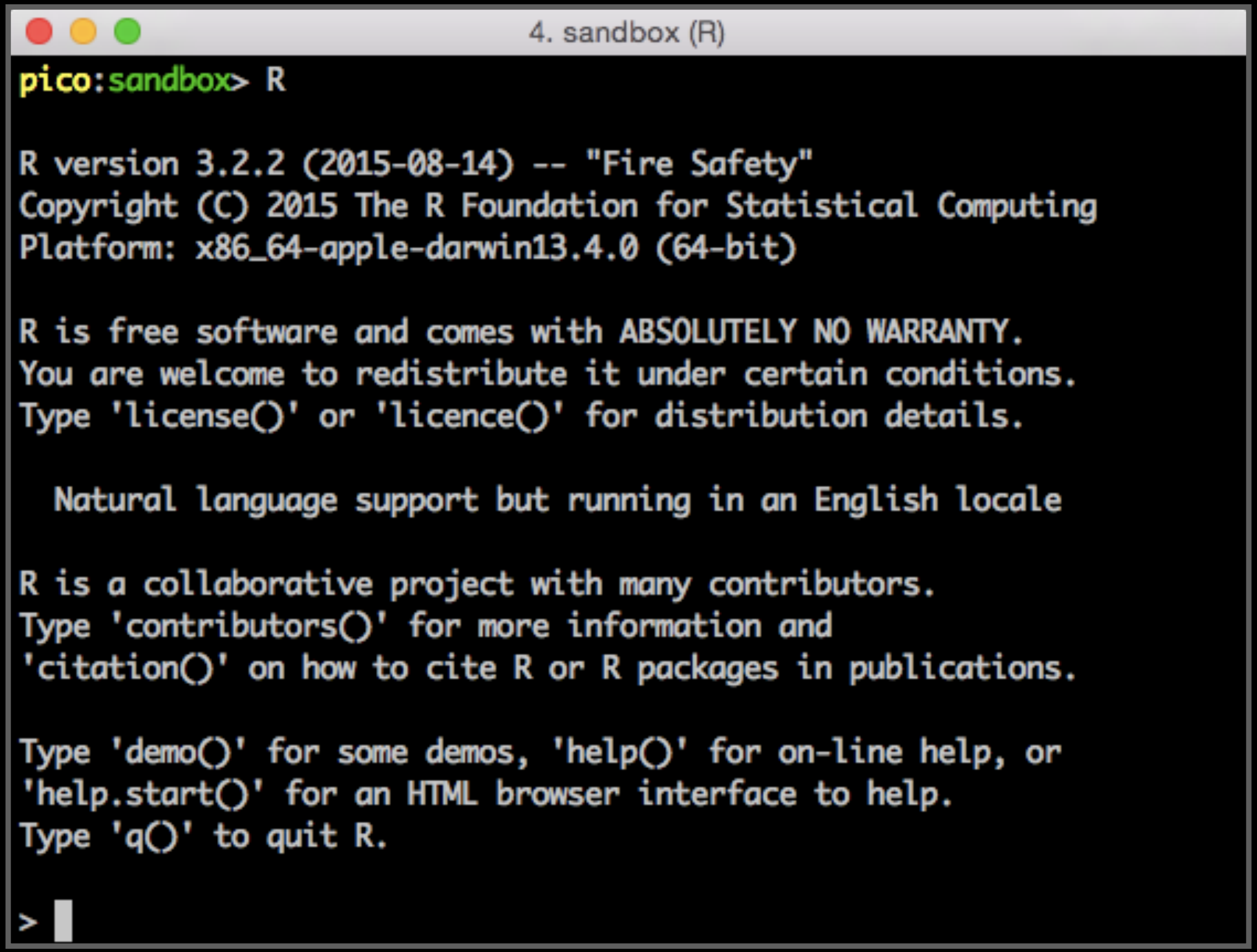
What is R?

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

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: 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

Use the right tool!

- There are many many things Excel just cannot do!

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	FAVORITES
11	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

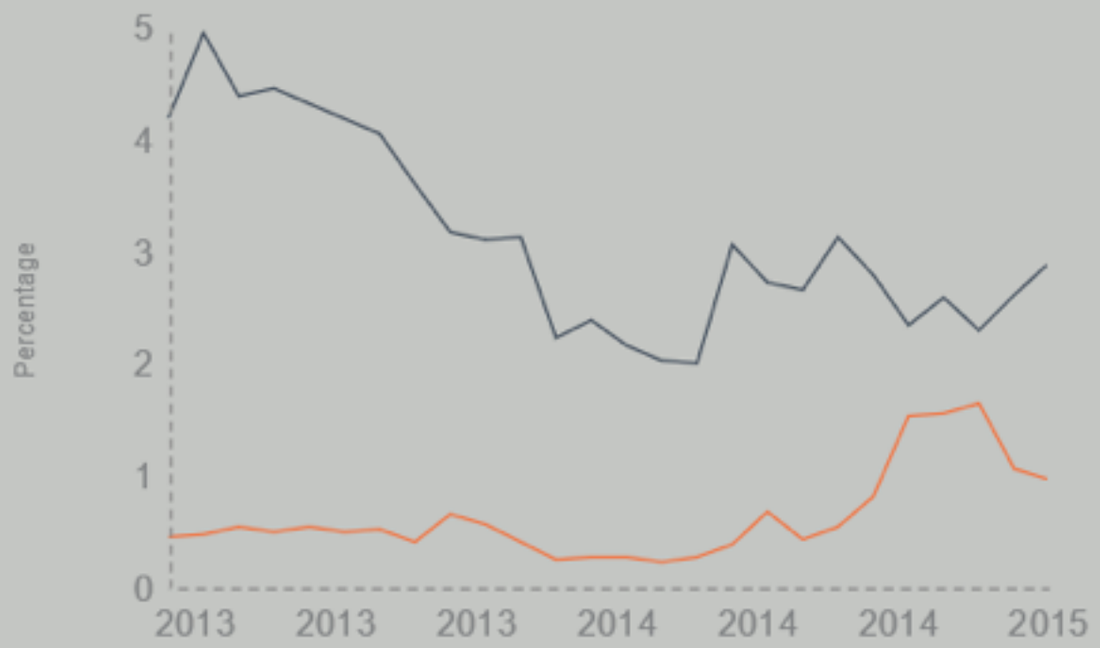
Language Rank	Types	Spectrum Ranking
1. C		100.0
2. Java		98.1
3. Python		98.0
4. C++		95.9
5. R		87.9
6. C#		86.7
7. PHP		82.8
8. JavaScript		82.2
9. Ruby		74.5
10. Go		71.9

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

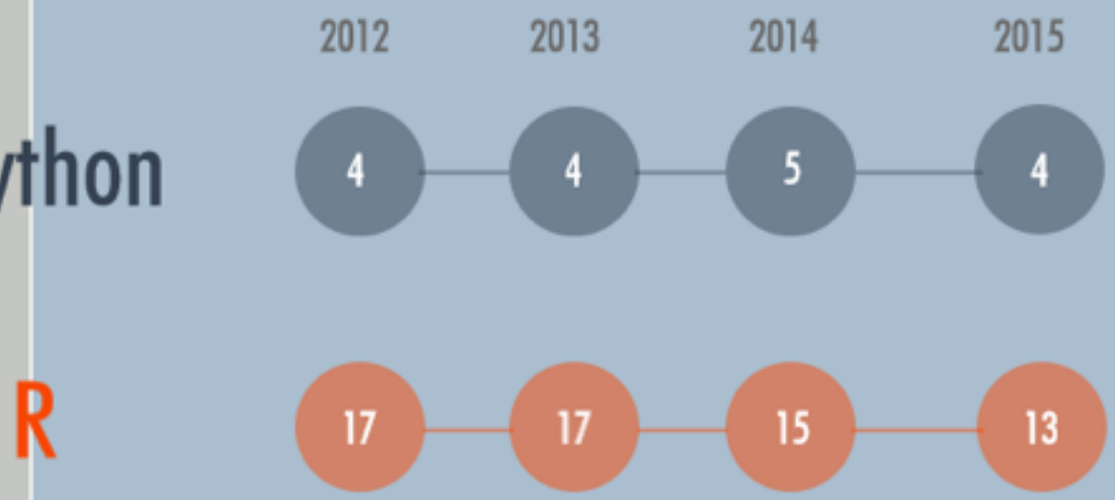
R and Python: The Numbers

Popularity Rankings

R and Pythons 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

- 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.

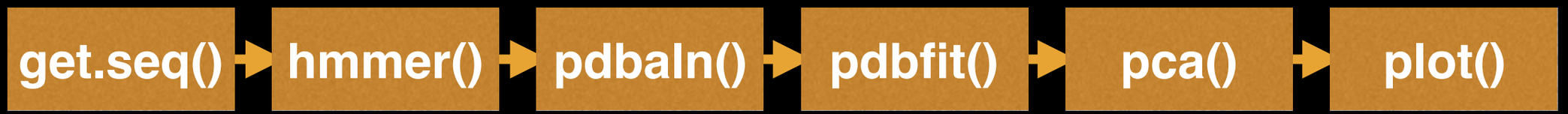
- Unparalleled exploratory data analysis environment.

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.



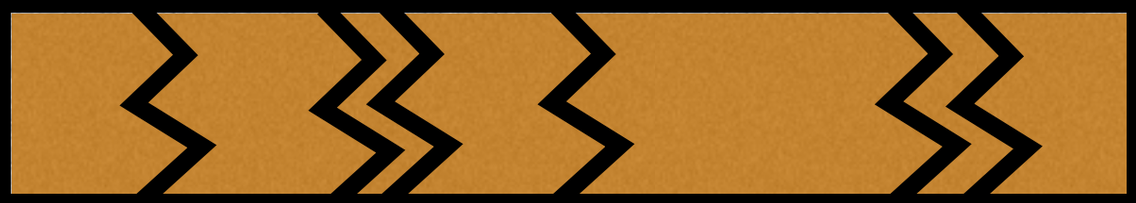
An alternative approach is to write a **single complex program** that takes raw data as input, and after hours of data processing, outputs publication figures and a final table of results.

‘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

1. 

2. 

3. 

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.

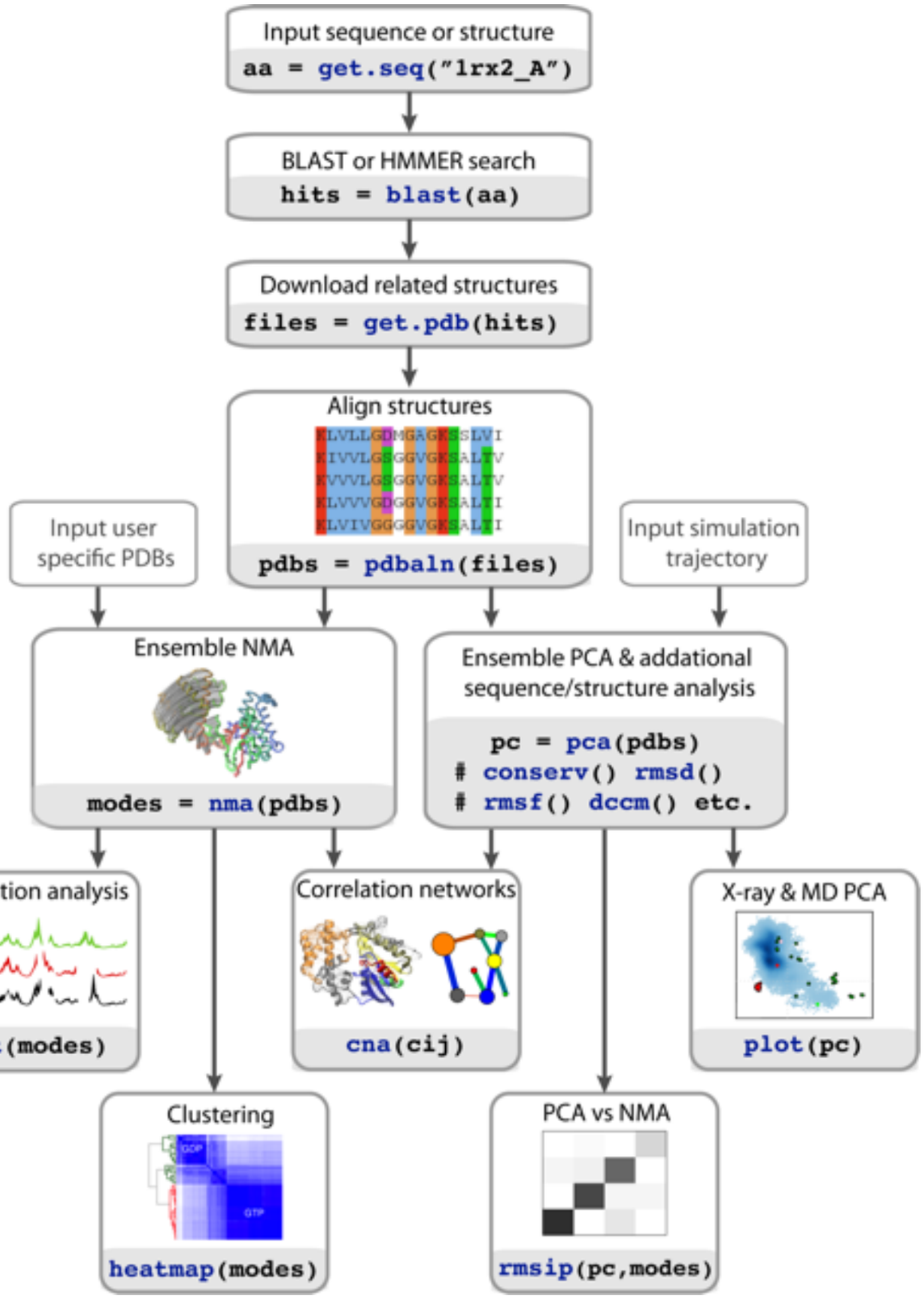
"*Data analysis, like experimentation, must be considered as a highly interactive, iterative process, whose actual steps are selected segments of a stubbily branching, tree-like pattern of possible actions*". [**J. W. Tukey**]

Interactivity & exploratory data analysis

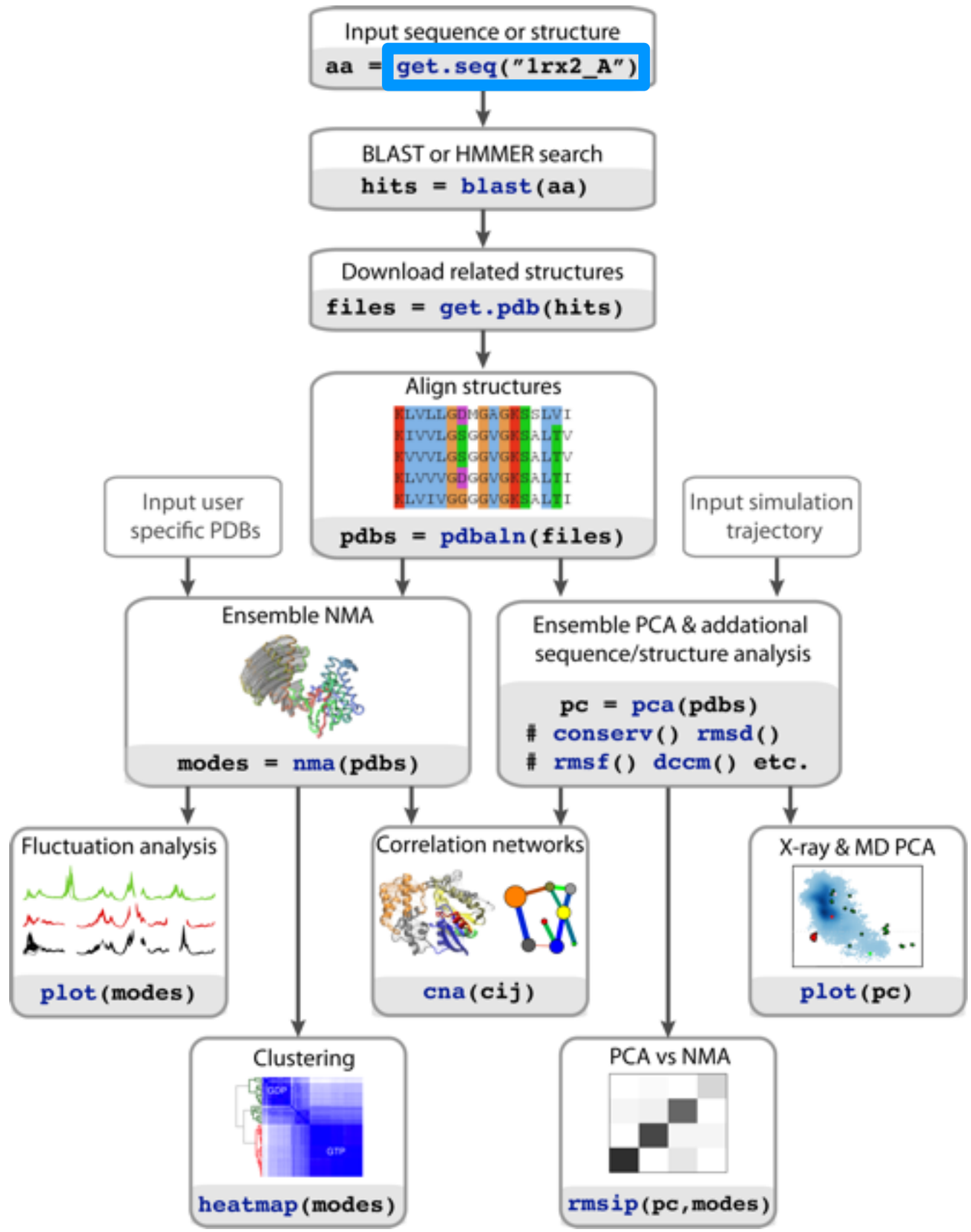
Learning R will give you the freedom to explore and experiment with your data.

"*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.

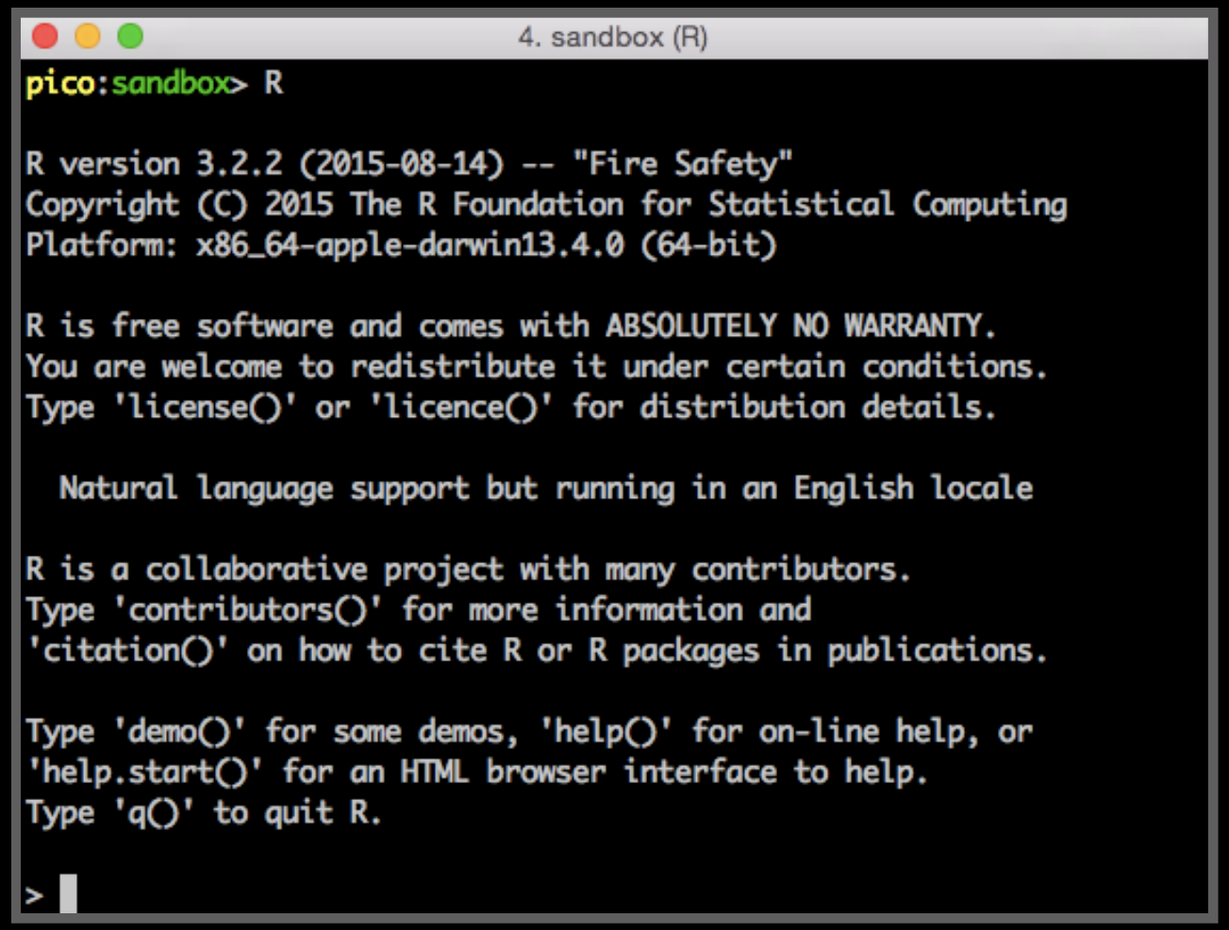

Input sequence or structure
aa = get.seq("1rx2_A")
BLAST or HMMER search
hits = blast(aa)
Download related structures
files = get.pdb(hits)
Align structures
pdbs = pdbaln(files)
Input user specific PDBs
Input simulation trajectory
Ensemble NMA
modes = nma(pdbs)
Ensemble PCA & addational sequence/structure analysis
pc = pca(pdbs)
# conserv() rmsd()
# rmsf() dccm() etc.
Fluctuation analysis
plot(modes)
Correlation networks
cna(cij)
X-ray & MD PCA
plot(pc)
Clustering
heatmap(modes)
PCA vs NMA
rmsip(pc,modes)

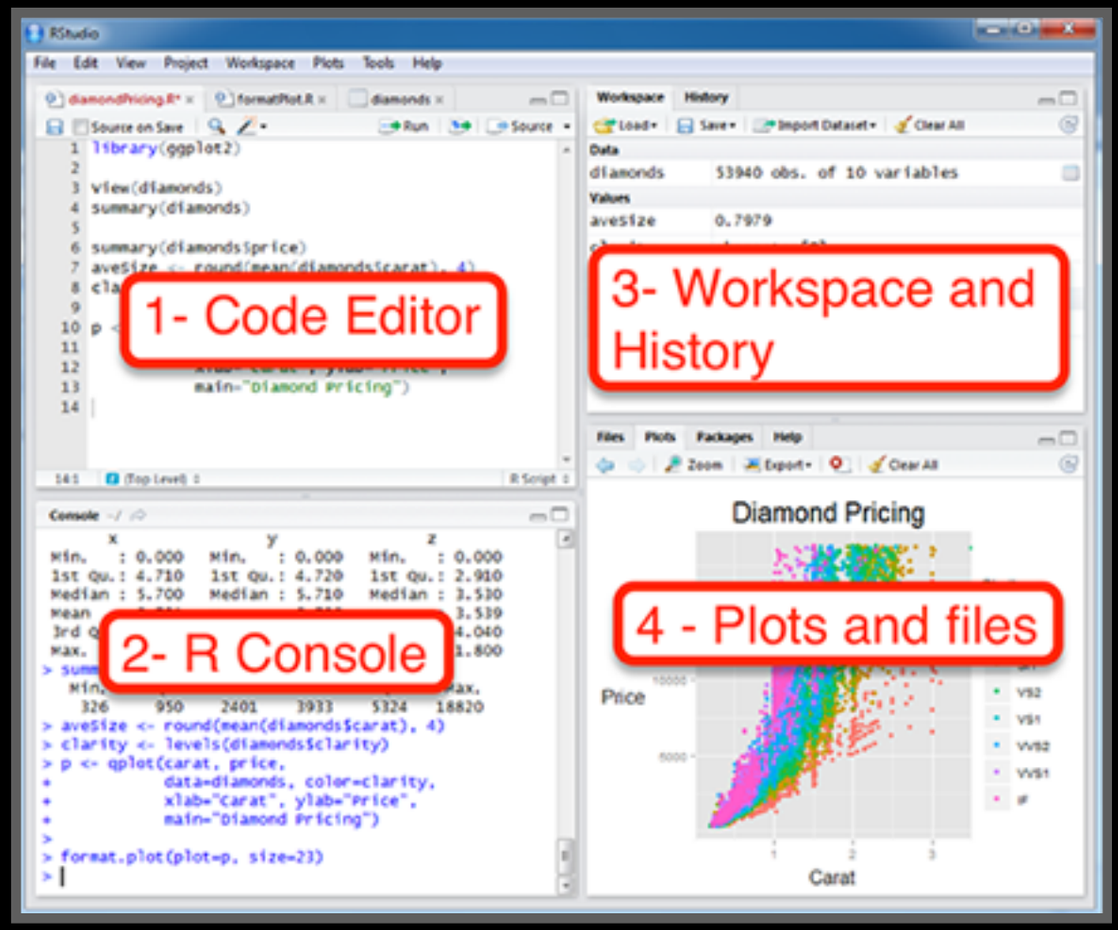
R Features = **functions()**



How do we use R?

Two main ways to use R

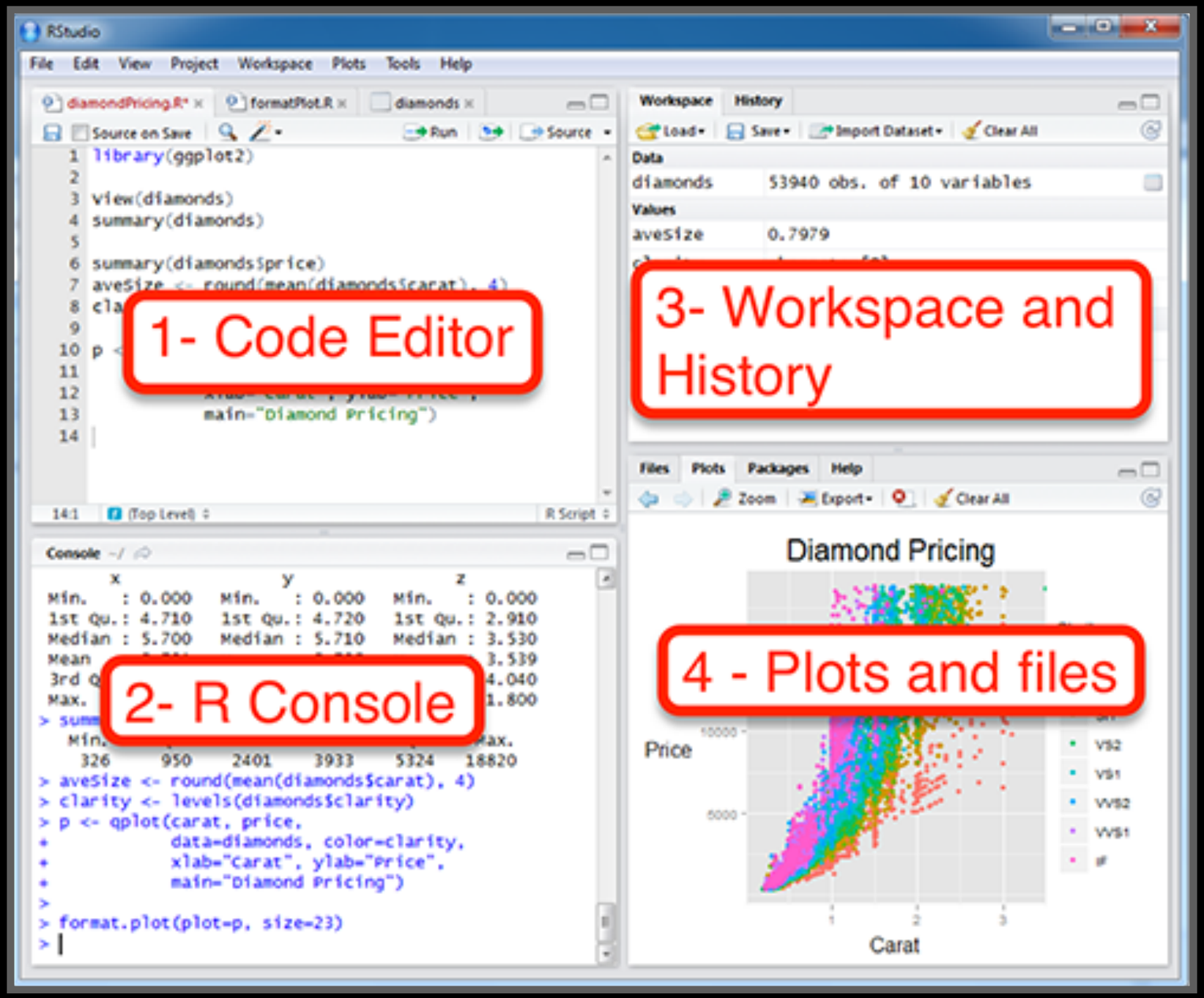




1. Terminal

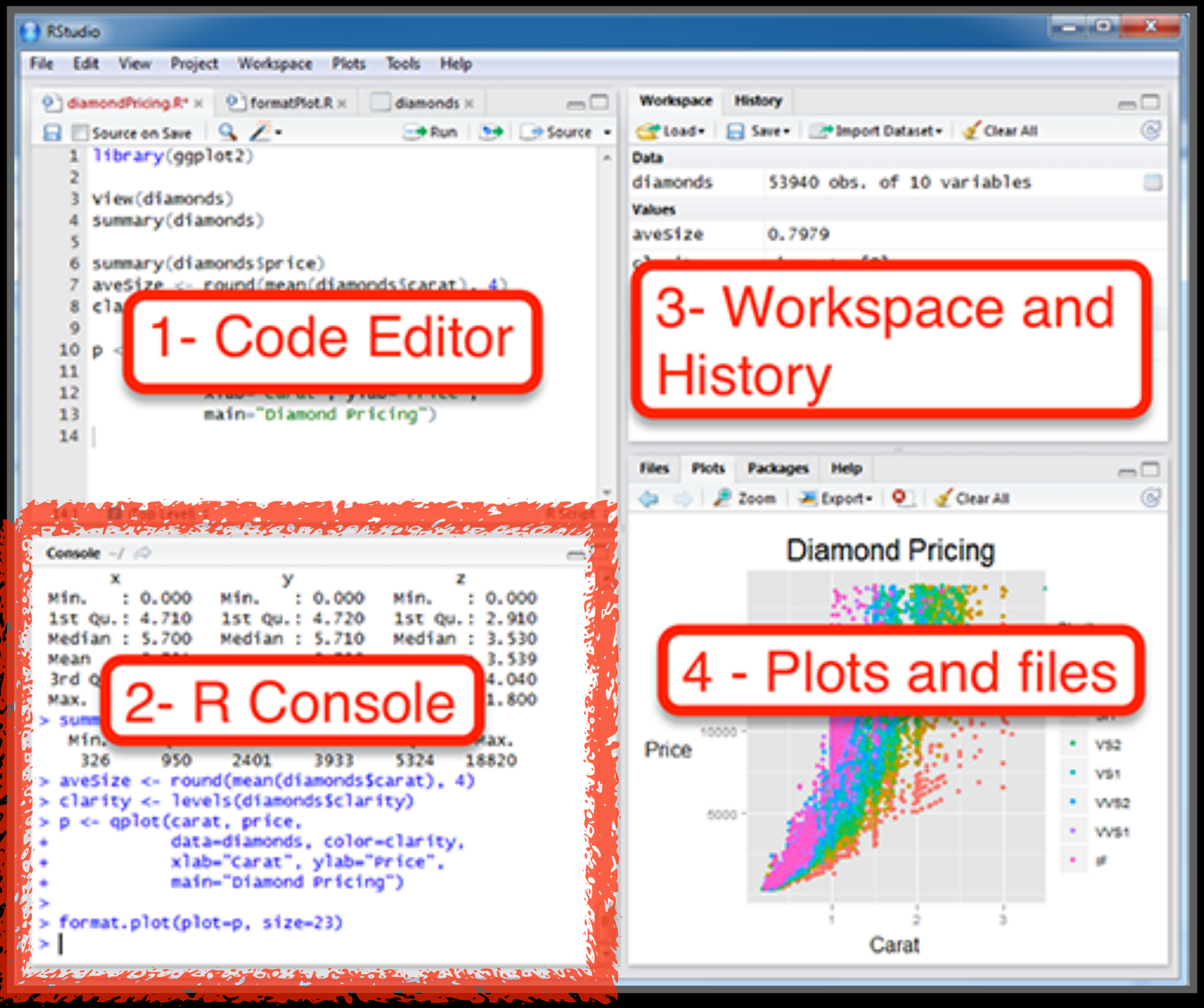
2. RStudio

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))
```


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 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!

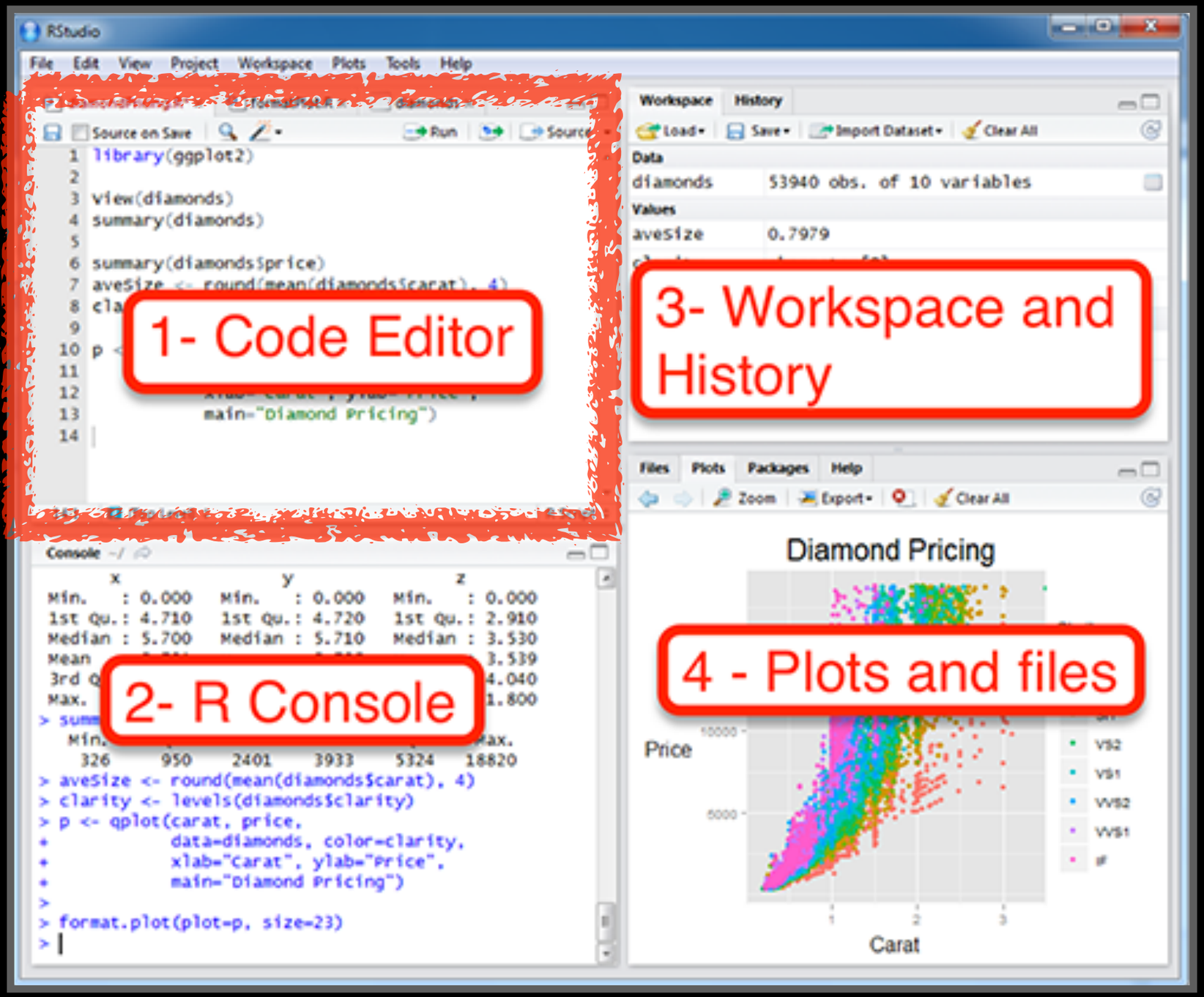
Your turn!

http://tinyurl.com/bioboot-R1

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.* day4.r) that contains your R code for one complete analysis

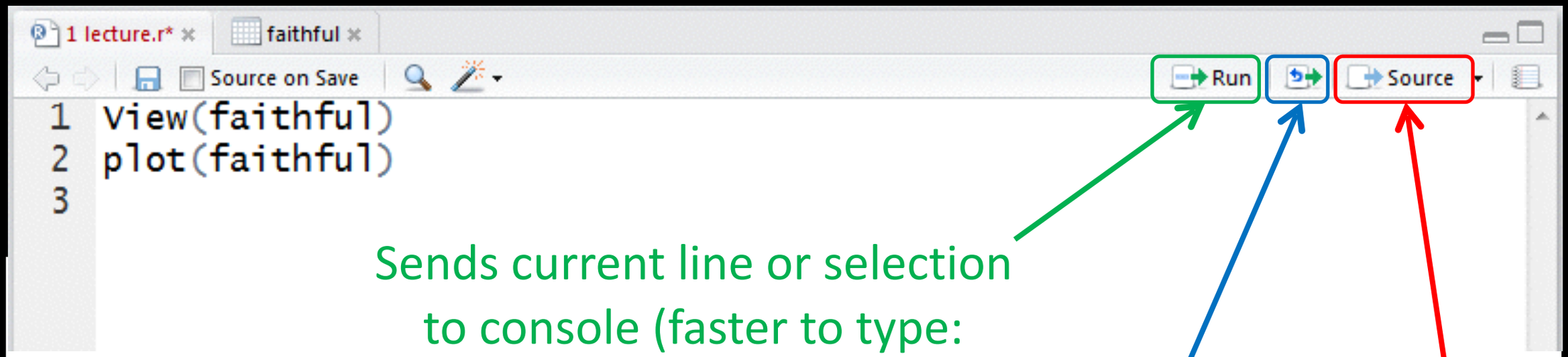
- **Scientific method**: complete record of your analysis

- **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

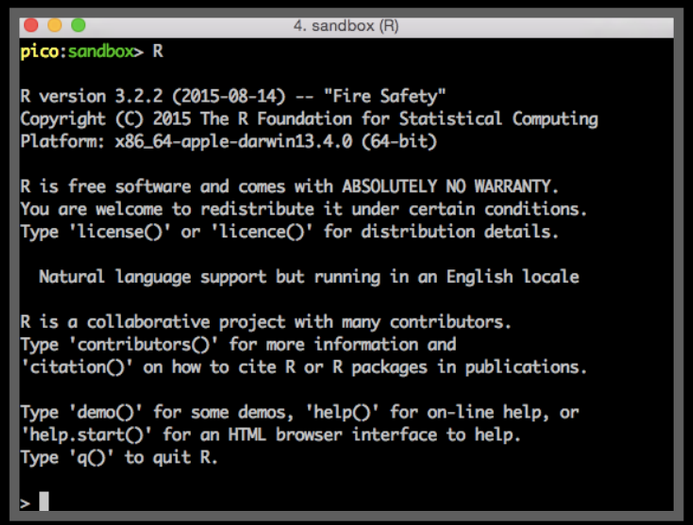
- **Key point**: Save your R script!

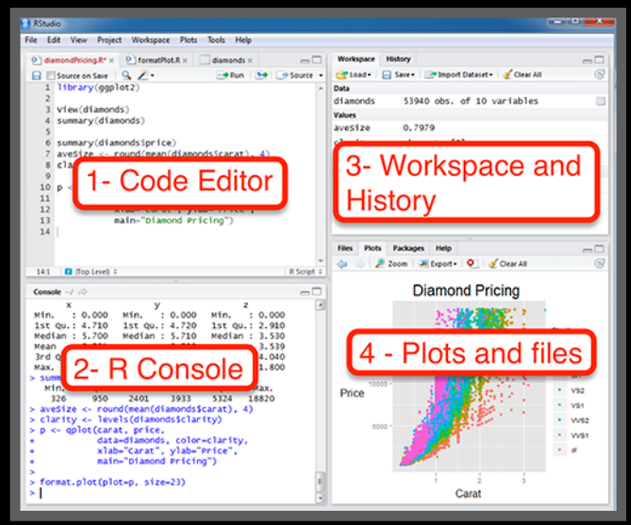
Side-note: RStudio shortcuts



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)
```

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!

Optional Exercise

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

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

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

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

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

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"
[19] "logical"                "loglin"                 "plogis"
     "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*| << *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(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 `dlnrel` 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`.

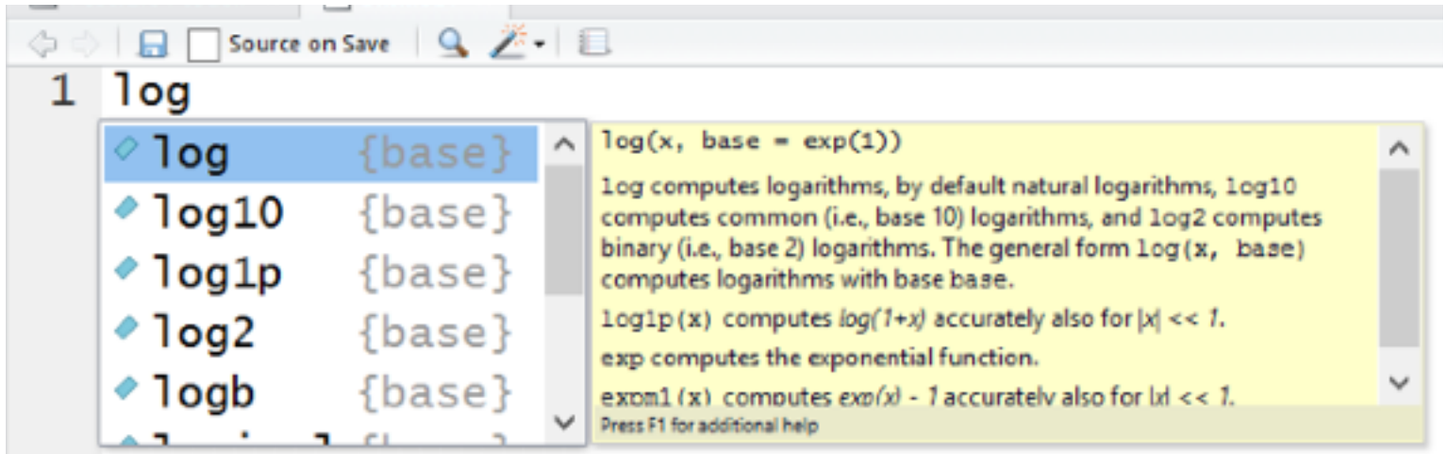
Examples

Sample code showing how it works

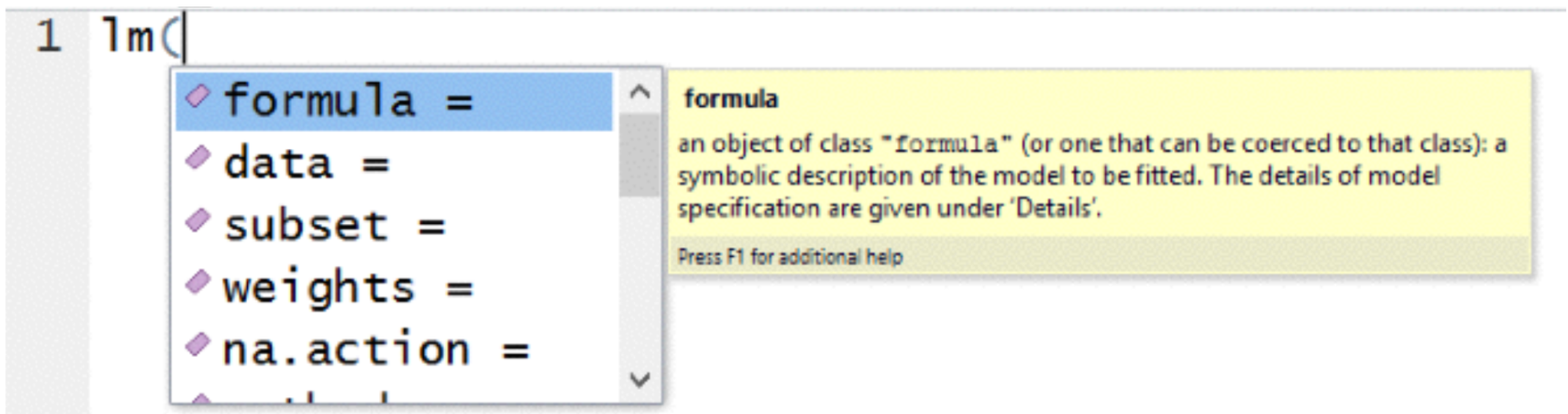
```
log(exp(3))
log10(1e7) # = 7

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

RStudio quick help

- Start typing `log` in the Scripts window (top-left) and a list of available functions starting with those letters appears, plus help



- Try typing `lm(` and then <Tab> for the arguments of the `lm()` function



Assigning values

```
answer <- log(2.5)
```

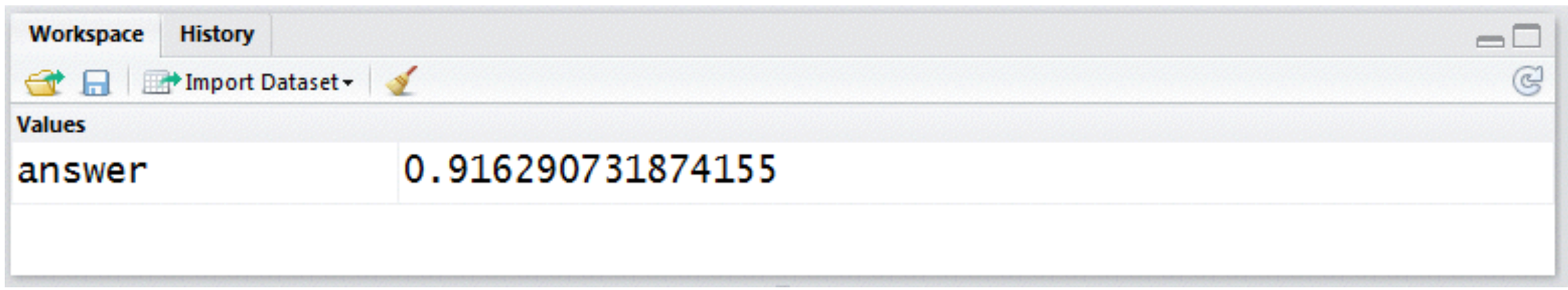
Assign the result of log(2.5) to a new object called “answer”

```
answer = log(2.5)
```

= can be used instead of <- but is less common

optional argument

```
answer <- log(2.5, base=10)
```



When you run this command, an object “answer” is created in the workspace that is assigned the value of 0.91629… In RStudio, the top right window lists all the objects in the current workspace

Vectors

A vector is a one-dimensional ordered collection of the same type of object

`c()` is a function that concatenates values together

```
> lengths <- c(7.8, 9.0, 7.1, 8.8, 8.8)
> lengths
```

this is a vector of numbers

```
[1] 7.8 9.0 7.1 8.8 8.8
```

```
1:10
```

the `:` function is used for consecutive numbers

```
seq(from=1, to=10, by=2)
```

`seq` function allows more flexibility

```
seq(1,10,2)
```

default order of parameters, no labels

```
seq(from=1, to=10,length.out=5)
```

vector of exactly five numbers between `from` and `to`

Vector operations work element-wise

```
> (x <- 1:3)
[1] 1 2 3

> log(x)
[1] 0.0000000 0.6931472 1.0986123

> x+1
[1] 2 3 4

> x*2
[1] 2 4 6
```

```
> y <- 4:6
> x + y
[1] 5 7 9

> y - x
[1] 3 3 3

> x / y
[1] 0.25 0.40 0.50

> x * y
[1] 4 10 18
```

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/ >