Week 2: Fundamentals of Coding

SESSION 1: PYTHON AND R

STARFISH SCHOOL 2022

How is this going to work?

Building skills in Python and R

Throughout this and next lecture, we will balance teaching you skills in Python and R. Some of you may be expert in one but need brushing up in the other.

We think these balance each other well and will help you also see how the fundamentals underneath both (e.g. planning, pseudocode) work

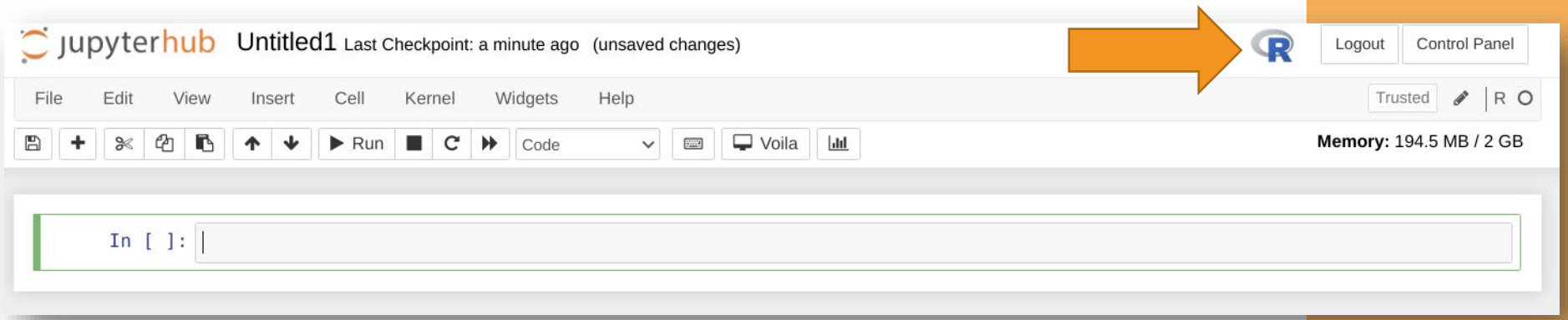
What is object-oriented programming anyway?

Rather than focusing only on functions or modules, the code is centered around the ***objects*** in the code (think: variables, lists, dictionaries) and their ***properties*** *and* ***methods*** that act on the objects.

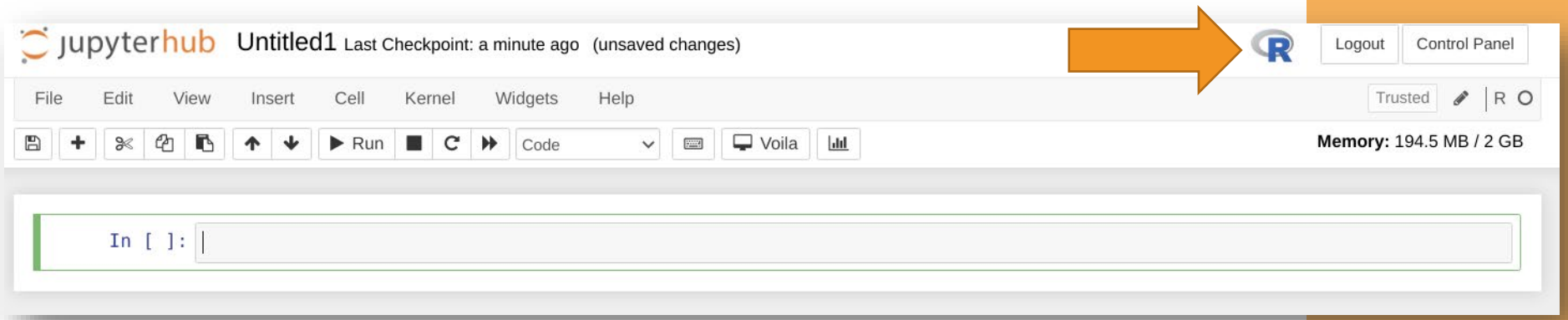
This can take some getting used to initially, but it helps code be more self-contained and readable/useable for future you!

Interfaces

- This is a screenshot of a Jupyter Notebook running R

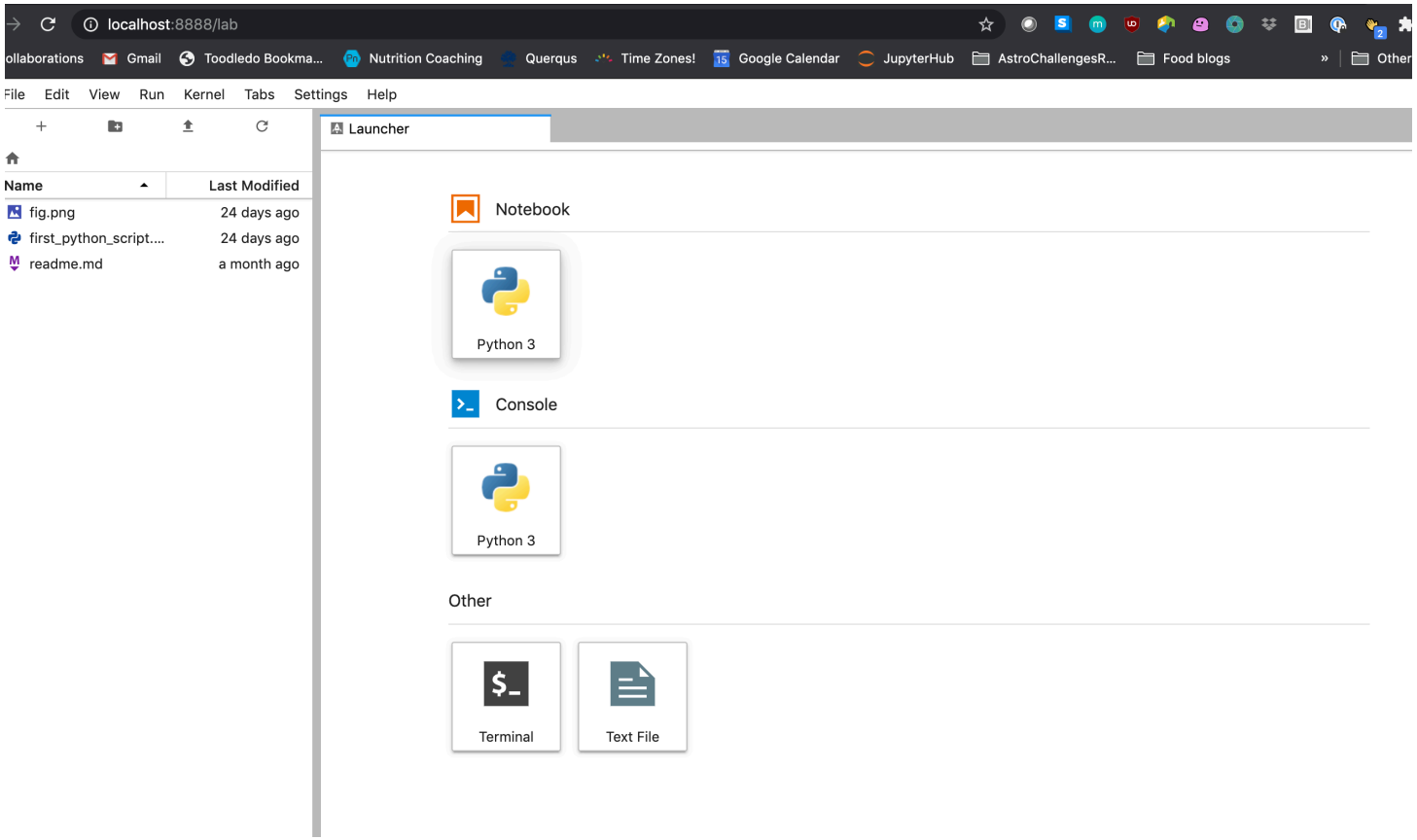


- This is a screenshot of a Jupyter Notebook running R



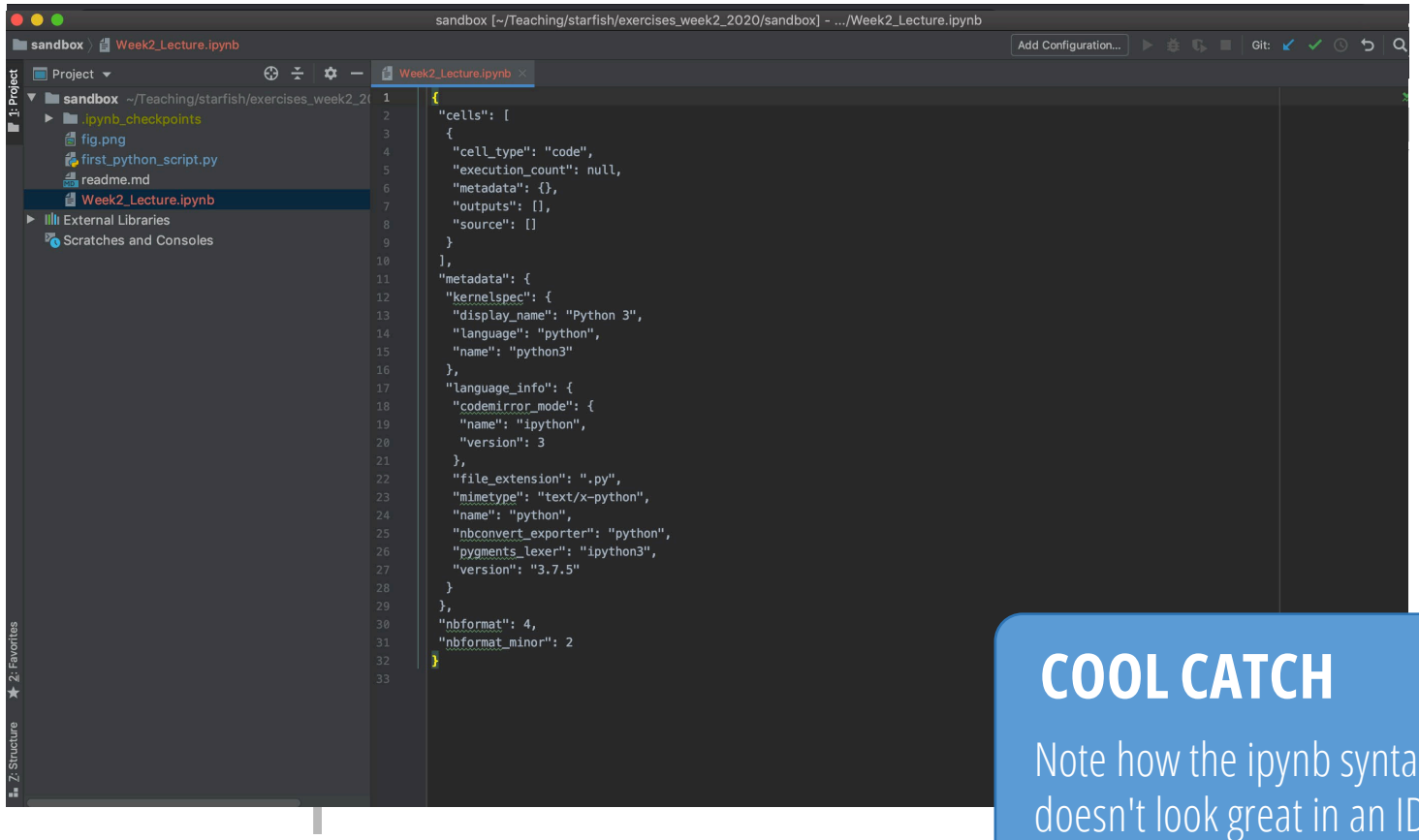
FUN FACT

JuPyteR is a play on the words *Julia*, *Python*, and *R*



- This is a screenshot of Jupyter Lab.

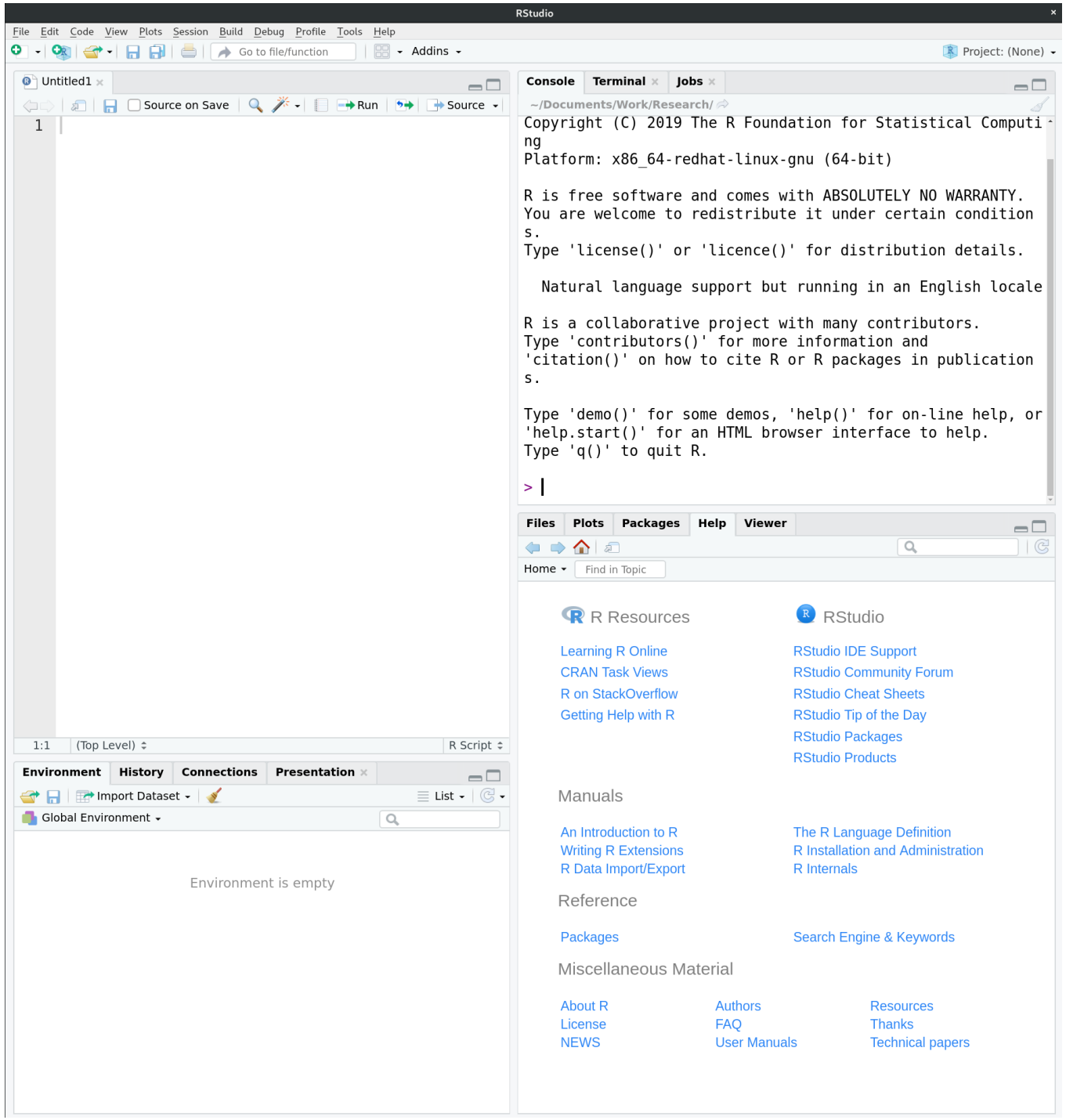
- You can use it to start a Juypter notebook where we will add some code



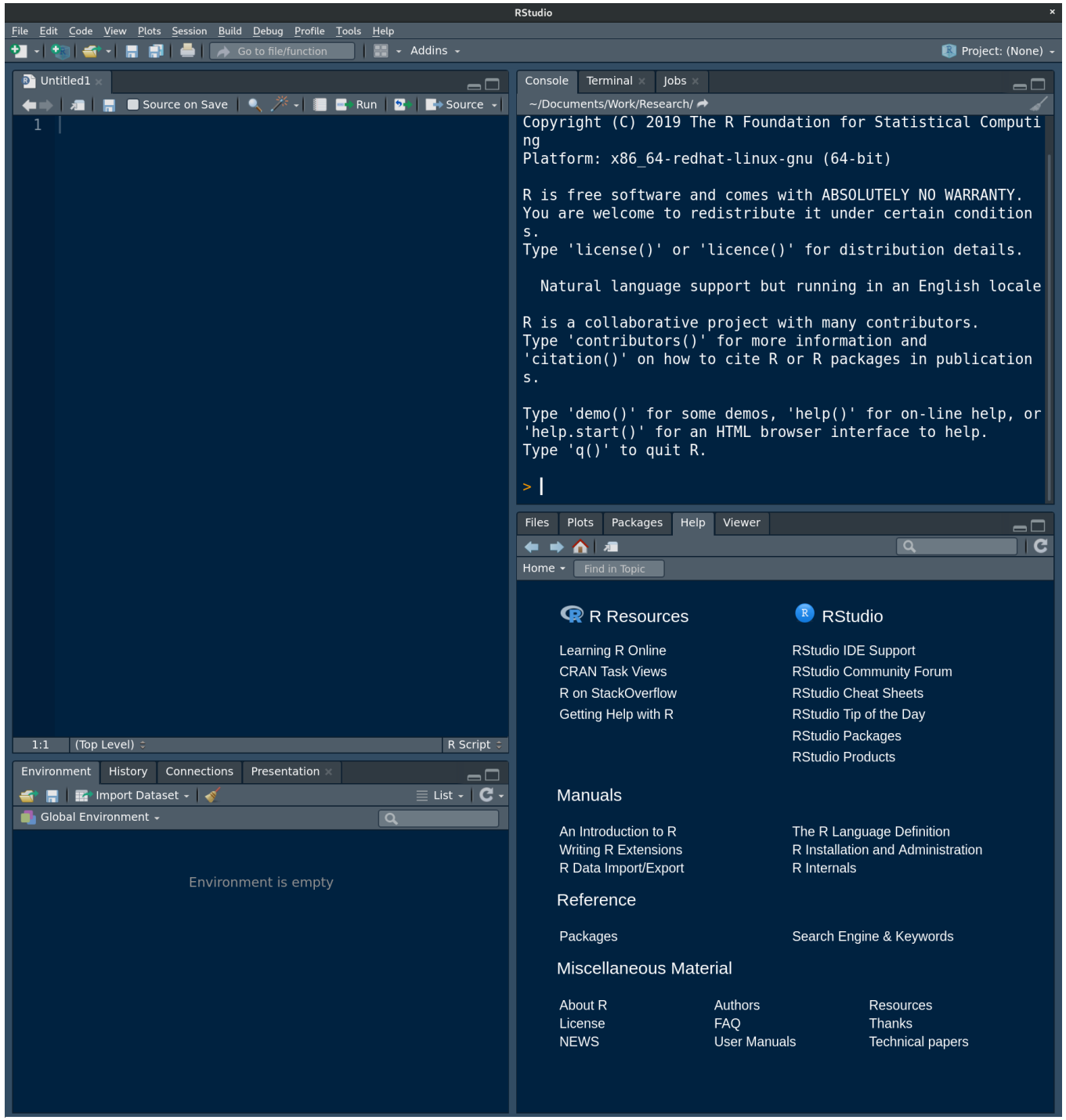
COOL CATCH

Note how the ipynb syntax doesn't look great in an IDE. It is better for scripts.

- You could also use an Integrated Development Environment (IDE) if you want to see your whole project all together.



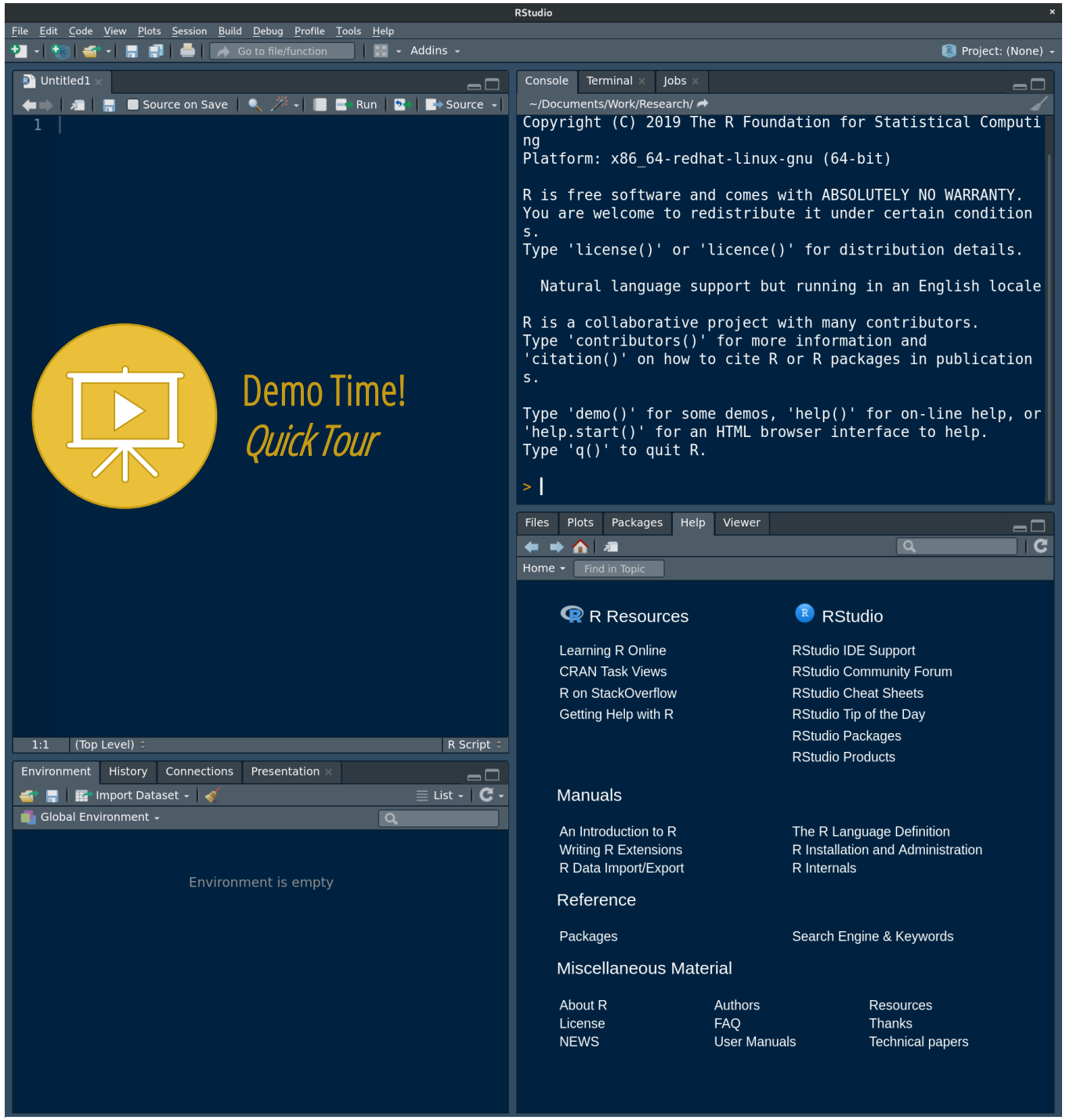
- This is a screenshot of RStudio, an IDE for R
- You can use it to start/run R scripts, R projects, R Markdown documents, .. many options!



- This is a screenshot of RStudio, an IDE for R
- You can use it to start/run R scripts, R projects, R Markdown documents, .. many options!

HOT TIP

You can change the colour theme, typeface, panel layout, etc. ... lots of options!



- This is a screenshot of RStudio, an IDE for R
- You can use it to start/run R scripts, R projects, R Markdown documents, .. many options!

HOT TIP

You can change the colour theme, typeface, panel layout, etc. ... lots of options!

How to begin

First you'll need to:

write some pseudocode (think of these as blueprints for the house)

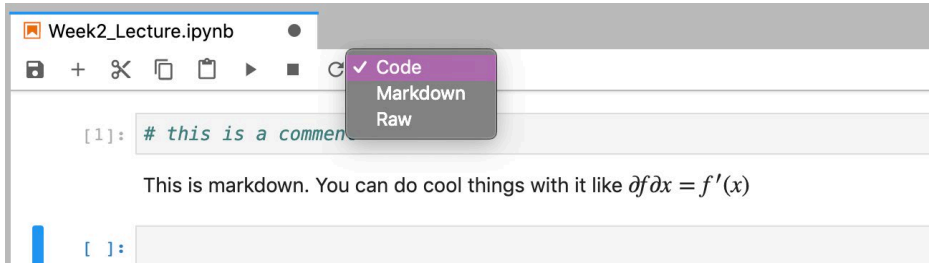
customize your space using libraries/modules (think of building your house)

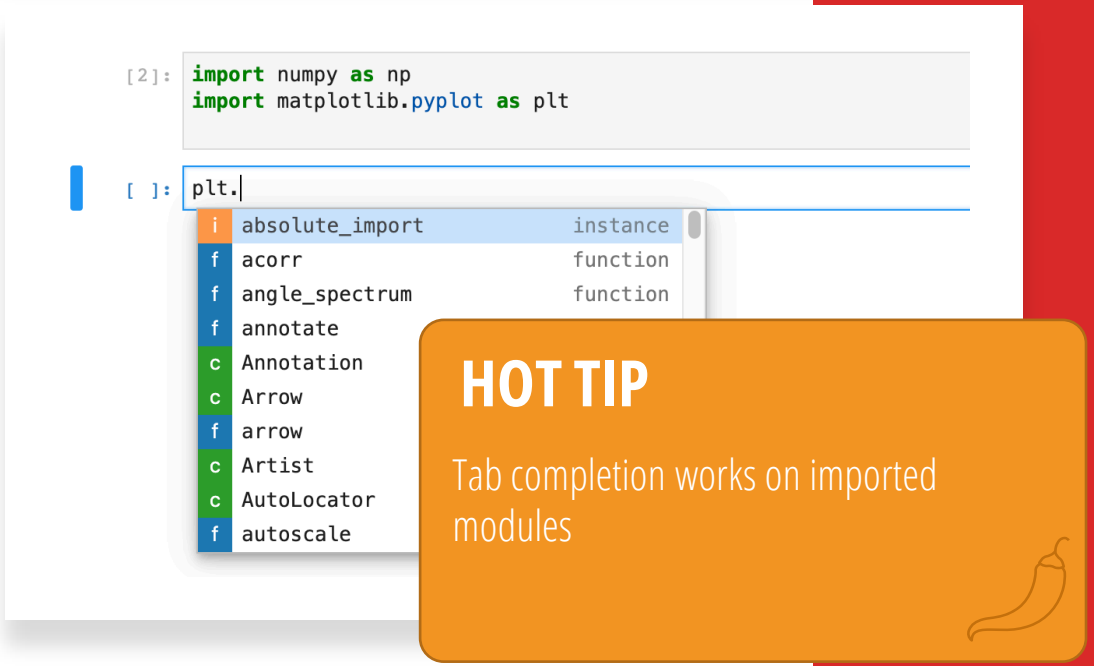
and then define your variables (who lives there)

Python 1

Modules and importing

- Adding lots of markdown to your notebooks really helps you keep things clean/organised and helps future you know what you are doing in the big picture rather than only comments (more line-by-line instructions)
- Modules can either those specified globally in your python path, or local files (turn your codes into modules to import!). Small codes are easier to read and good to reuse.
- Using an alias (e.g. np) is useful to keep modules organised



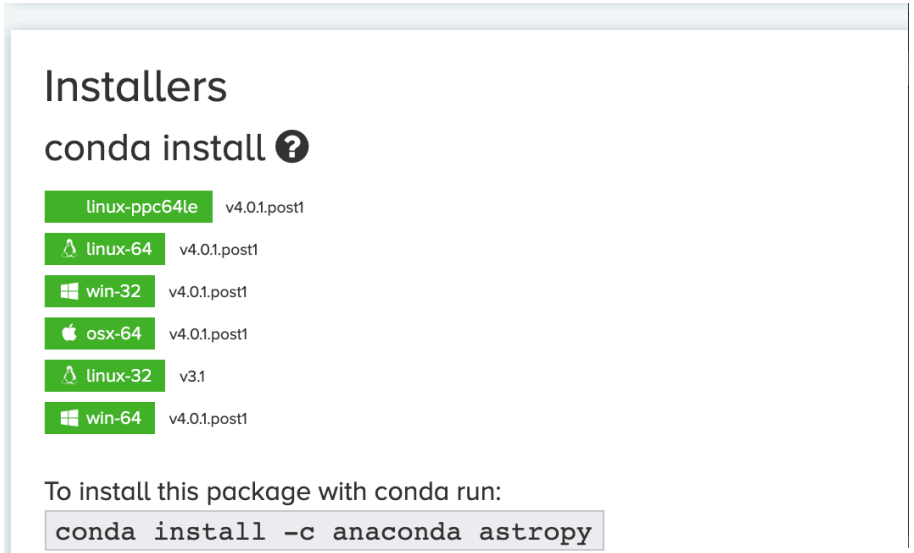


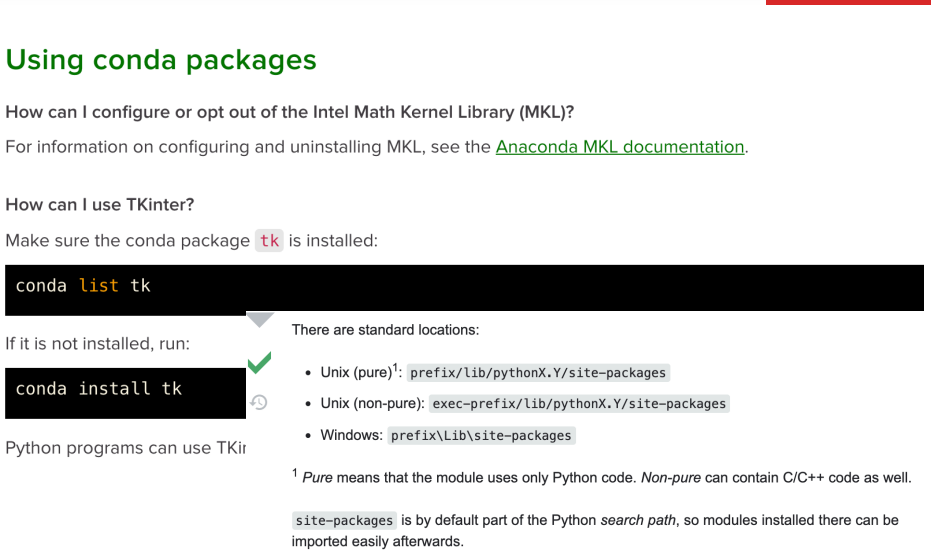
Installing Python Packages

- Anaconda makes this pretty easy – but it is important to also look up the package/dependenies

- Installing a python package:
  - Buying a book and putting it on the shelf
  - `conda install package`

- Where are my packages located?
  - Finding the spice rack
  - Conda list





Some Basic Python Variables

Variable Type	What it is	Examples
int	An integer	2, -30, 3002
float	A real number, and some special values	0.0, 3.4, -2.3E33, nan, inf
string	A “string” of text	“a”, “abba”, “0.3”
bool	A True/False value	True, False

Some Basic Python Variables

Variable Type	What it is	Examples
int	An integer	2, -30, 3002
float	A real number, and some special values	0.0, 3.4, -2.3E33, nan, inf
string	A “string” of text	“a”, “abba”, “0.3”
bool	A True/False value	True, False

HOT TIP

If you want to figure out what type of variable you have, use the `type()` command!

Some Basic Python Data Structures

Variable Type	What it is	Examples
list	A group of variables	`["a", 4.3, True]`
tuple	A group of variables (that you can't change)	`("a", 4.3, True)`
dict	A group of variables with keys to refer to them	`{"a":3, "b": True, "x": "mama"}`
set	A group of variables where every item is unique	`{'a', 'b', 'c'}`

Some Basic Python Data Structures

Variable Type	What it is	
list	A group of variables	["
tuple	A group of variables (that you can't change)	(" True)
dict	A group of variables with keys to refer to them	{"a":3, "b": True, "x": "mama"}
set	A group of variables where every item is unique	{'a', 'b', 'c'}

HOT TIP

We're going to go into "indexing" in depth next week, but for now, what you need to know is that the first item in a list/tuple can be grabbed using the index "0", second is "1". You can grab everything using ":"

Some Basic Python Data Structures

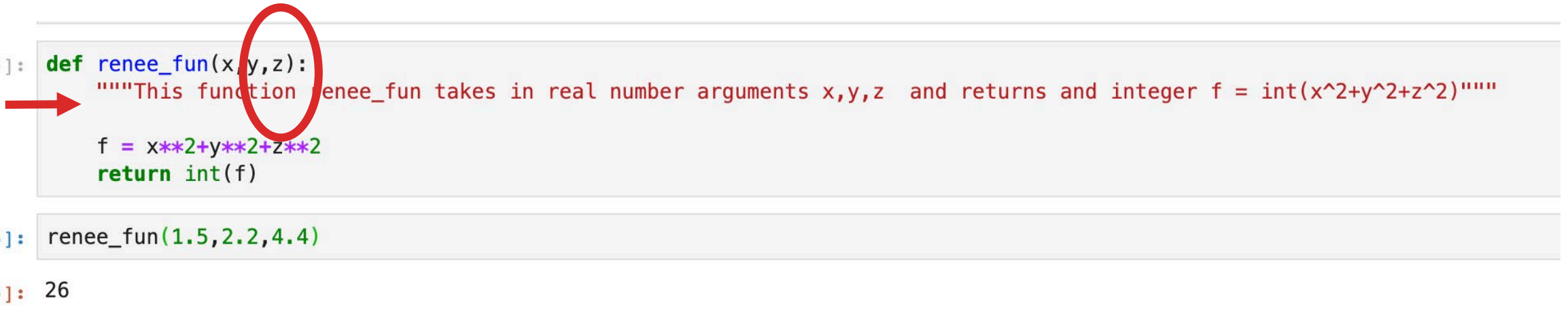
Variable Type	What it is	Examples
list	A group of variables	`["a", 4.3, True]`
tuple	A group of variables (that you can't change)	`("a", 4.3, True)`
dict	A group of variables with keys to refer to them	`{"a":3, "b": True, "x": "mama"}`
	A group of variables where every item is unique	`{'a', 'b', 'c'}`

COOL CATCH

All variables within a python data structure don't have to be the same variable type!

Functions, variables, docstrings in Python

- A few things to remember about function definitions in python



```
[6]: def renee_fun(x,y,z):
         """This function renee_fun takes in real number arguments x,y,z  and returns and integer f = int(x^2+y^2+z^2)"""

         f = x**2+y**2+z**2
         return int(f)

[5]: renee_fun(1.5,2.2,4.4)

[5]: 26
```

- Tab completion automatic in most browsers/IDE. Be careful coding without it!
- Three apostrophes = comment
- Make sure to return something from your function

Functions, variables, docstrings in Python

- A few things to remember about function definitions in python

```
[6]: def renee_fun(x,y,z):
         """This function renee_fun takes in real number arguments x,y,z  and returns and integer f = int(x^2+y^2+z^2)"""

         f = x**2+y**2+z**2
         return int(f)
```

```
[5]: renee_fun(1.5,2.2,4.4)
```

```
[5]: 26
```

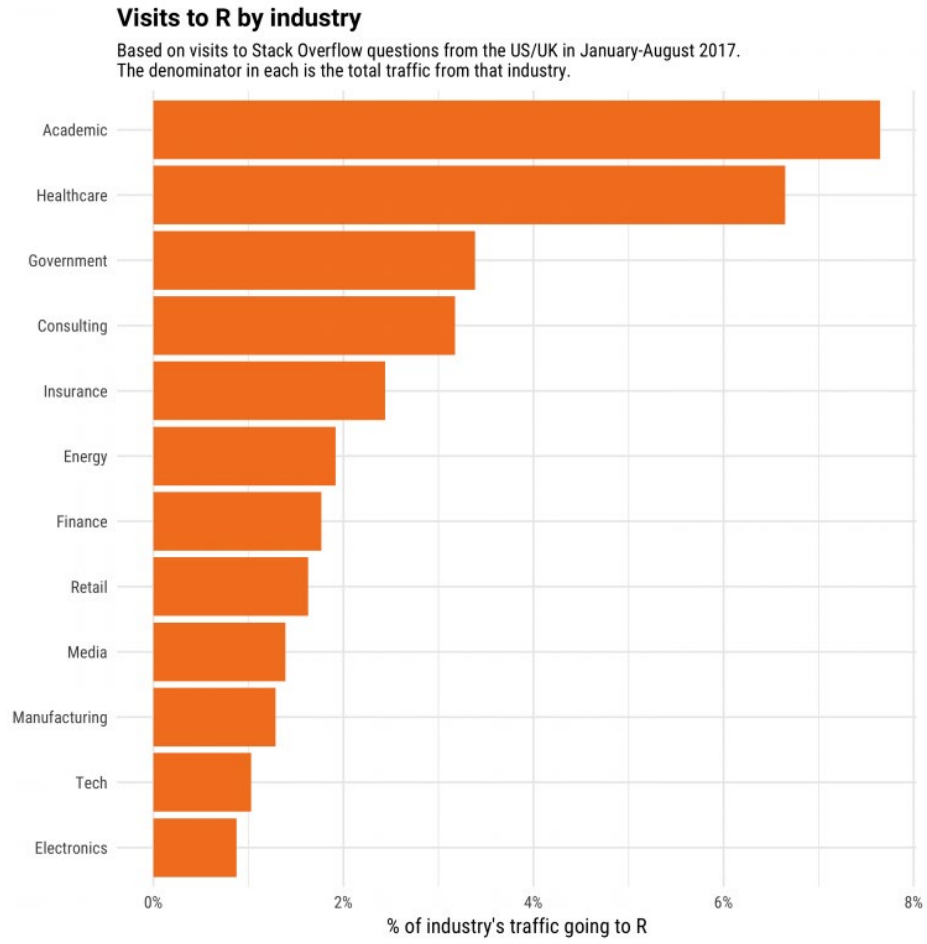
- Tab completion automatic in most browsers/IDE. Be careful coding without it!
- Three apostrophes = comment
- Make sure to return something from your function

Demo Time!
Writing functions

R 1

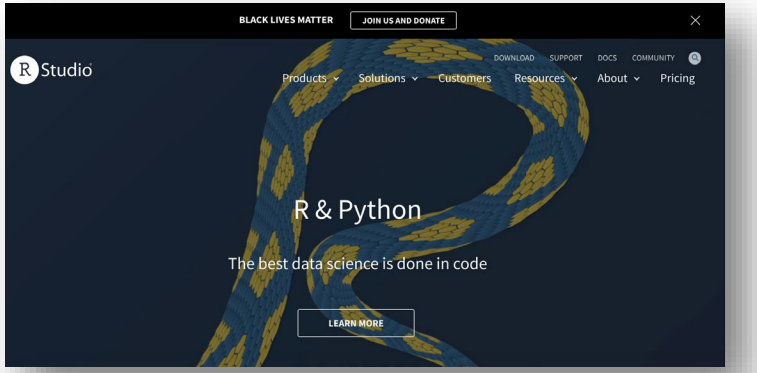
What is R? Who uses R?

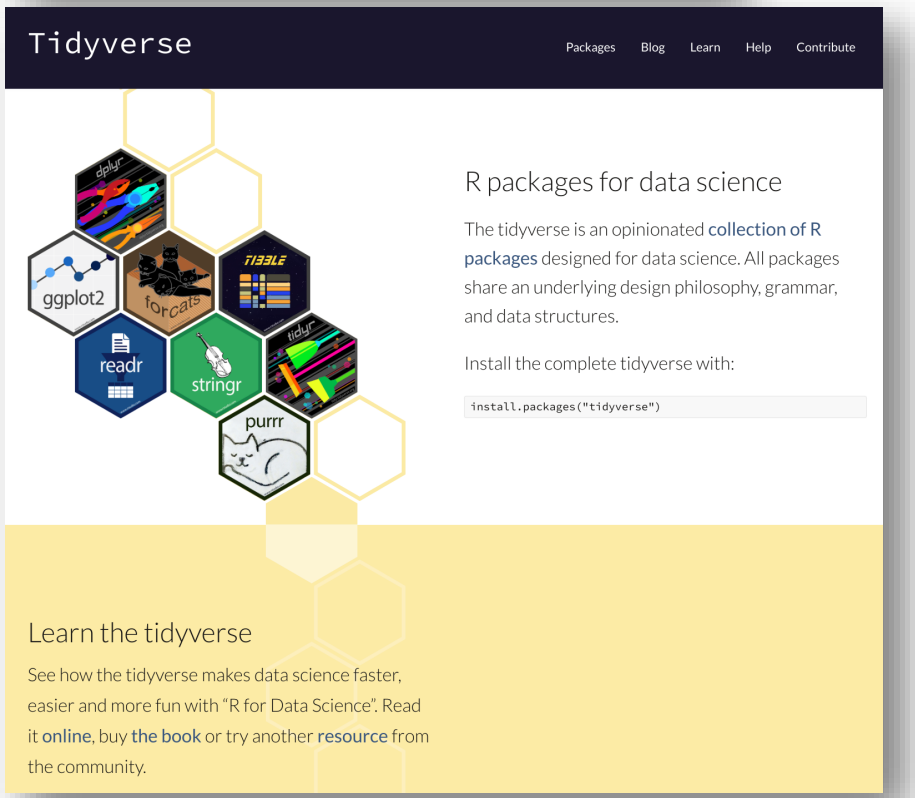
- In academia: *statisticians, biostatisticians, genomics, social science, ecology, environmental science, actuarial science, ...*
- In industry: *government, healthcare, consulting, insurance, finance, ...*



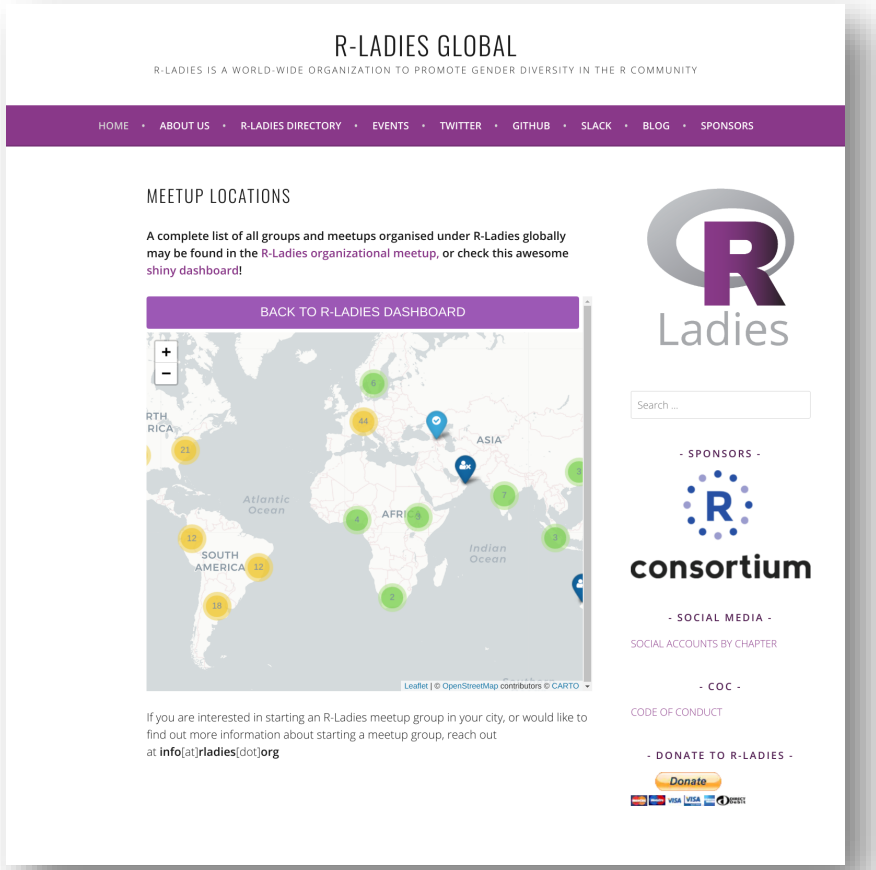
stackoverflow.blog/2017/10/10/impressive-growth-r/

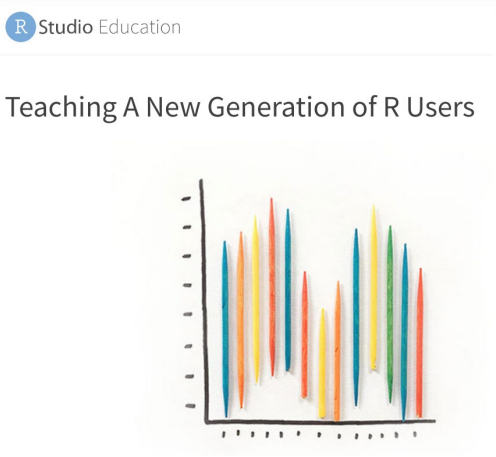
Communities





R Foundation taskforce on women and other under-represented groups



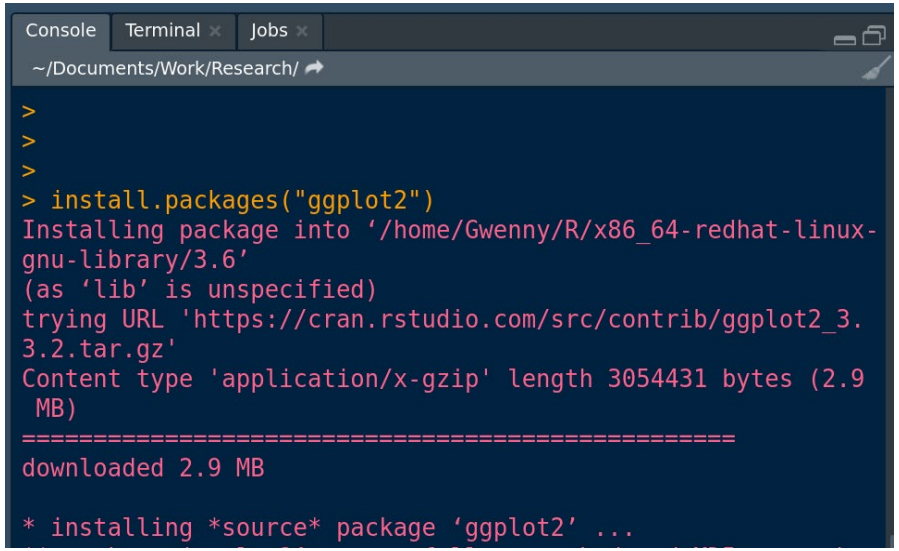


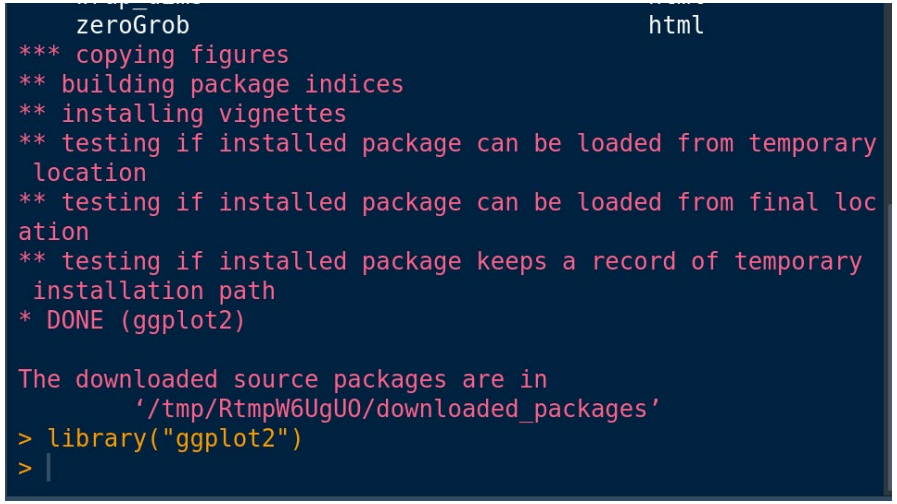
Installing & Using R Packages

- R makes this relatively painless (because there are strict rules for packages submitted to the Comprehensive R Archive Network)

- Installing an R package:
  - Buying a book and putting it on the shelf
  - `install.packages("nameofpackage")`

- Using an R package:
  - Taking that book off the shelf and using it
  - `library("nameofpackage")`





Searching CRAN Task Views

Packages organized by subject and methodology

https://cran.r-project.org/

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

CRAN Task Views

CRAN task views aim to provide some guidance which packages on CRAN are relevant for tasks related to a certain topic. They give a brief overview of the included packages and can be automatically installed using the ctv package. The views are intended to have a sharp focus so that it is sufficiently clear which packages should be included (or excluded) - and they are *not* meant to endorse the "best" packages for a given task.

- To automatically install the views, the ctv package needs to be installed, e.g., via `install.packages("ctv")` and then the views can be installed via `install.views` or `update.views` (where the latter only installs those packages are not installed and up-to-date), e.g., `ctv::install.views("Econometrics")` `ctv::update.views("Econometrics")`
- The task views are maintained by volunteers. You can help them by suggesting packages that should be included in their task views. The contact e-mail addresses are listed on the individual task view pages.
- For general concerns regarding task views contact the ctv package maintainer.

Topics

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Databases	Databases with R
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
ExtremeValue	Extreme Value Analysis
Finance	Empirical Finance
FunctionalData	Functional Data Analysis
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
Hydrology	Hydrological Data and Modeling
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
MissingData	Missing Data
ModelDeployment	Model Deployment with R
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
NumericalMathematics	Numerical Mathematics
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods

Searching CRAN Task Views

Packages organized by subject and methodology

https://cran.r-project.org/

Demo Time!
Find a package on CRAN

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

CRAN Task Views

CRAN task views aim to provide some guidance which packages on CRAN are relevant for tasks related to a certain topic. They give a brief overview of the included packages and can be automatically installed using the ctv package. The views are intended to have a sharp focus so that it is sufficiently clear which packages should be included (or excluded) - and they are *not* meant to endorse the "best" packages for a given task.

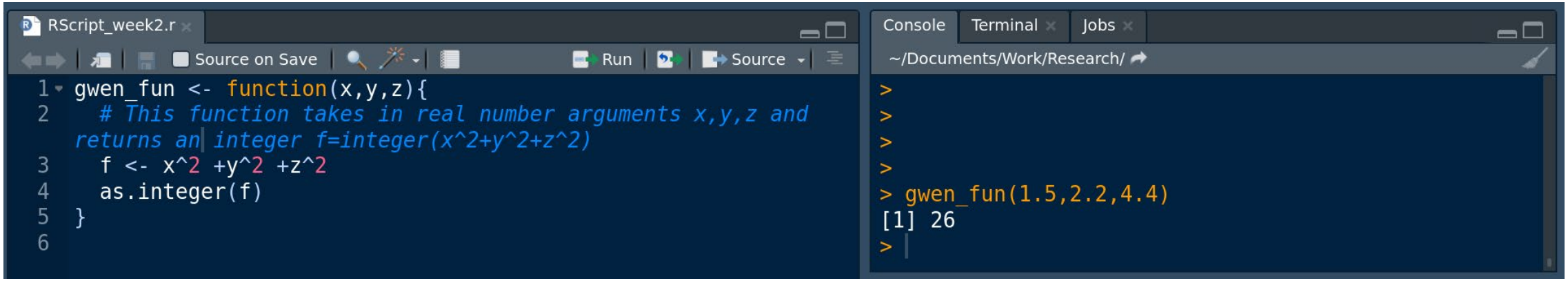
- To automatically install the views, the ctv package needs to be installed, e.g., via `install.packages("ctv")`
  and then the views can be installed via `install.views` or `update.views` (where the latter only installs those packages are not installed and up-to-date), e.g.,
  `ctv::install.views("Econometrics")`
  `ctv::update.views("Econometrics")`
- The task views are maintained by volunteers. You can help them by suggesting packages that should be included in their task views. The contact e-mail addresses are listed on the individual task view pages.
- For general concerns regarding task views contact the ctv package maintainer.

Topics

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Databases	Databases with R
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
ExtremeValue	Extreme Value Analysis
Finance	Empirical Finance
FunctionalData	Functional Data Analysis
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
Hydrology	Hydrological Data and Modeling
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
MissingData	Missing Data
ModelDeployment	Model Deployment with R
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
NumericalMathematics	Numerical Mathematics
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods

Functions and variables in R

- Defining functions



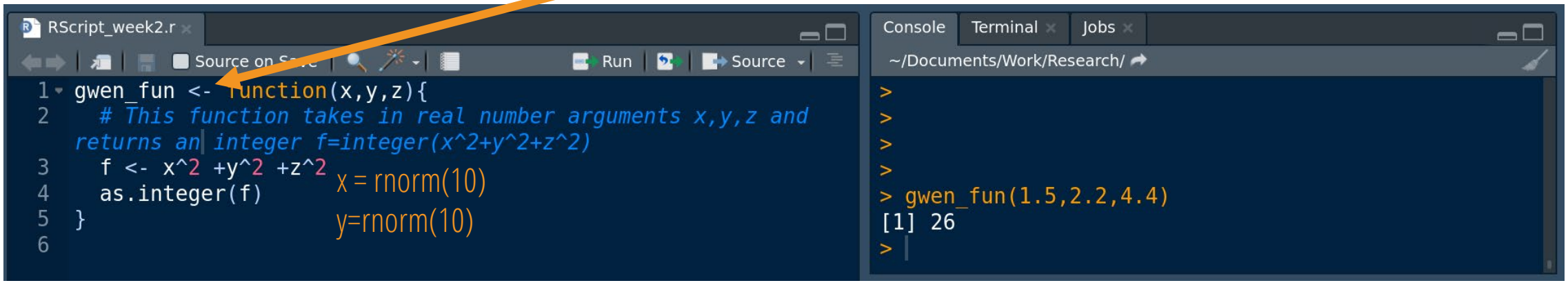
- # precede any comments
- Tab completion works for packages, functions
- Just like in python, you have to actually return something!

Functions and variables in R

- Defining functions

HOT TIP

You can use an "arrow" or a = sign to define stuff. Either works!



- # precede any comments
- Tab completion works for packages, functions
- Just like in python, you have to actually return something!

BREAK!

Python 2

Input and output in Python

- Simple (numerical) reading from file using numpy loadtxt. Doesn't work well for mixed data -- which is most data.

```
filename = 'test.dat'
data = np.loadtxt('test.dat', unpack=True)
```

- Pandas dataframe already loads file into dataframe

```
data = pd.read_csv('test.csv')
data
```

	x data	y data	colour
0	0.00	0.0000	blue
1	0.01	0.0001	green
2	0.02	0.0004	blue
3	0.03	0.0009	green
4	0.04	0.0016	green
5	0.05	0.0025	blue
6	0.06	0.0036	blue
7	0.07	0.0049	green

Input and output in Python

- Simple (numerical) writing to file using numpy savetxt. This is quick and dirty, but doesn't work well for mixed data -- which is most data.

```
x = np.logspace(-3,4,100)
y = x**2
np.savetxt('test.dat', np.transpose((x,y)), delimiter=' ')
```

- Pandas dataframe already loads file into dataframe

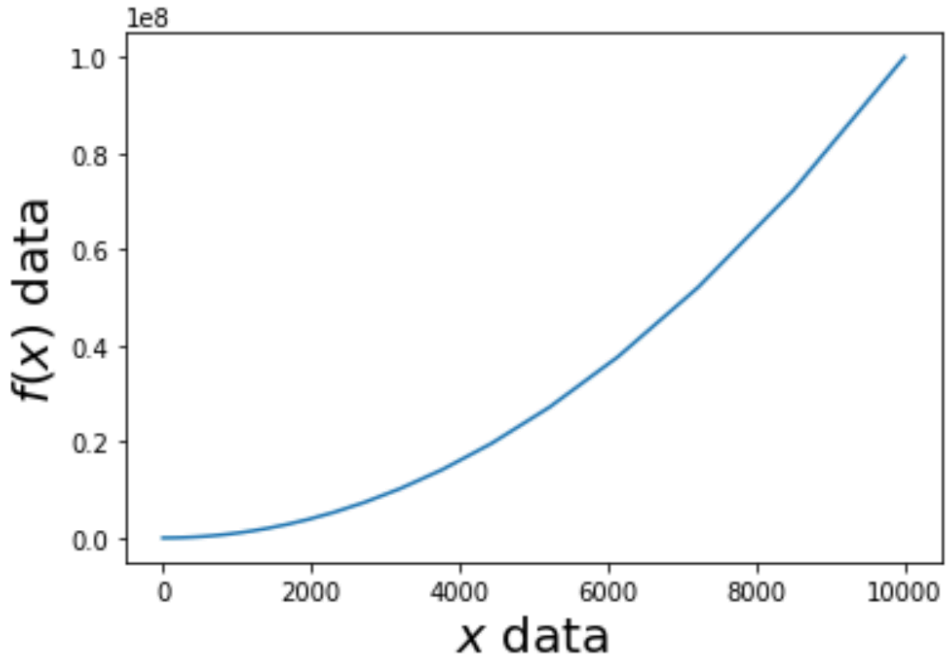
```
data = pd.read_csv('test.csv')
data.to_csv('test2.csv')
```

Plotting and visuals in Python

- Simplest and very basic thing for arrays/numerical data: matplotlib

```
filename = 'test.dat'
data = np.loadtxt('test.dat', unpack=True)
plt.plot(data[0],data[1])
plt.xlabel(r'$x$ data', fontsize=20)
plt.ylabel(r'$f(x)$ data', fontsize=20)
```

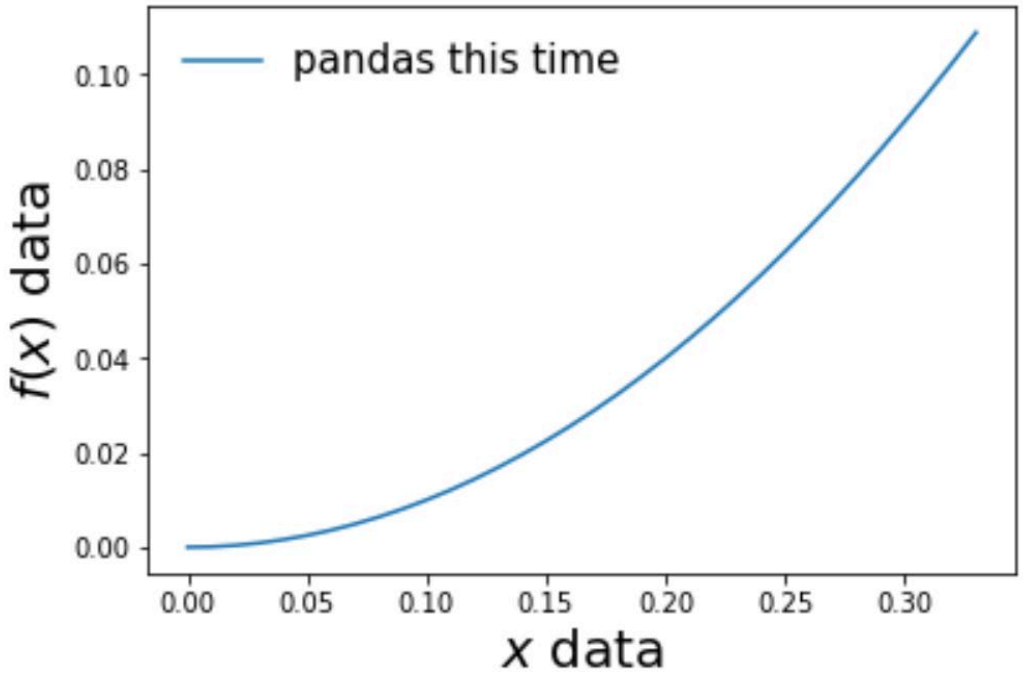
```
Text(0,0.5,'$f(x)$ data')
```



Plotting and visuals in Python

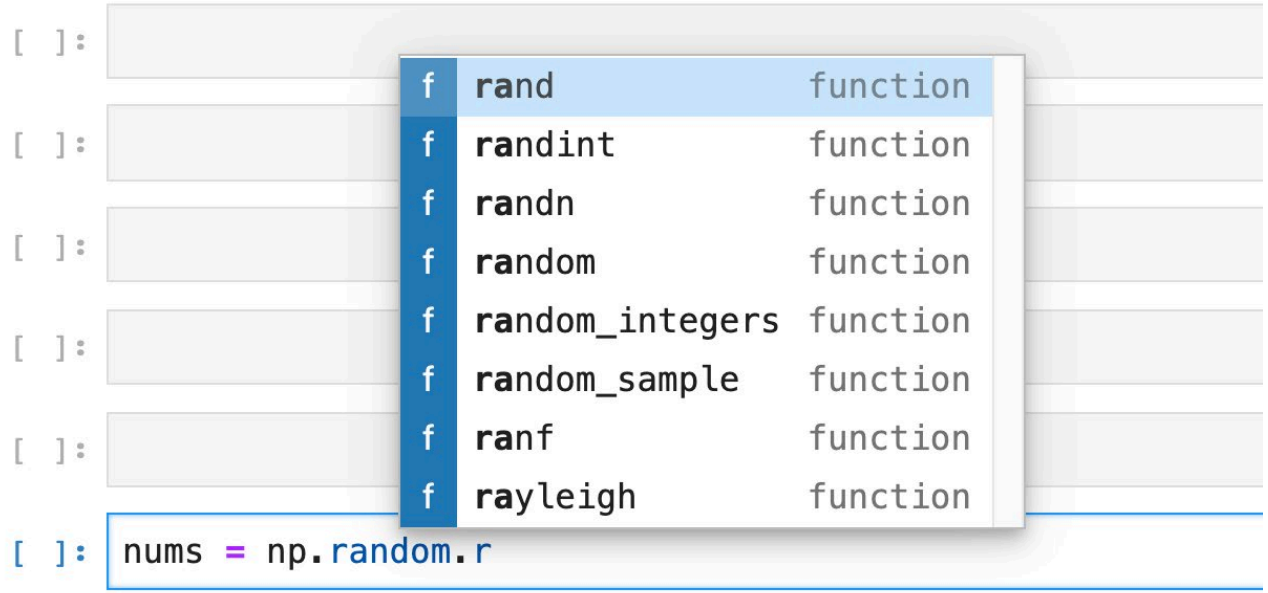
- Matplotlib also works for pandas dataframe

```
plt.plot(data['x data'],data['y data'], label='pandas this time')
plt.xlabel(r'$x$ data', fontsize=20)
plt.ylabel(r'$f(x)$ data', fontsize=20)
leg=plt.legend(loc='best', fontsize=15)
leg.draw_frame(False)
```



Random numbers

- Lots of ways to get random numbers



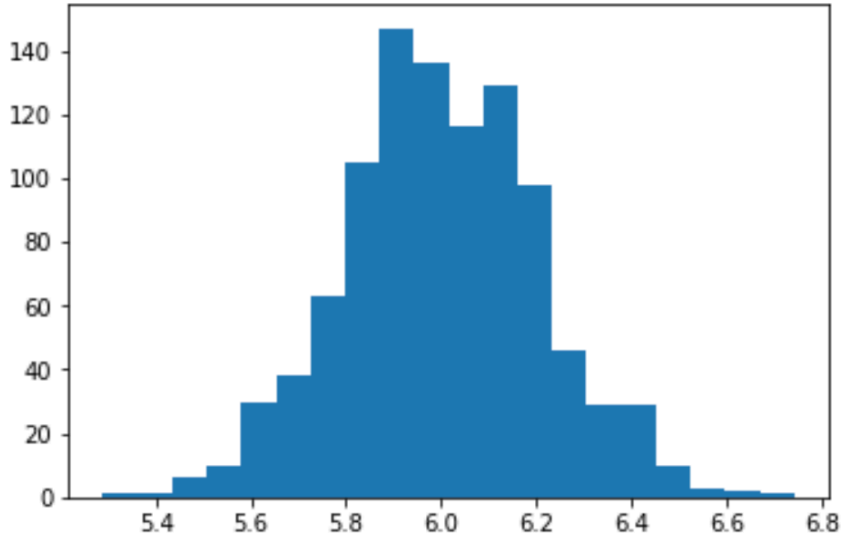
Random numbers

- Mean, variance = 0,1 by default, easy to generate any distribution

```
nums = 6 + 0.2* np.random.randn(1000)
```

```
np.histogram(nums)
plt.hist(nums,20)
```

```
(array([  1.,   1.,   6.,  10.,  30.,  38.,  63., 105., 147., 136., 116.,
        129.,  98.,  46.,  29.,  29.,  10.,   3.,   2.,   1.]),
 array([5.28869683, 5.36138985, 5.43408286, 5.50677587, 5.57946888,
        5.65216189, 5.72485491, 5.79754792, 5.87024093, 5.94293394,
        6.01562695, 6.08831997, 6.16101298, 6.23370599, 6.306399  ,
        6.37909202, 6.45178503, 6.52447804, 6.59717105, 6.66986406,
        6.74255708]),
 <a list of 20 Patch objects>)
```



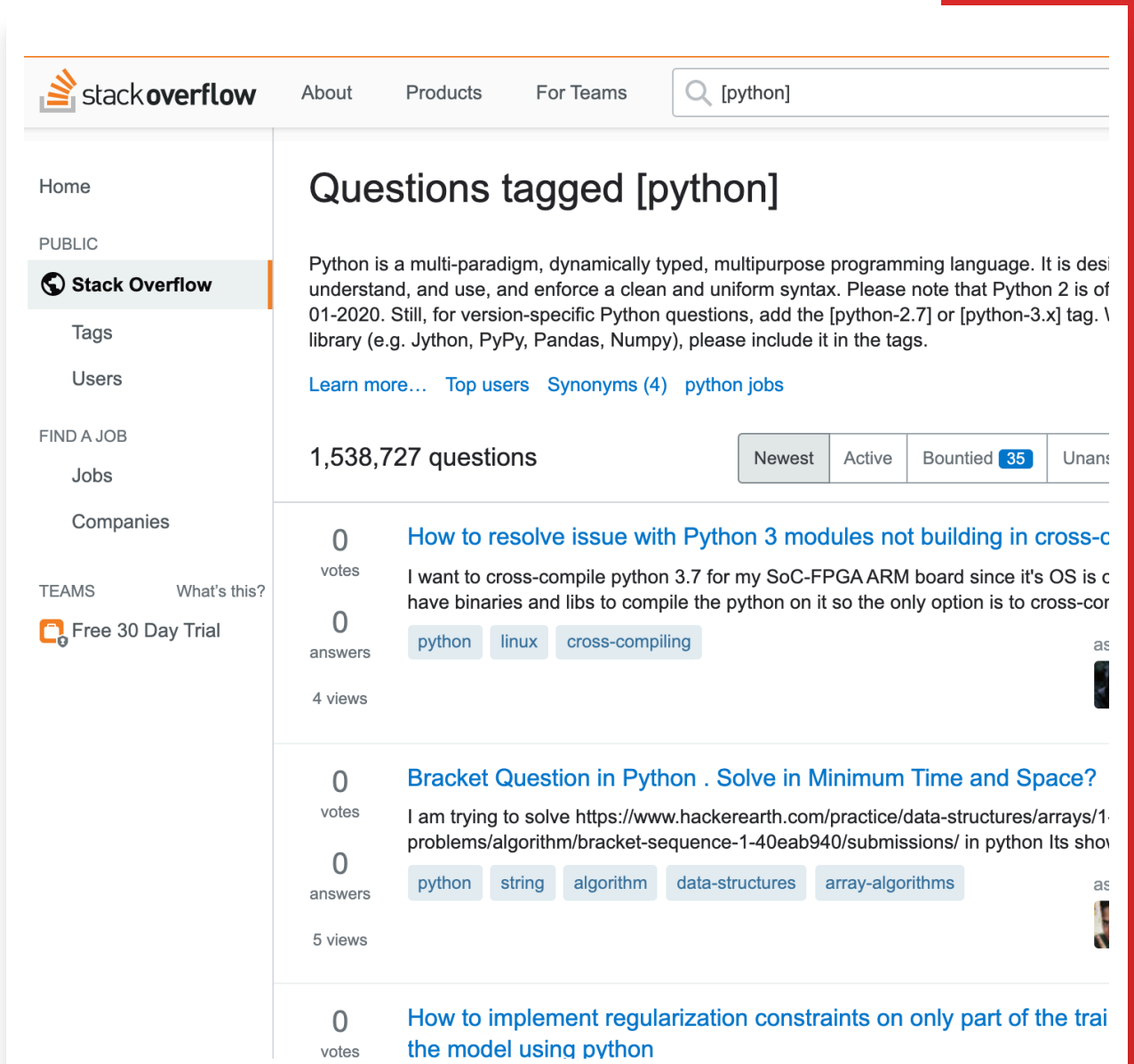
HOT TIP

Note the difference between matplotlib and numpy histograms

Python Tips

HOT TIP

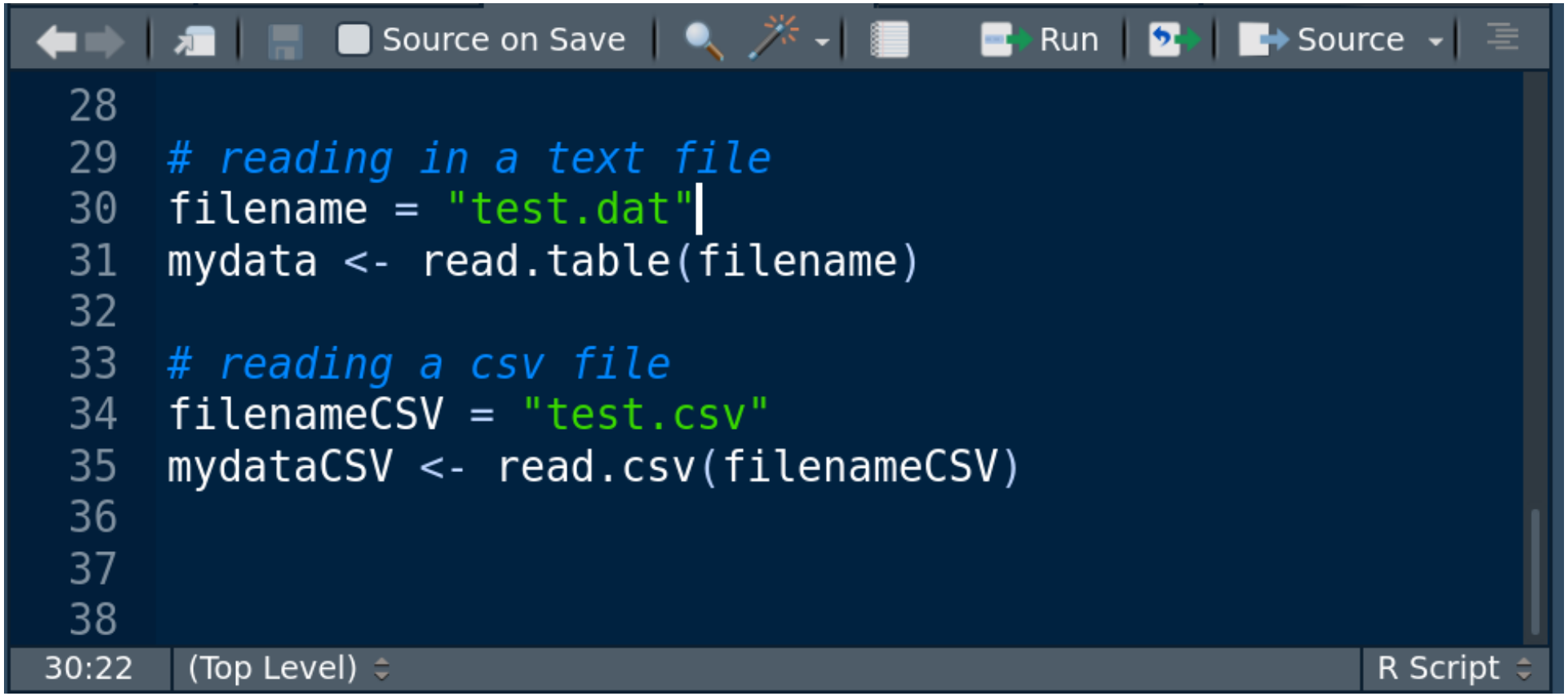
Figuring our how to be efficient in your search terms makes a HUGE difference – also know which python version to include helps!



R 2

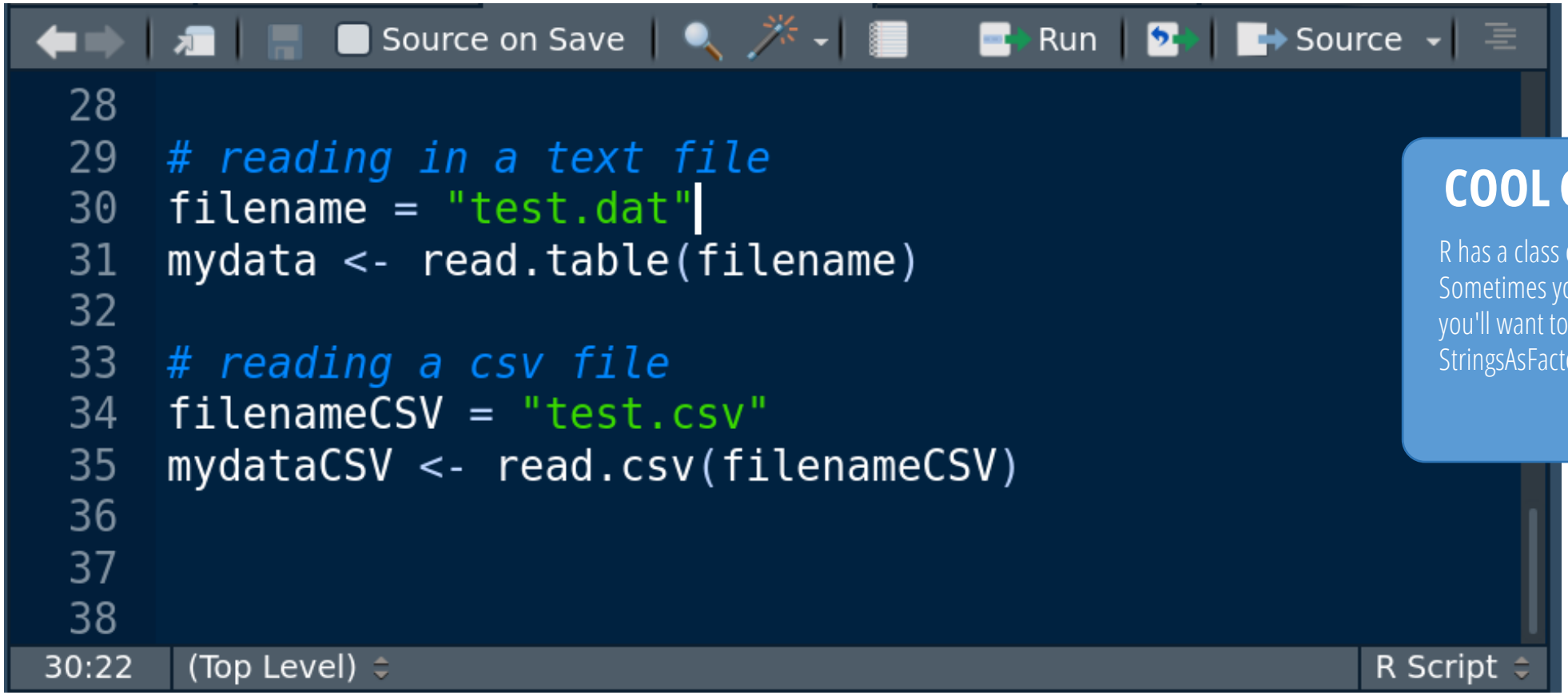
Input and Output into R

- Lots of options for reading in data in basic R (i.e. without extra packages)



Input and Output into R

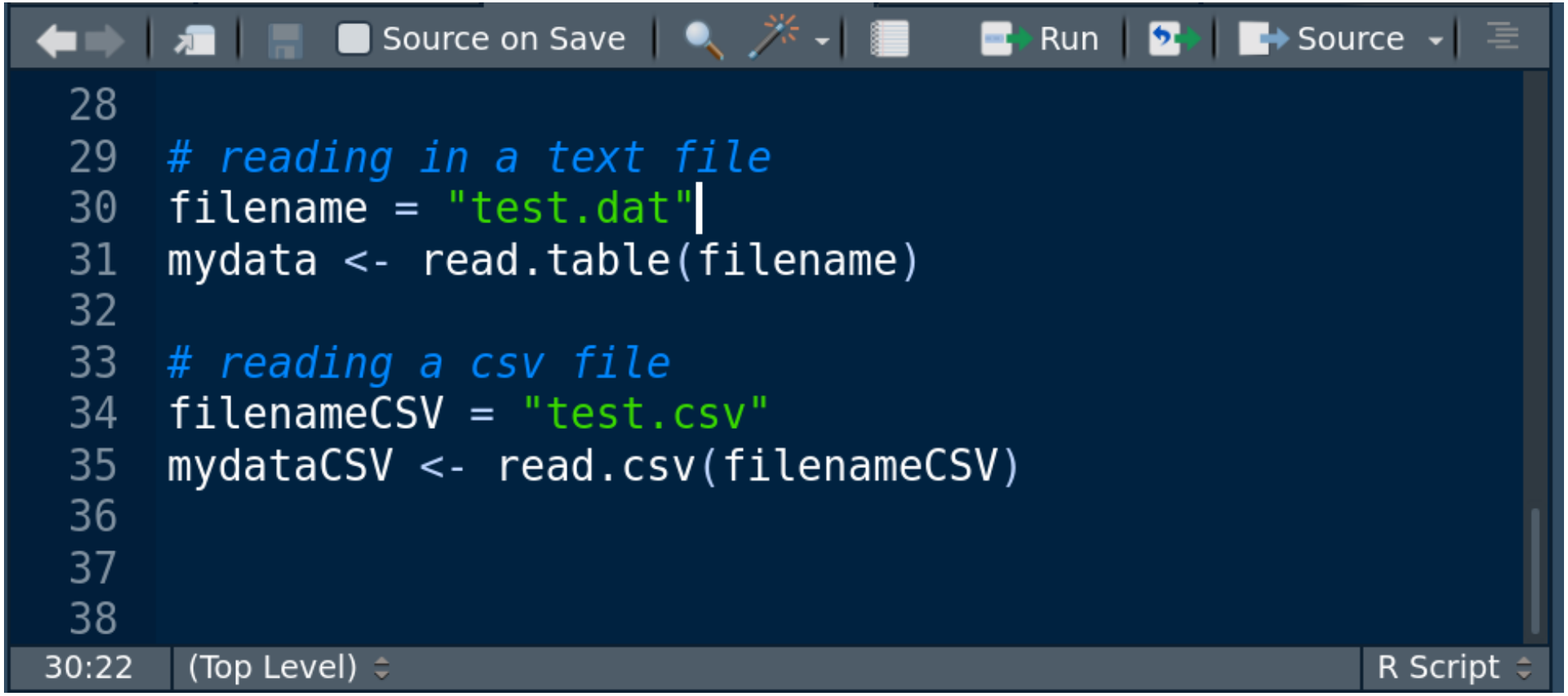
- Lots of options for reading in data in basic R (i.e. without extra packages)



COOL CATCH

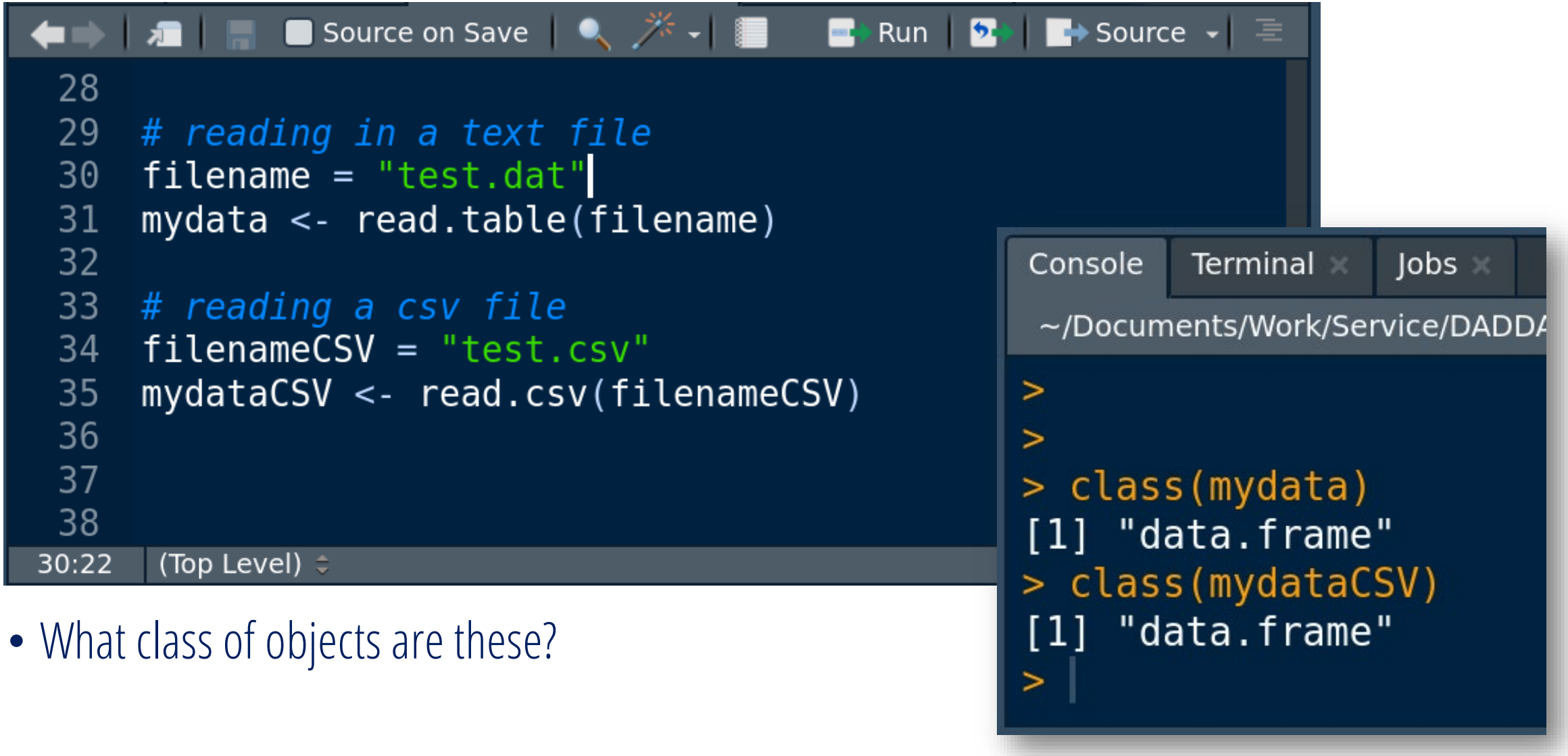
R has a class of objects called *Factors*. Sometimes you don't want these, so you'll want to set the argument StringsAsFactors = FALSE

Input and Output into R



- What class of objects are these?

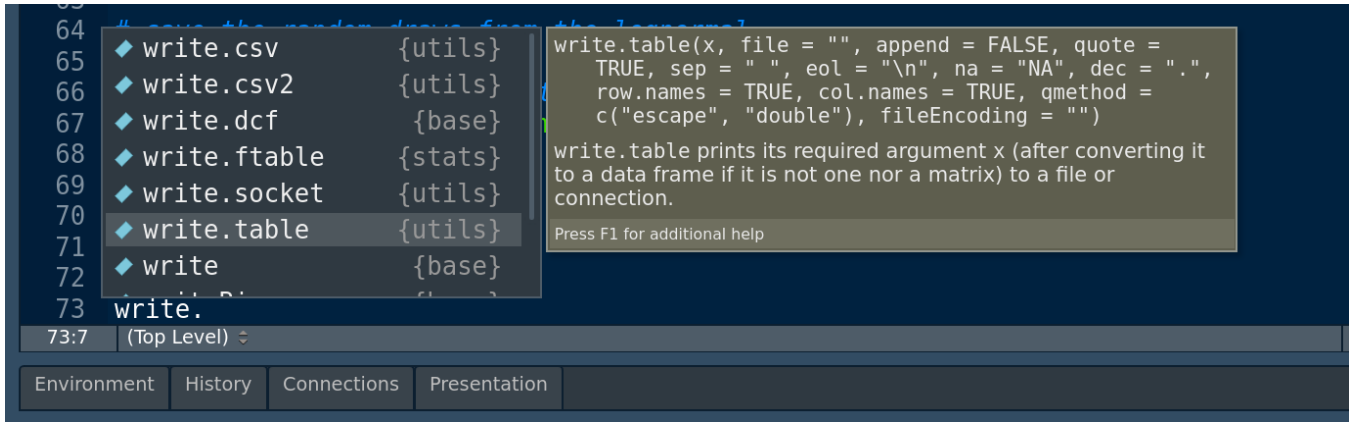
Input and Output into R



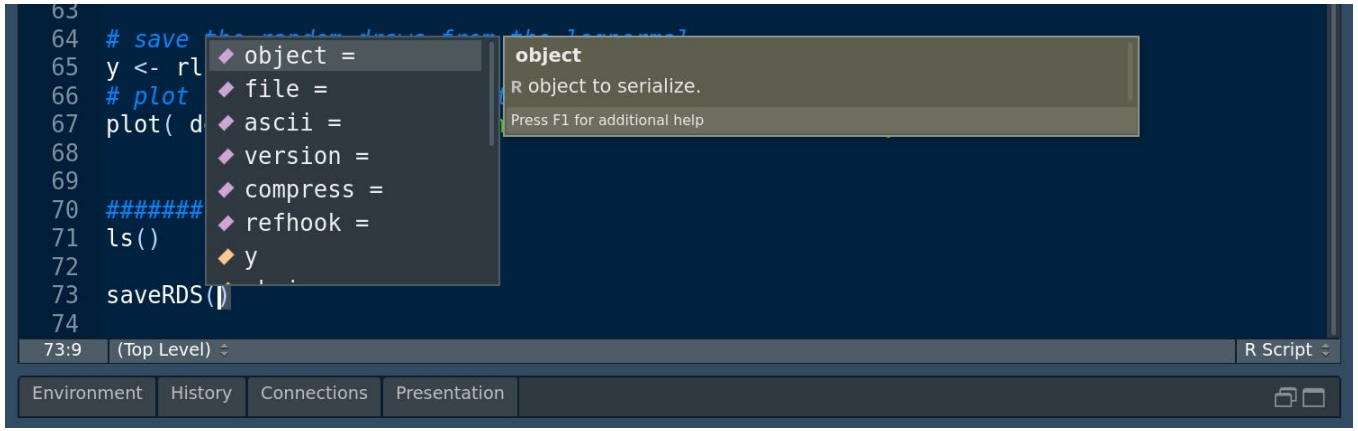
- What class of objects are these?

Input and Output into R

- You can write to different kinds of files



- And save R objects to reload later



HOT TIP

Want to look at the help file for a particular function? Search in the command line directly with, e.g.,

```
>?write.table
```

Looking for a function?

```
>??write
```

Plotting and visuals in RStudio

- "base" R has generic plotting functions, and they are customizable and *object-oriented*

```
# default, simplest plot of x and y data
plot(mydata$V1, mydata$V2)
```

Plotting and visuals in RStudio

- "base" R has generic plotting functions, and they are customizable and *object-oriented*

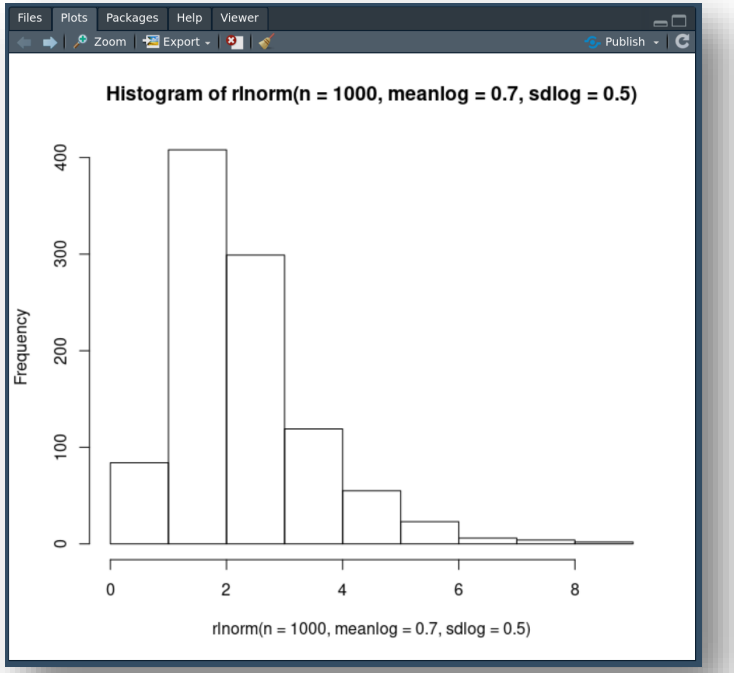
```
# default, simplest plot of x and y data
plot(mydata$V1, mydata$V2)
```

```
# make it a little prettier with some options
par(mar=c(5,5,3,3)) # bottom left top right margins
plot(mydata$V1, mydata$V2, xlab="x data", ylab="f(x)
data", cex.lab=1.7, type="l")
grid()
box()
```

Plotting and visuals in RStudio

- basic histogram

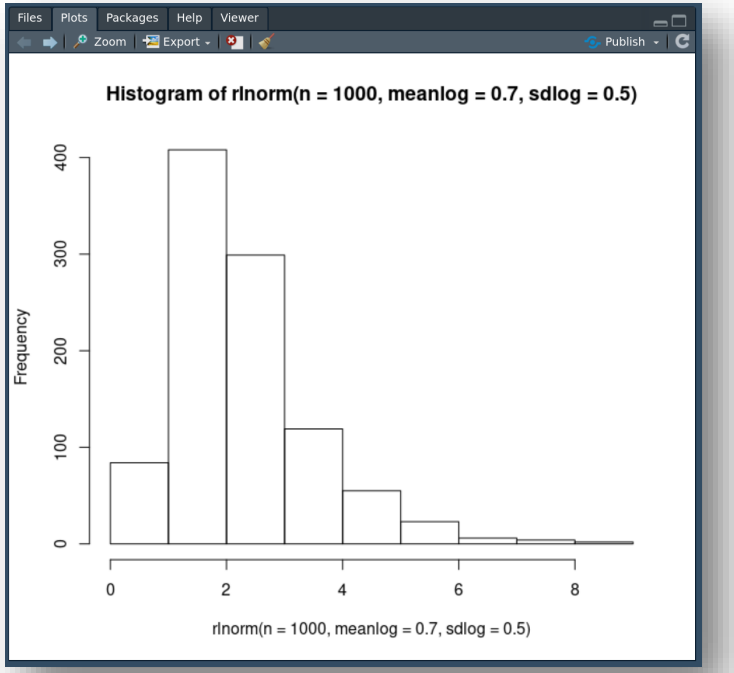
```
46  # default simple histogram
47  hist(rlnorm(n = 1e3, meanlog = 0.7, sdlog = 0.5))
48
49  # simple histogram where we set the bin size
50  hist(rlnorm(n = 1e3, meanlog = 0.7, sdlog = 0.5), breaks = 15)
51
```

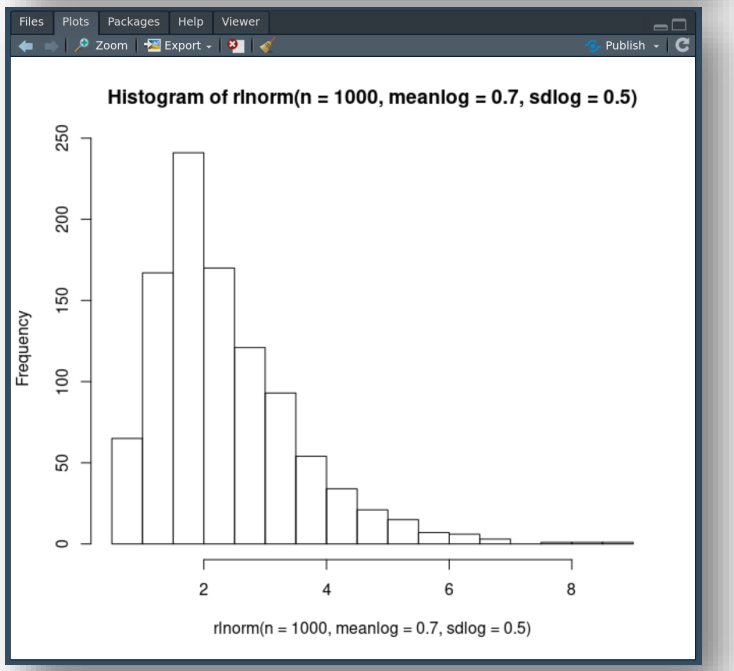


Plotting and visuals in RStudio

- basic histogram

```
46  # default simple histogram
47  hist(rlnorm(n = 1e3, meanlog = 0.7, sdlog = 0.5))
48
49  # simple histogram where we set the bin size
50  hist(rlnorm(n = 1e3, meanlog = 0.7, sdlog = 0.5), breaks = 15)
51
```

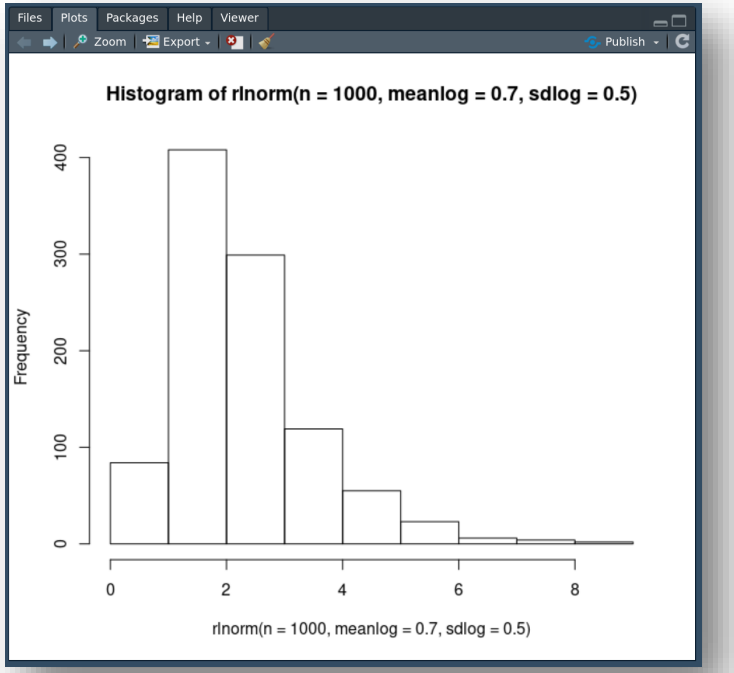




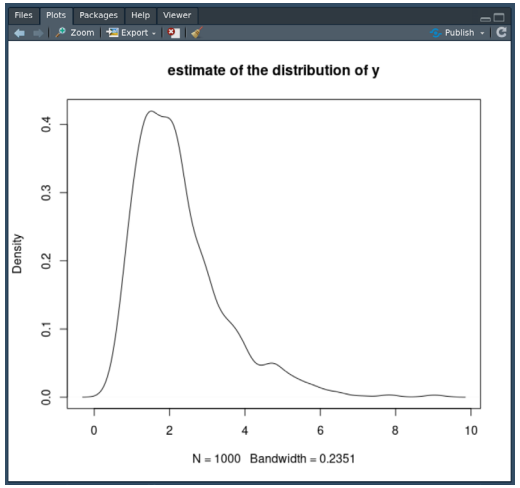
Plotting and visuals in RStudio

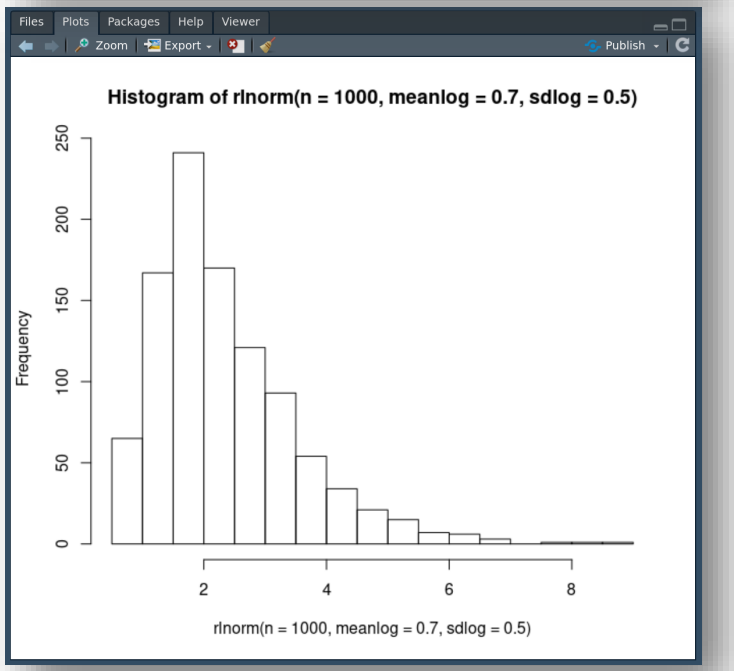
- basic histogram

```
46  # default simple histogram
47  hist(rlnorm(n = 1e3, meanlog = 0.7, sdlog = 0.5))
48
49  # simple histogram where we set the bin size
50  hist(rlnorm(n = 1e3, meanlog = 0.7, sdlog = 0.5), breaks = 15)
51
```



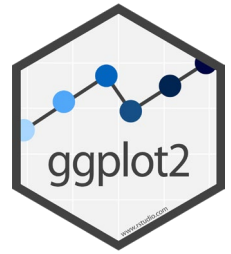

```
52  # save the random draws from the lognormal
53  y <- rlnorm(n = 1e3, meanlog = 0.7, sdlog = 0.5)
54  # plot the kernal density estimate of the random y values
55  plot( density(y), main="estimate of the distribution of y" )
```





- *Object-oriented programming!*

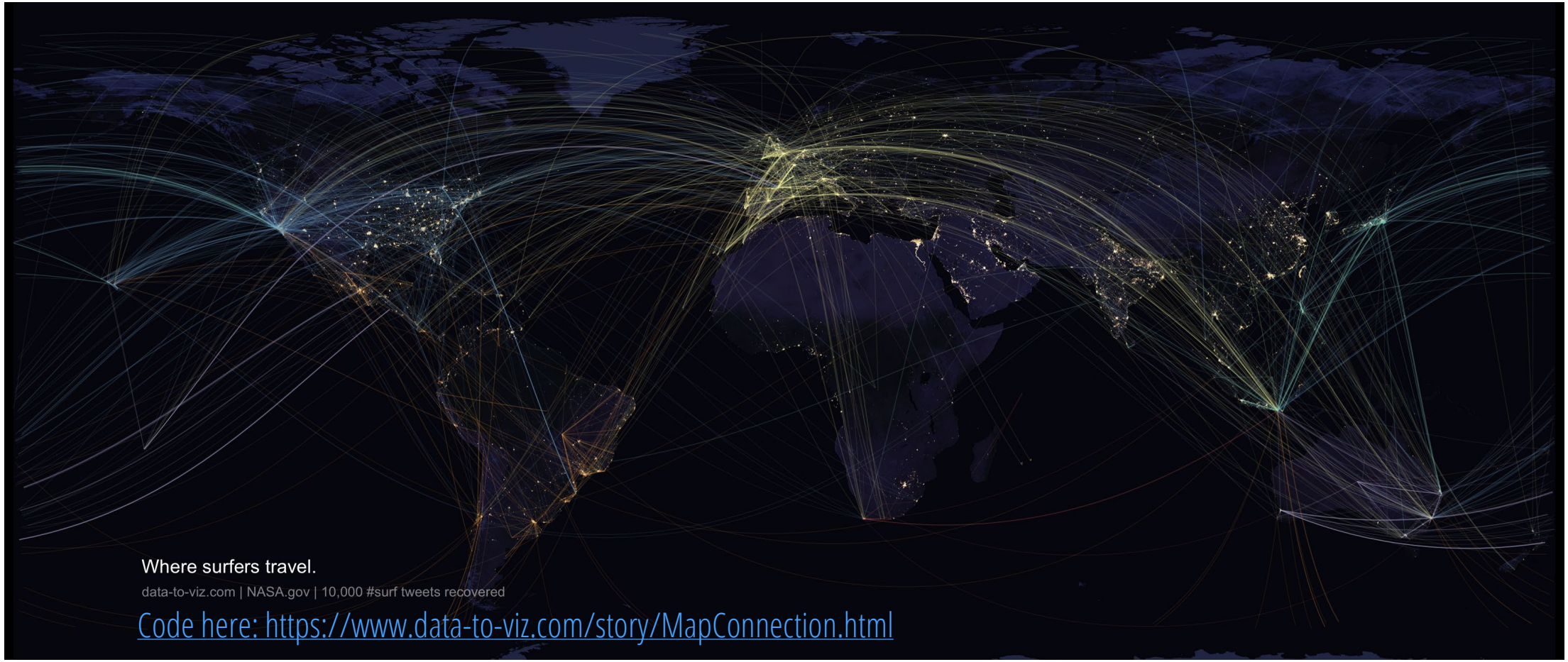
ggplot2



- The *ggplot2* package is extremely popular for data visualization
  - (you've probably seen plots made in R with ggplot2 without knowing it!)

ggplot2

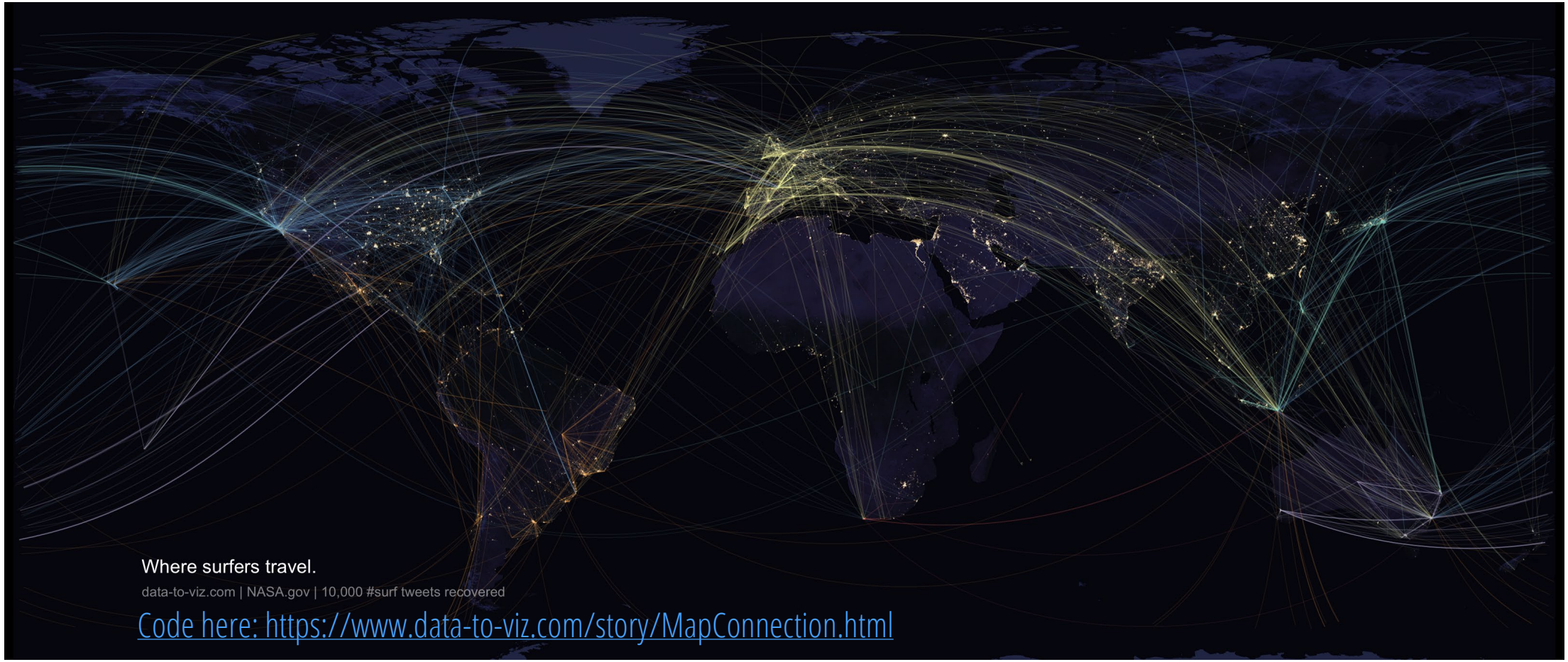
- The *ggplot2* package is extremely popular for data visualization
  - (you've probably seen plots made in R with ggplot2 without knowing it!)



ggplot2

Demo Time!
Super simple ggplot2 plot

- The *ggplot2* package is extremely popular for data visualization
  - (you've probably seen plots made in R with ggplot2 without knowing it!)



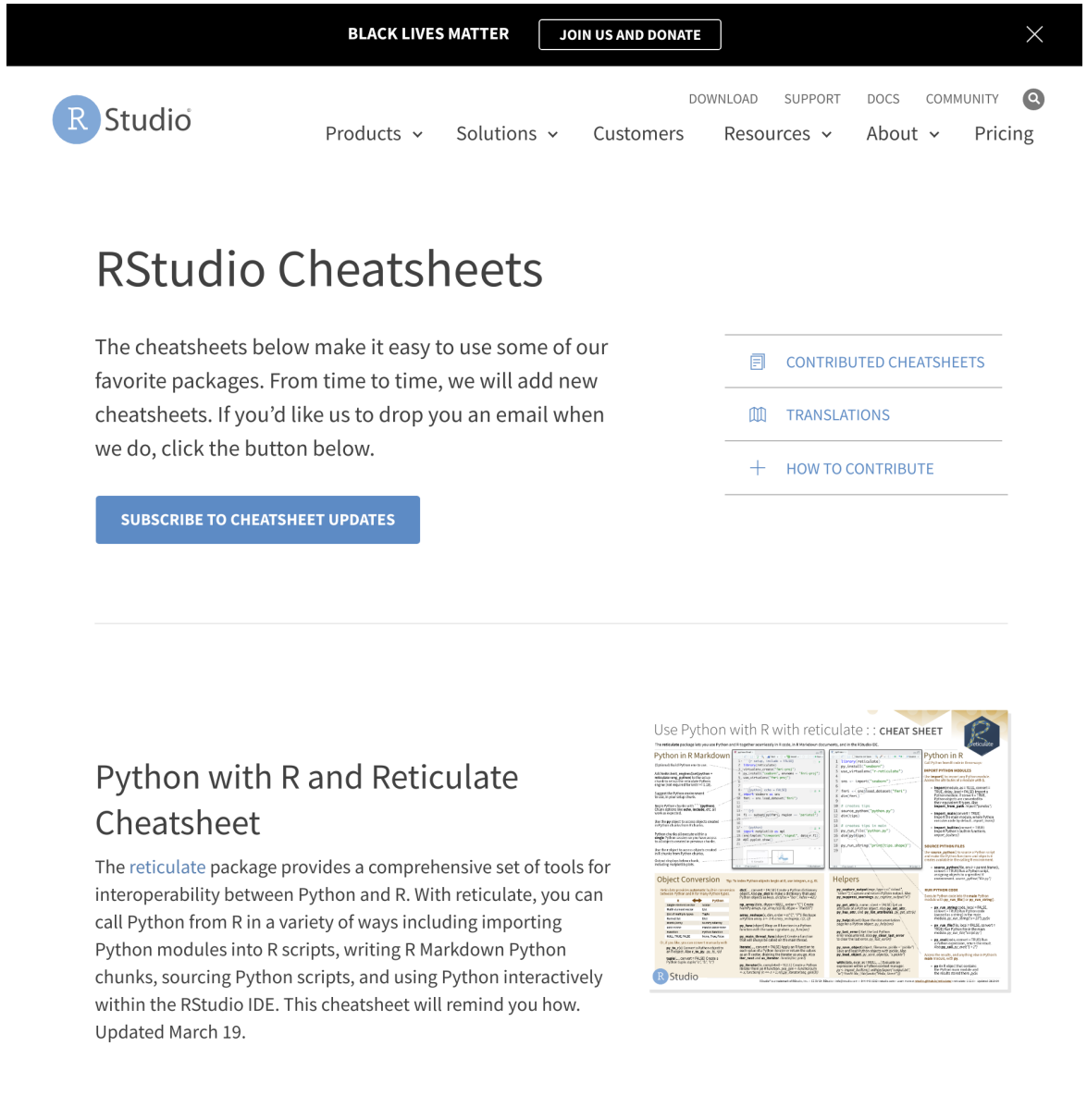
Code here: https://www.data-to-viz.com/story/MapConnection.html

RStudio Cheatsheets

https://rstudio.com/resources/cheatsheets/

HOT TIP

Check out the ones for Data Visualisation (ggplot2), Reticulate, Shiny Apps, and more



Exercise 1

- Write a function (in Python and R) that takes as input your height (**L0**), and computes your height **L** if you were travelling at **v = 0.3c**

- You may need to know the following equations for Lorentz contraction

$$L = L_0 / \gamma(v)$$

$$\gamma(v) \equiv \frac{1}{\sqrt{1 - v^2/c^2}}$$

COOL CATCH

Don't forget that an input string will have a different type to a float or integer.

Exercise 2

- Generate 100 random numbers from a Gaussian distribution

- Generate 100000 random numbers from a Gaussian distribution and show that they are much better approximated by a Gaussian distribution (i.e. plot a Gaussian curve)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

HINT: In R, look up `?rnorm`

HOT TIP

Here is where you'll really see the difference between R and Python

Exercise

Work in groups to complete this exercise:

- *In Python*
  - Read in the data from the data file A provided
  - Change the precision of the data file (reduce the floating point precision to 1 decimal place, or make them integers)
  - Apply a 50% Gaussian scatter to the data points
  - Make a plot of your data and save it to file B
  - Save the data to a text file
- *In R*
  - Read in that text file B that you saved in the step above
  - Compute the summary statistics of the data
  - Plot the data
  - Plot the five number summary statistics (hint: look up `?boxplot`)
  - Read in the original data file A used in the step above
  - Compare the summary statistics of the A data with the B data
  - Make a plot and save it to file
  - Save original data A to file with increased precision

HOT TIP

For your stretch goal – make doc strings for this whole package!