Week 5: Practical Coding and Statistics in Astronomy

SESSION 1: SQL AND DEBUGGING

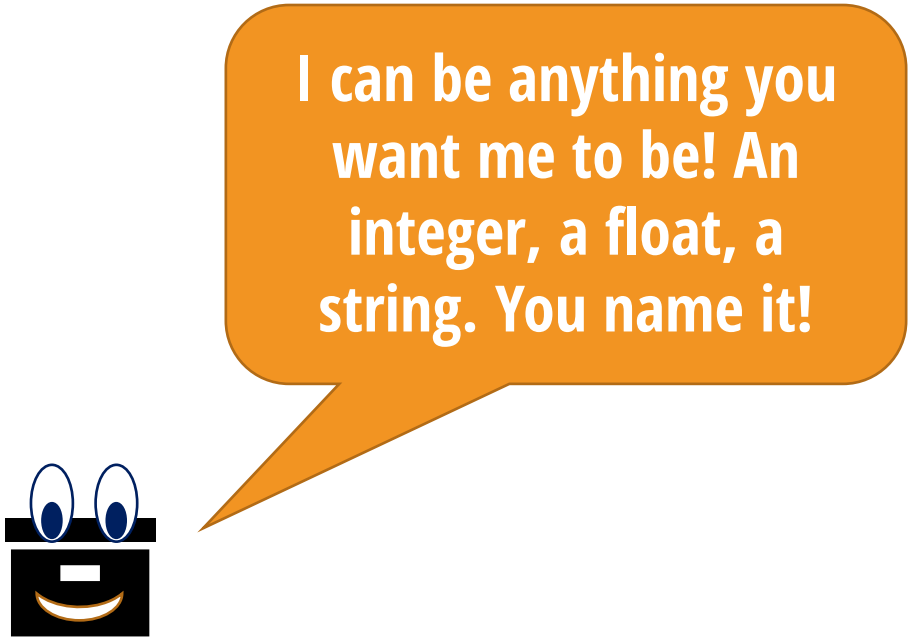
STARFISH SCHOOL 2021

Databases and their jargon

How do you solve a problem like data?

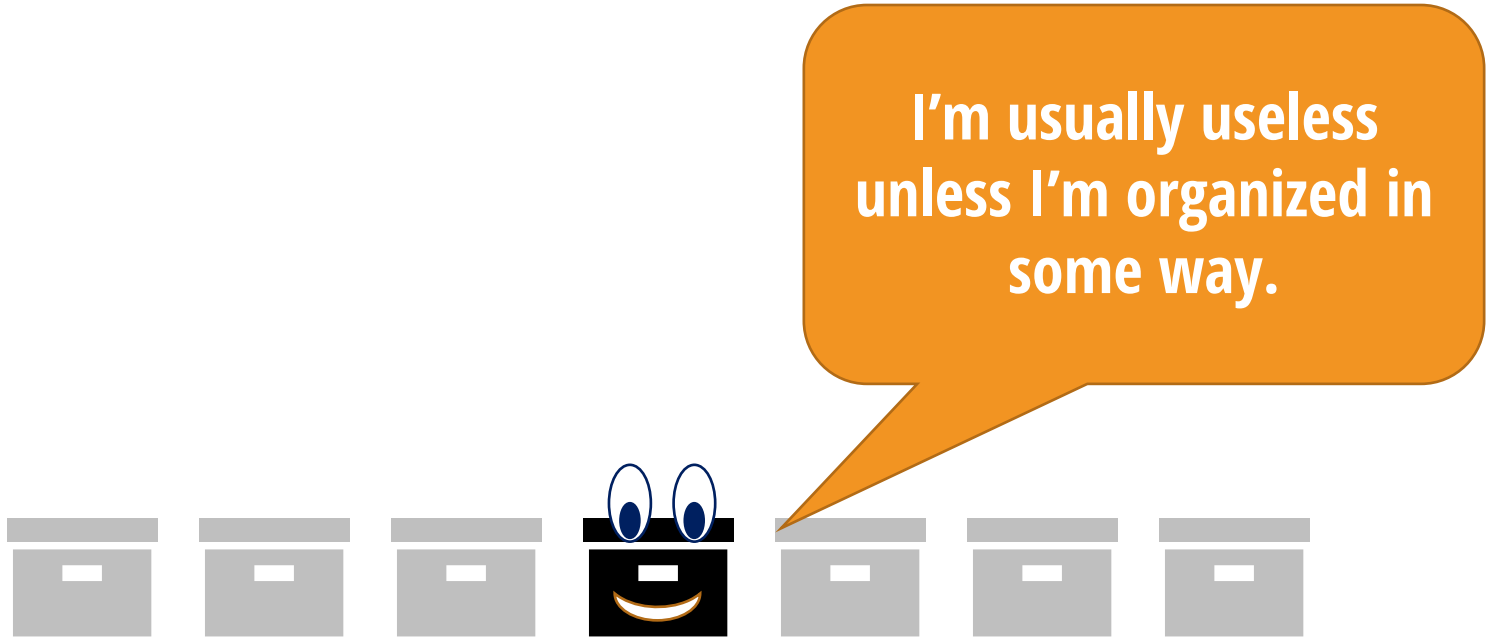
How do you solve a problem like data?

How do you solve a problem like data?

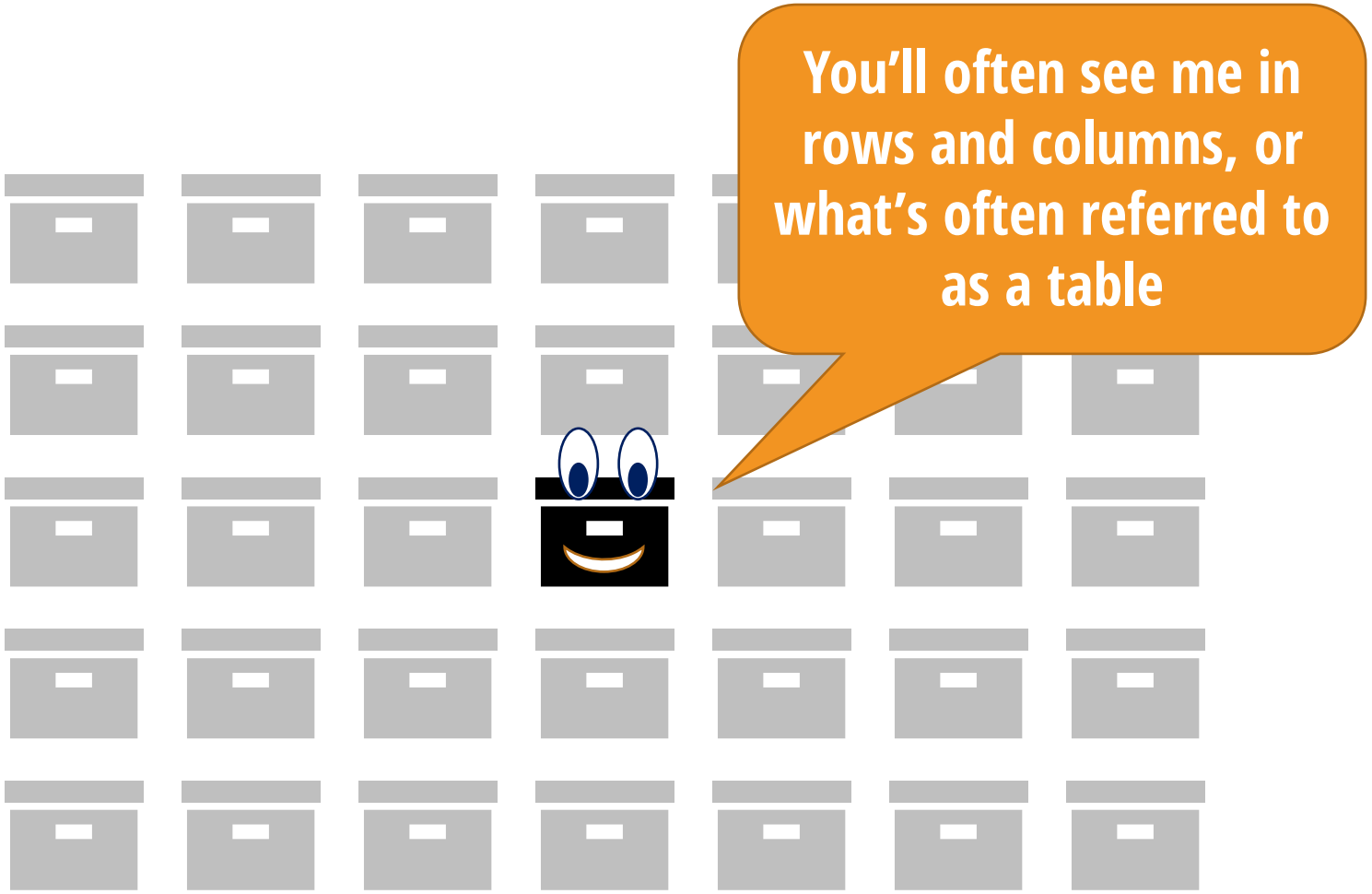


How do you solve a problem like data?

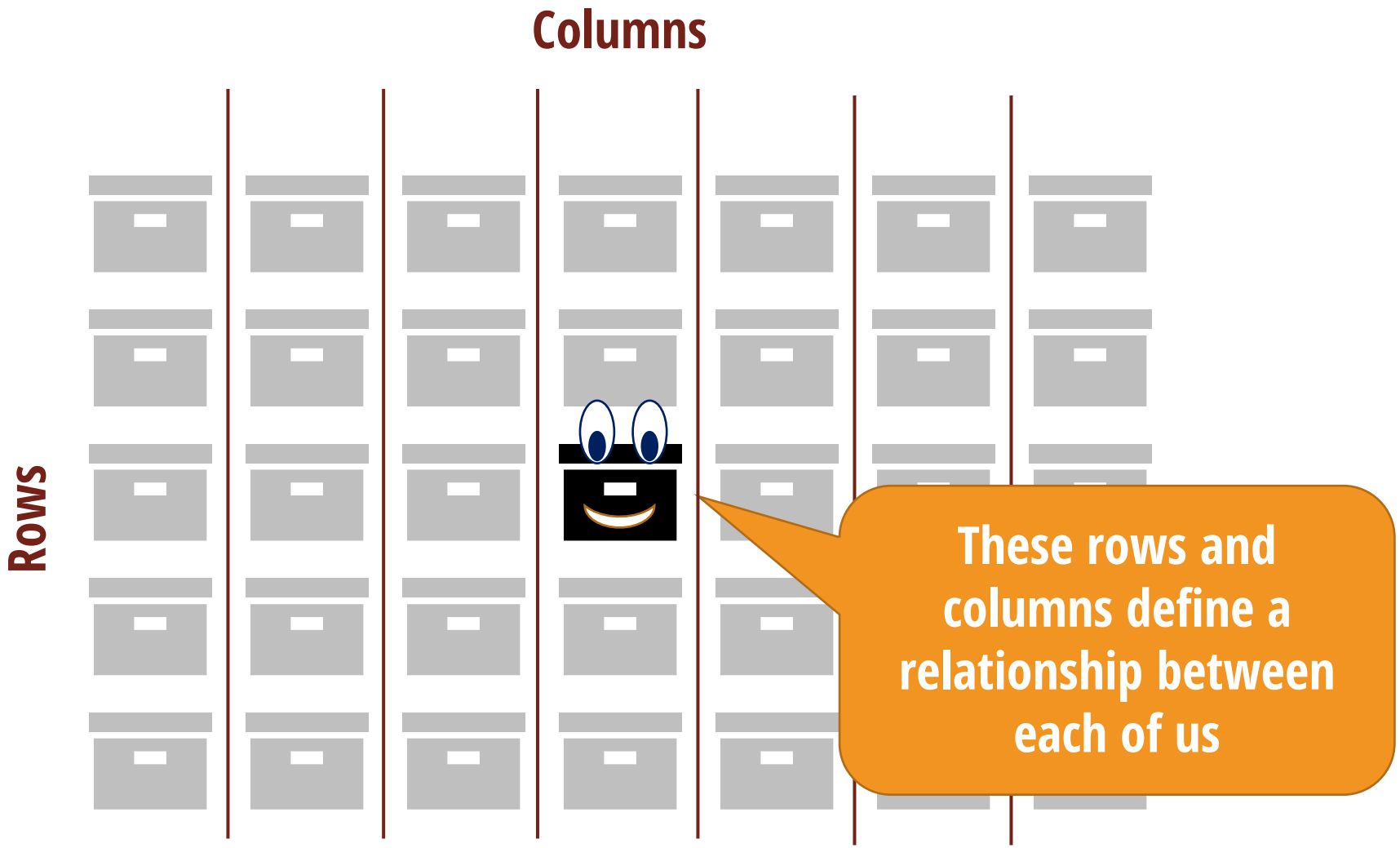
How do you solve a problem like data?



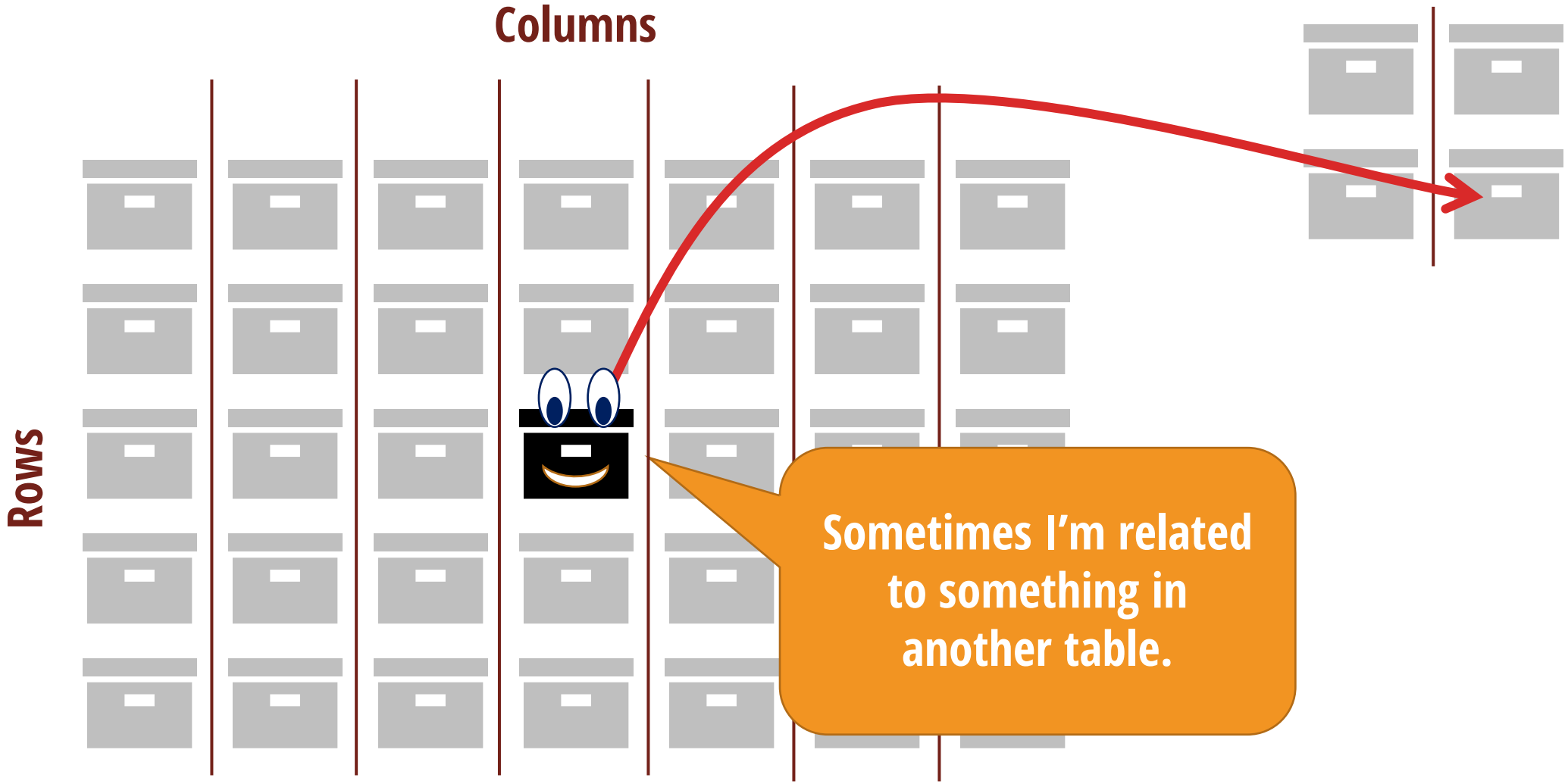
How do you solve a problem like data?



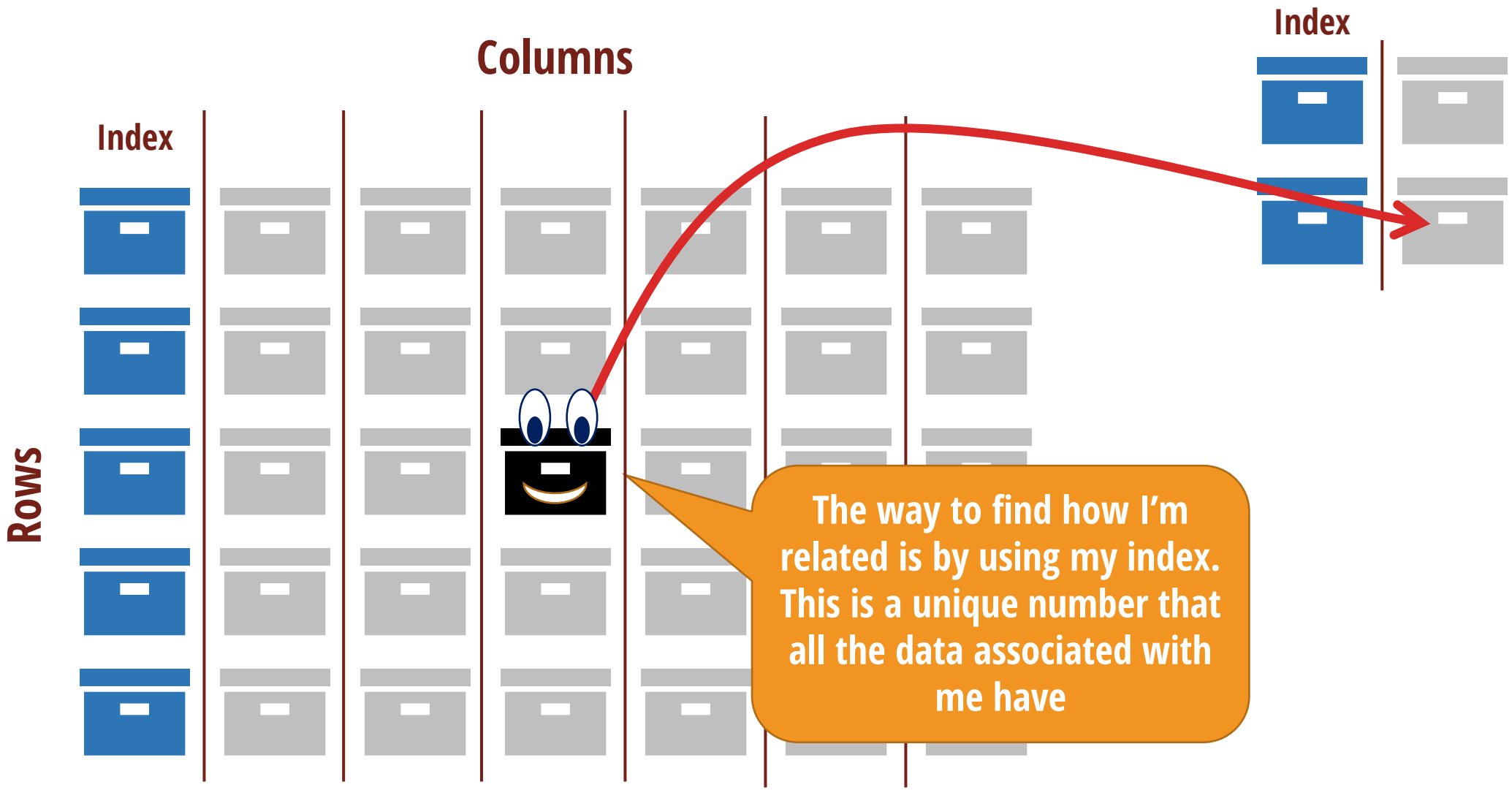
How do you solve a problem like data?



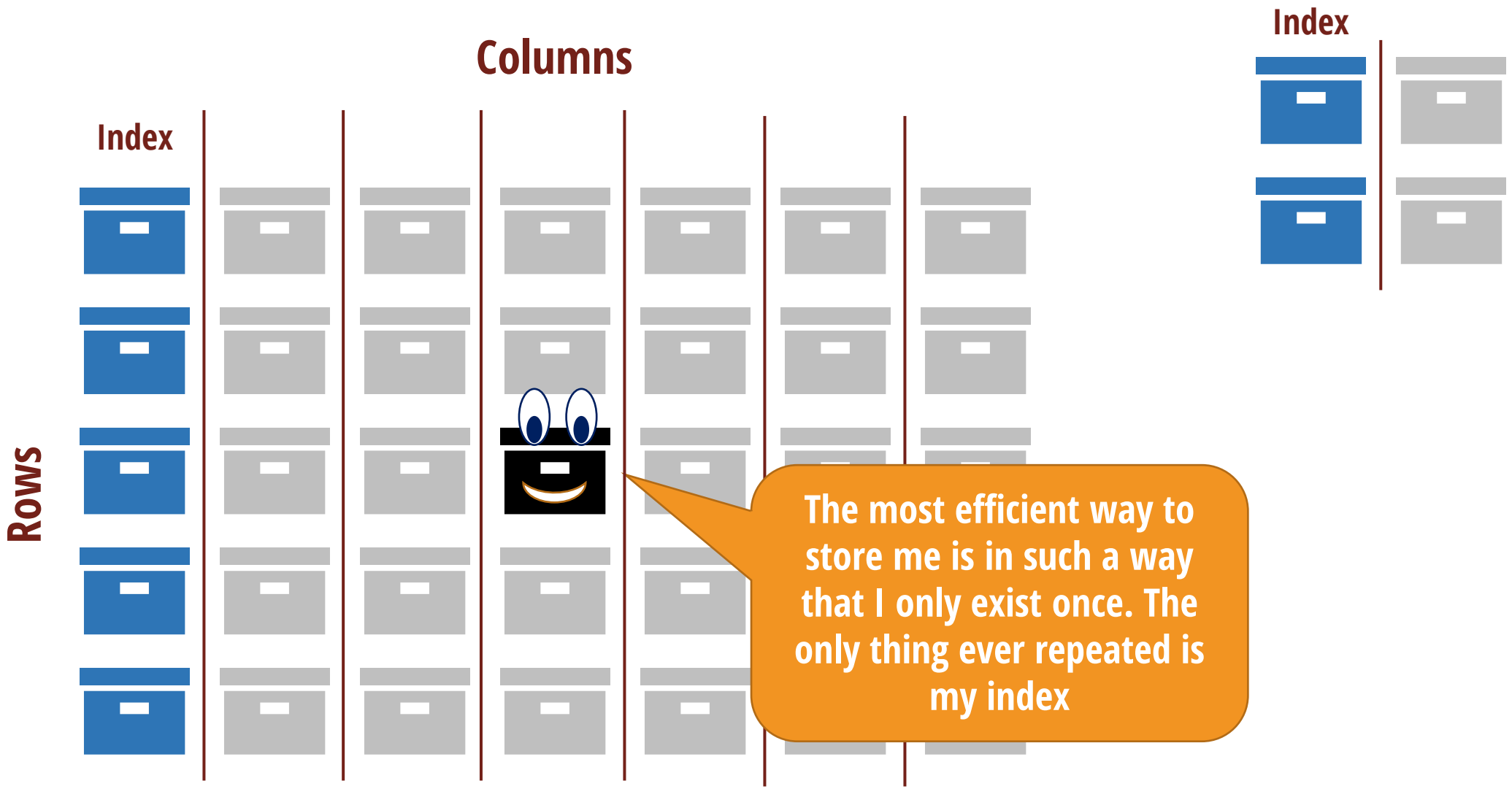
How do you solve a problem like data?



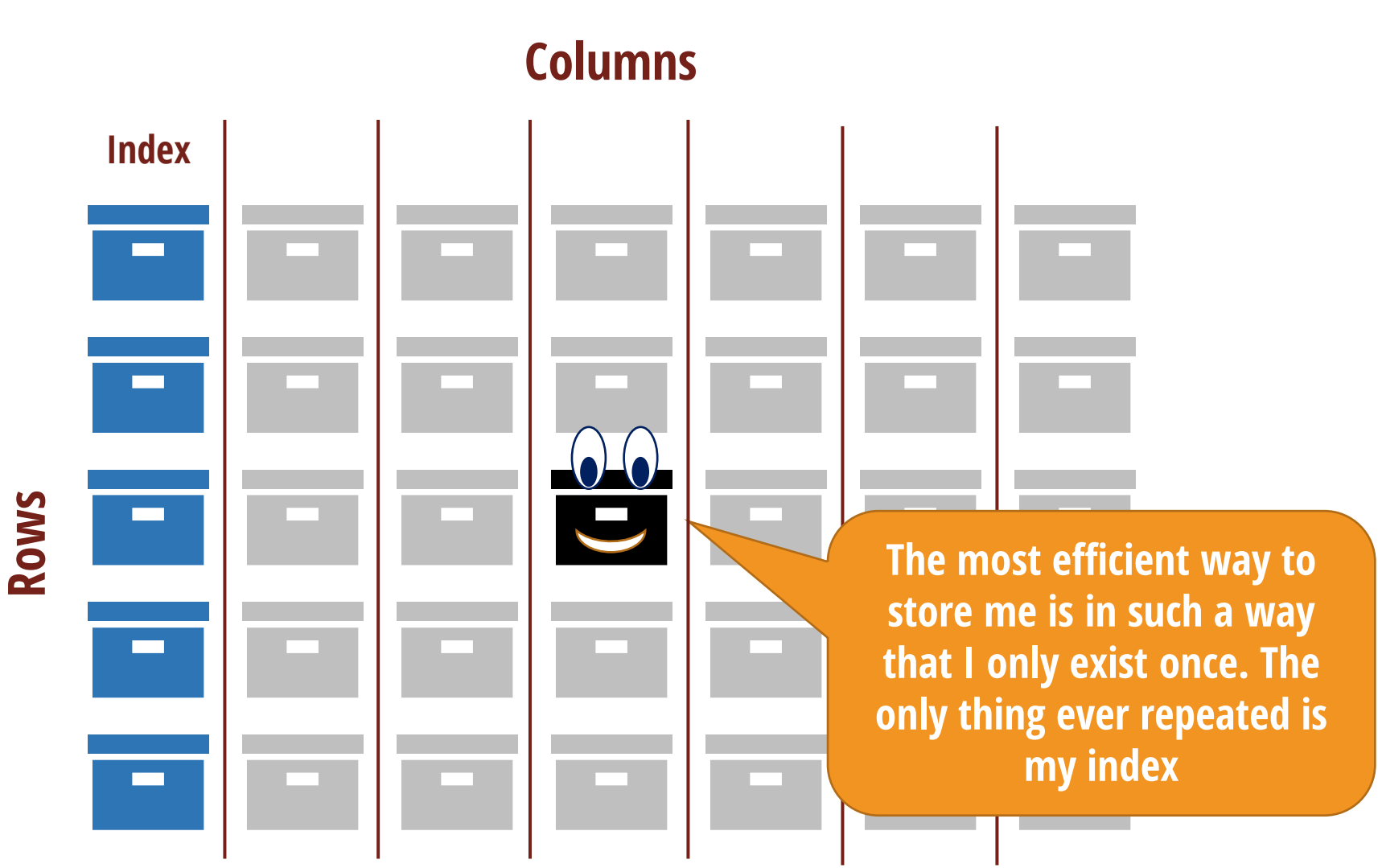
How do you solve a problem like data?

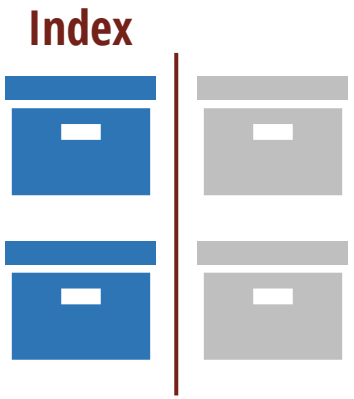


How do you solve a problem like data?



How do you solve a problem like data?





What we have here now is a database!

SQL

Structured Query Language say wha?

This is the programmatic way of dealing with data in a database!

- Moves a lot of the computing to where the data is. This way, you don't waste time
- Standardized (with some flavours). Generally, if you know the basics, you'll be able to figure out the rest.
- How much of the world works!

SELECTing Your Poison

The SELECT command is how to grab data from the database. The simplest query of all:

```
select * from TABLENAME
```

This command gets you everything from the table TABLENAME.

SELECTing Your Poison

The SELECT command is how to grab data from the database. The simplest query of all:

```
select * from TABLENAME
```

This command gets you everything from the table TABLENAME.

HOT TIP

When you have a large table, you might not want to do this initially. A quick way to make sure your query is working is using "top 10" to limit what it brings back

SELECTing Your Poison

The SELECT command is how to grab data from the database. The simplest query of all:

```
select col1, col2 from TABLENAME
```

This command gets you col1 and col2 from the table TABLENAME.

SELECTing Your Poison

The SELECT command is how to grab data from the database. The simplest query of all:

```
select col1 as a, col2 as b from TABLENAME
```

This command gets you col1 and col2 from the table TABLENAME, renaming them “a” and “b” respectively.

SELECTing Your Poison

The SELECT command is how to grab data from the database. The simplest query of all:

```
select col1 as a, col2 as b from TABLENAME
```

This command gets you col1 and col2 from the table TABLENAME, renaming them “a” and “b” respectively.

COOL CATCH

When we start talking about joining multiple tables, you may have multiple columns called the same thing from different tables. Renaming this using the “as” keyword helps you get around this.

SELECTing Your Poison

The SELECT command is how to grab data from the database. The simplest query of all:

```
select t.col1 as a, t.col2 as b
from TABLENAME as t
```

This command gets you col1 and col2 from the table TABLENAME which we refer to as “t”, renaming them “a” and “b” respectively.

SCHEMA scheema!

Where the details of how the database is structured are. Will tell you what tables there are, what columns, and what variable types they are.

Schema Browser

TABLE **TwoMassXSC**

2MASS extended-source catalog quantities for matches to SDSS hotometry

is table contains one entry for each match between the SDSS photometric catalog otoObjAll) and the 2MASS extended-source catalog (XSC). See http://tdc-w.harvard.edu/catalogs/tmx.format.html for full documentation.

e	type	length	unit	ucd	description
	bigint	8			Unique SDSS identifier composed from [skyVersion,rerun,run,camcol,field,obj].
sxsc_ra	float	8	deg		2MASS right ascension, J2000
sxsc_dec	float	8	deg		2MASS declination, J2000
e	float	8			Julian Date of the source measurement accurate to +30 seconds. (See 2MASS PSC documentation).
ignation	varchar	100			Sexagesimal, equatorial position-based source nam in the form: hhmmssss+ddmmsss[ABC...].
p_ra	float	8			Super-coadd centroid RA (J2000 decimal deg).
sup_dec	float	8			Super-coadd centroid Dec (J2000 decimal deg).
density	real	4			Coadd log(density) of stars with k<14 mag.
R_K20FE	real	4	mag		20mag/sq arcsec isophotal K fiducial ell. ap. semi-major axis.
J_M_K20FE	real	4	mag		J 20mag/sq arcsec isophotal fiducial ell. ap. magnitude.

SCHEMA scheema!

Where the details of how the database is structured are. Will tell you what tables there are, what columns, and what variable types they are.

HOT TIP

Always find the schema when you start making your query

Schema Browser

TABLE **TwoMassXSC**

2MASS extended-source catalog quantities for matches to SDSS hotometry

is table contains one entry for each match between the SDSS photometric catalog otoObjAll) and the 2MASS extended-source catalog (XSC). See http://tdc-w.harvard.edu/catalogs/tmx.format.html for full documentation.

e	type	length	unit	ucd	description
	bigint	8			Unique SDSS identifier composed from [skyVersion,rerun,run,camcol,field,obj].
sxsc_ra	float	8	deg		2MASS right ascension, J2000
sxsc_dec	float	8	deg		2MASS declination, J2000
	float	8			Julian Date of the source measurement accurate to +30 seconds. (See 2MASS PSC documentation).
	varchar	100			Sexagesimal, equatorial position-based source nam in the form: hhmmssss+ddmmsss[ABC...].
	float	8			Super-coadd centroid RA (J2000 decimal deg).
	float	8			Super-coadd centroid Dec (J2000 decimal deg).
density	real	4			Coadd log(density) of stars with k<14 mag.
R_K20FE	real	4	mag		20mag/sq arcsec isophotal K fiducial ell. ap. semi-major axis.
J_M_K20FE	real	4	mag		J 20mag/sq arcsec isophotal fiducial ell. ap. magnitude.

WHERE is my data?

Just grabbing all of your data, or all of the selected columns probably isn't what you want to do. You'll want to select some limited set of data with the WHERE conditional:

```
select col1, col2 from TABLENAME
WHERE col1 > 2
```

Grab col1 and col2 from TABLENAME where the value in col1 is greater than 2.

WHERE is my data?

Just grabbing all of your data, or all of the selected columns probably isn't what you want to do. You'll want to select some limited set of data with the WHERE conditional:

```
select col1, col2 from TABLENAME
WHERE col1 = 2
```

Grab col1 and col2 from TABLENAME where the value in col1 is equal to 2.

WHERE is my data?

Just grabbing all of your data, or all of the selected columns probably isn't what you want to do. You'll want to select some limited set of data with the WHERE conditional:

```
select col1, col2 from TABLENAME
WHERE col1 = 2
```

Grab col1 and col2 from TABLENAME where the value in col1 is equal to 2.

COOL CATCH

In SQL, notice that for "is equals", you use a single equal sign, rather than the `==` in Python.

WHERE is my data?

Just grabbing all of your data, or all of the selected columns probably isn't what you want to do. You'll want to select some limited set of data with the WHERE conditional:

```
select col1, col2 from TABLENAME
  WHERE (col1 = 2 and col2 > 5)
```

Grab col1 and col2 from TABLENAME where the value in col1 is equal to 2 and col2 is greater than 5.

WHERE is my data?

Just grabbing all of your data, or all of the selected columns probably isn't what you want to do. You'll want to select some limited set of data with the WHERE conditional:

```
select col1, col2 from TABLENAME
  WHERE (col1 = 2 and col2 > 5)
```

Grab col1 and col2 from TABLENAME where the value in col1 is equal to 2 and col2 is greater than 5.

COOL CATCH

In SQL, logical operators use the words "and", "or", or "not" – not the symbols.

JOINing together

The real power from databases comes from taking information from multiple tables and processing it together. Let's say there's table1 and table2 that have different data, but both have the unique identifier in a column called "ID", you can get all data merged from the two tables:

```
select a.col1, a.col2, b.col3, b.col4 from
table1 as a
JOIN table2 as b on a.ID = b.ID
```

JOINing together

The real power from databases comes from taking information from multiple tables and processing it together. Let's say there's table1 and table2 that have different data, but both have the unique identifier in a column called "ID", you can get all data merged from the two tables:

```
select a.col1, a,col2, b.col3, b.col4 from
table1 as a
JOIN table2 as b on a.ID = b.ID
```

HOT TIP

Generally, when you join tables, you use a special column called a "Primary Key". It's a column where every entry is **unique** and **required**. All database tables have a primary key.

JOINing together

The real power from databases comes from taking information from multiple tables and processing it together. Let's say there's table1 and table2 that have different data, but both have the unique identifier in a column called "ID", you can get all data merged from the two tables:

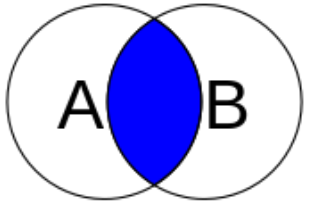
```
select a.col1, a,col2, b.col3, b.col4 from
table1 as a
JOIN table2 as b on a.ID = b.ID
```

COOL CATCH

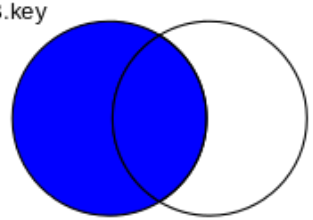
All of your joins should be done before your WHEREs.

SQL JOINS

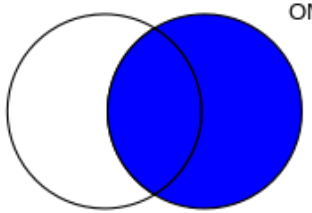
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



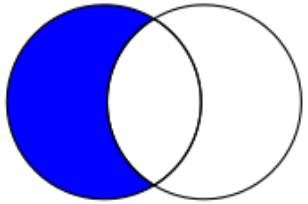
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key

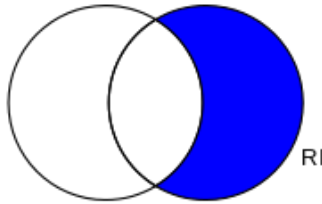


SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key

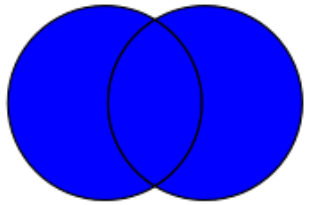


SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL

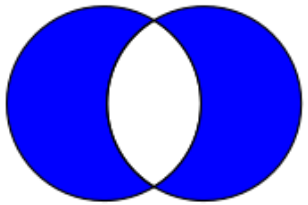




SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE a.key IS NULL



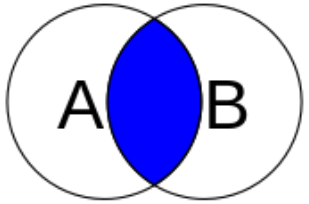
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key



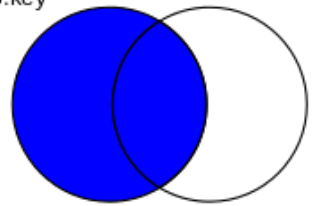
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key iIS NULL

SQL JOINS

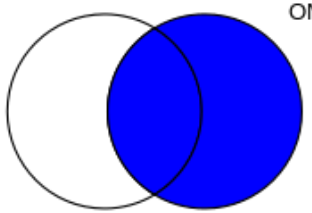
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



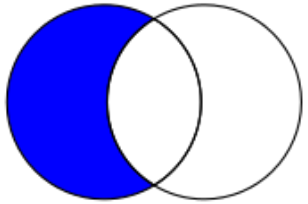
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key

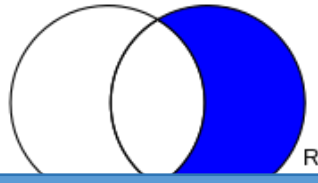


SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key



SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL

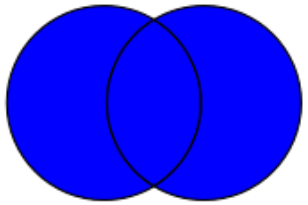




SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B

COOL CATCH

By default, all joins in SQL are inner joins



SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key

FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key iIS NULL

JOINing together

The real power from databases comes from taking information from multiple tables and processing it together. Let's say there's table1 and table2 that have different data, but both have the unique identifier in a column called "ID", you can get all data merged from the two tables:

```
select a.col1, a,col2, b.col3, b.col4 from
table1 as a
LEFT JOIN table2 as b on a.ID = b.ID
```

In this query, everything from table1 will be grabbed, even if there isn't an entry in table2.

Notebooks -> Production Code -> Packages

Why Notebooks aren't everything

- Great for exploration and discovery
- Can run/rerun things non-linearly
- Rapid development
- Plots, Markdown, Code all in the same place
- Difficult to Organize

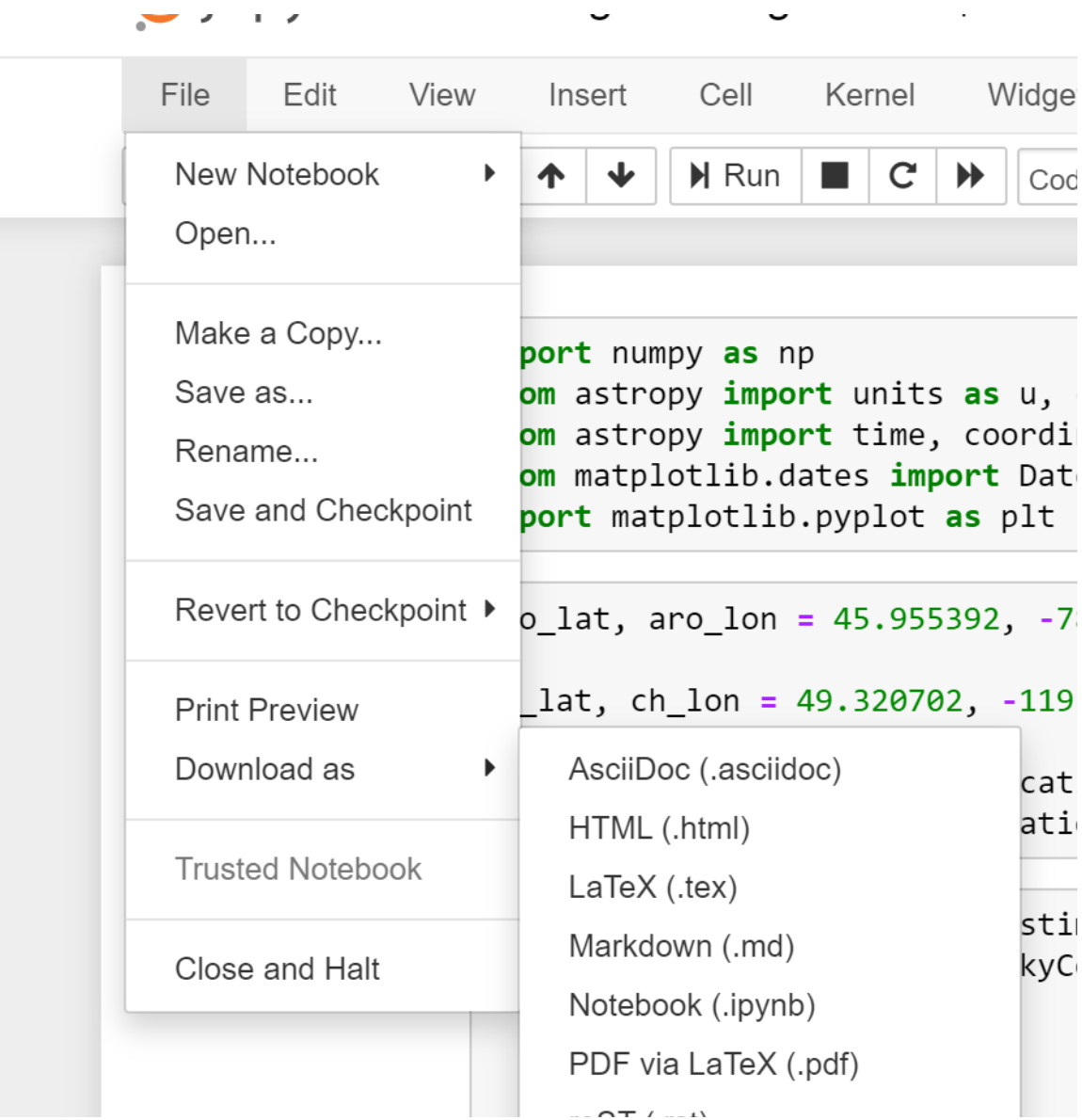
- Not reusable
- Difficult to Automate
- Missing debugging and advanced development tools
- Difficult to Scale

Converting between the two is an essential part of the development process

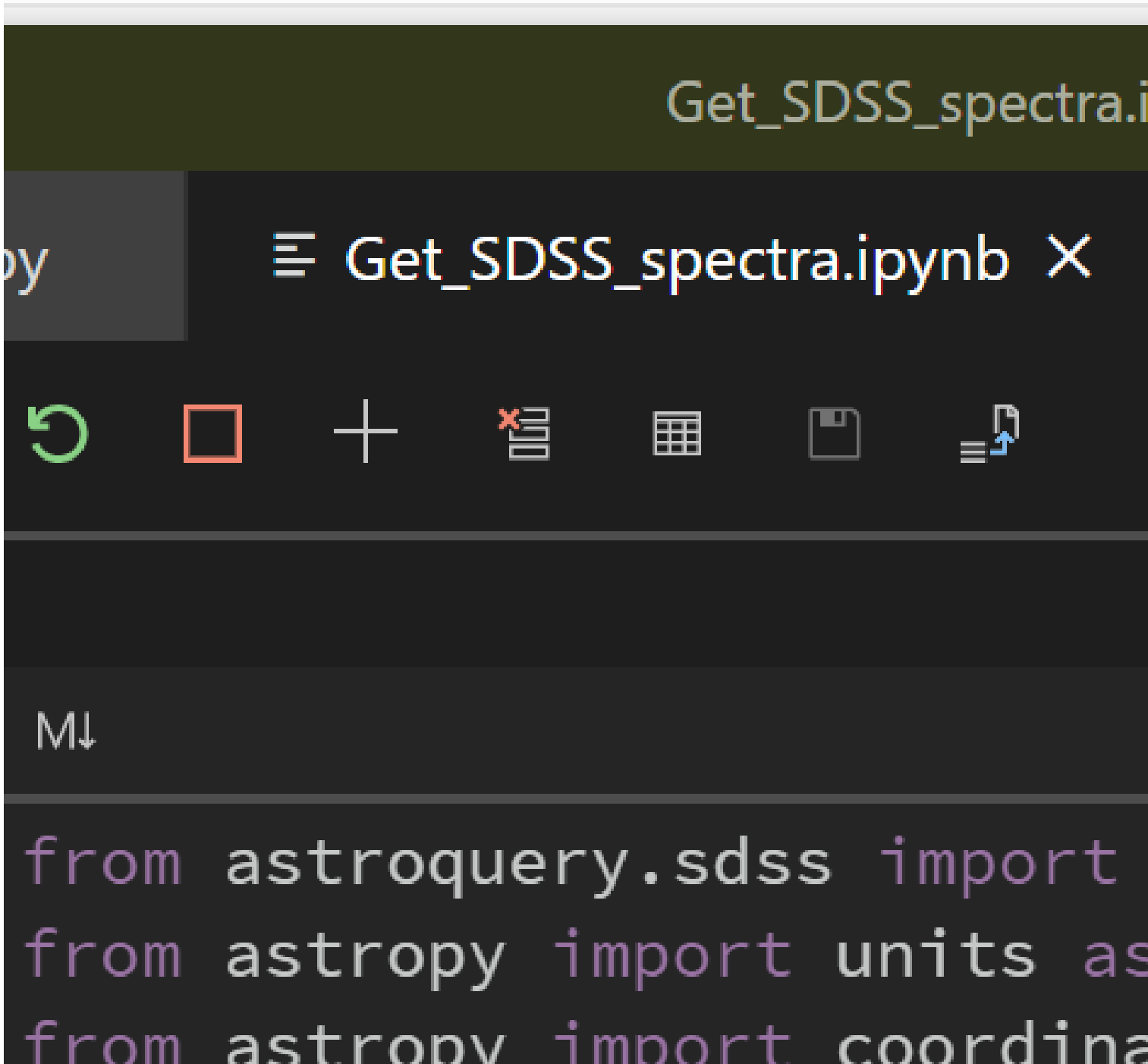
Getting a notebook ready for conversion

Rearrange	Rearrange Cells so that they run in order
Delete	Delete test/random prints
Move	Move all imports to the top
Merge	Merge cells that should be together
Separate	Separate function definitions from straight/sequential code
Clear	Clear variables and run all cells to make sure they run correctly without error

Converting a Notebook to a Script (via Jupyter)



Converting a Notebook to a Script (via VS Code)



Going from R-script to R markdown notebook

- Situation:
  - You wrote an R script to do some stuff and make a plot, and your supervisor/collaborator wants to understand it quickly.
  - You want to give them an easy-to-read "report" in html, pdf, etc.

- Solution:
  - install the knitr package in R via the R command line:
    `install.packages("knitr")`
  - In RStudio, this will enable some helpful icons to use the knitr package quickly.

Going from R-script to R markdown notebook

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

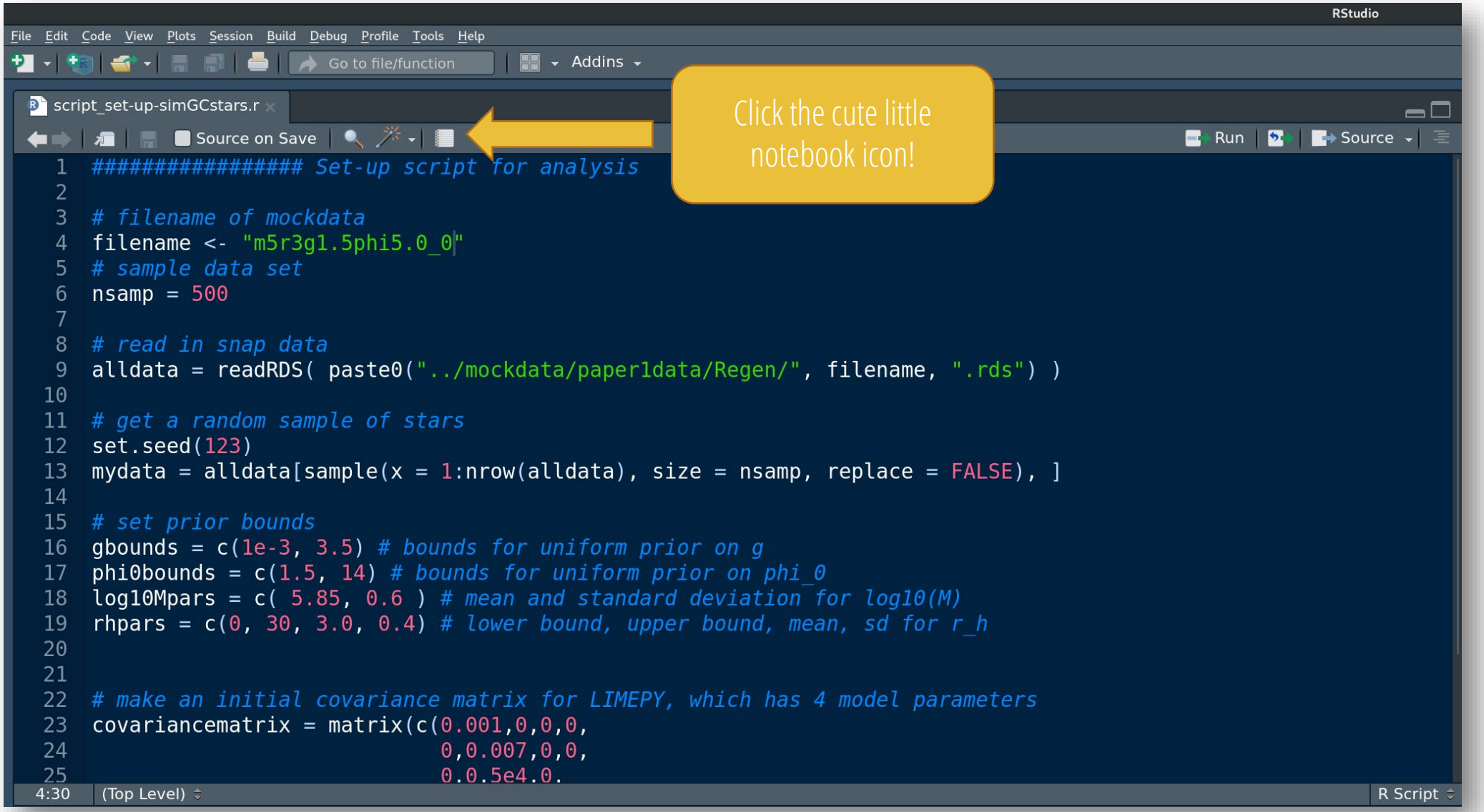
script_set-up-simGCstars.r

Source on Save Run Source

```r
################ Set-up script for analysis

# filename of mockdata
filename <- "m5r3g1.5phi5.0_0"
# sample data set
nsamp = 500

# read in snap data
alldata = readRDS( paste0("../mockdata/paper1data/Regen/", filename, ".rds") )

# get a random sample of stars
set.seed(123)
mydata = alldata[sample(x = 1:nrow(alldata), size = nsamp, replace = FALSE), ]

# set prior bounds
gbounds = c(1e-3, 3.5) # bounds for uniform prior on g
phi0bounds = c(1.5, 14) # bounds for uniform prior on phi_0
log10Mpars = c( 5.85, 0.6 ) # mean and standard deviation for log10(M)
rhpars = c(0, 30, 3.0, 0.4) # lower bound, upper bound, mean, sd for r_h


# make an initial covariance matrix for LIMEPY, which has 4 model parameters
covariancematrix = matrix(c(0.001,0,0,0,
                            0,0.007,0,0,
                            0,0,5e4,0,
```

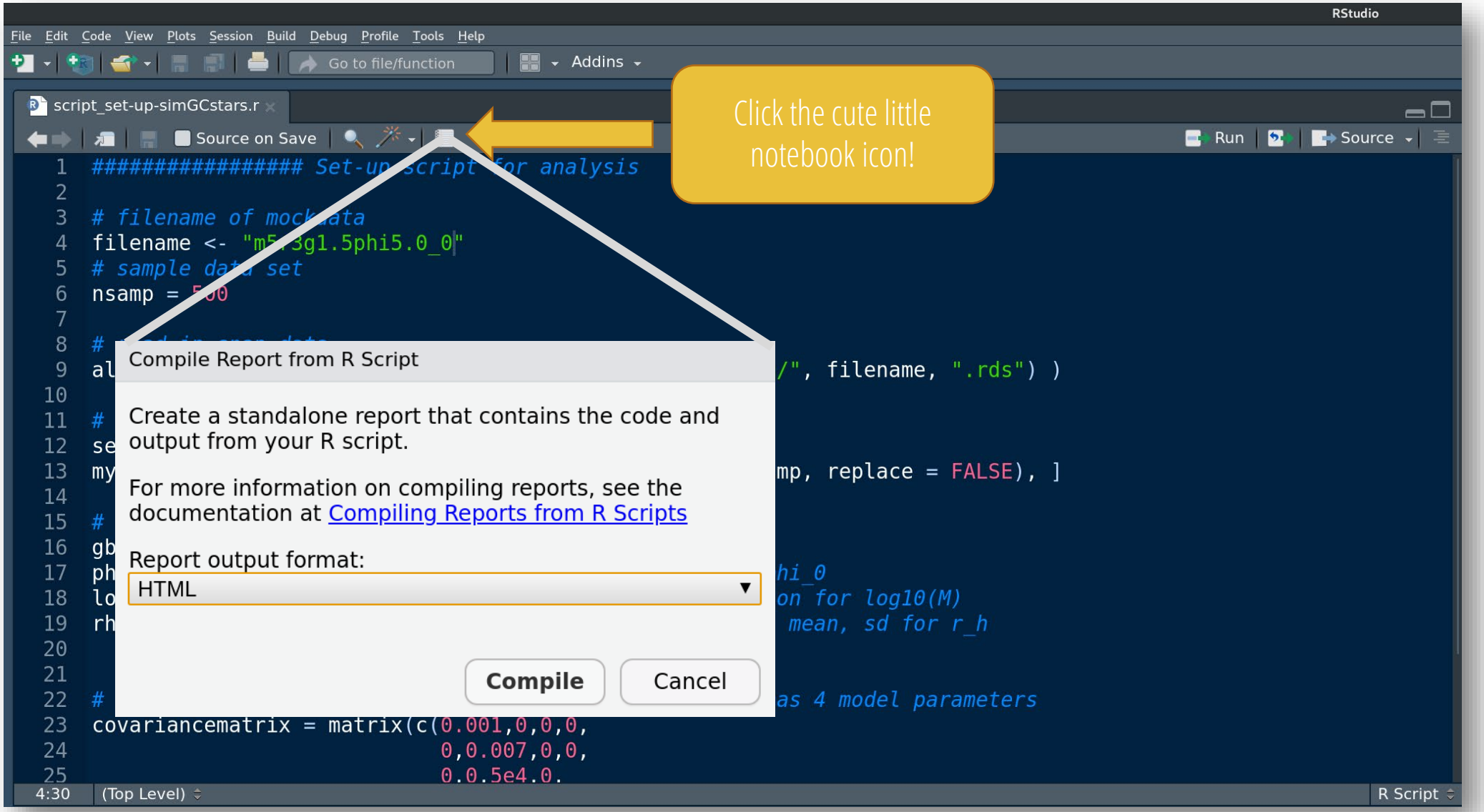
4:30 (Top Level) R Script

Going from R-script to R markdown notebook

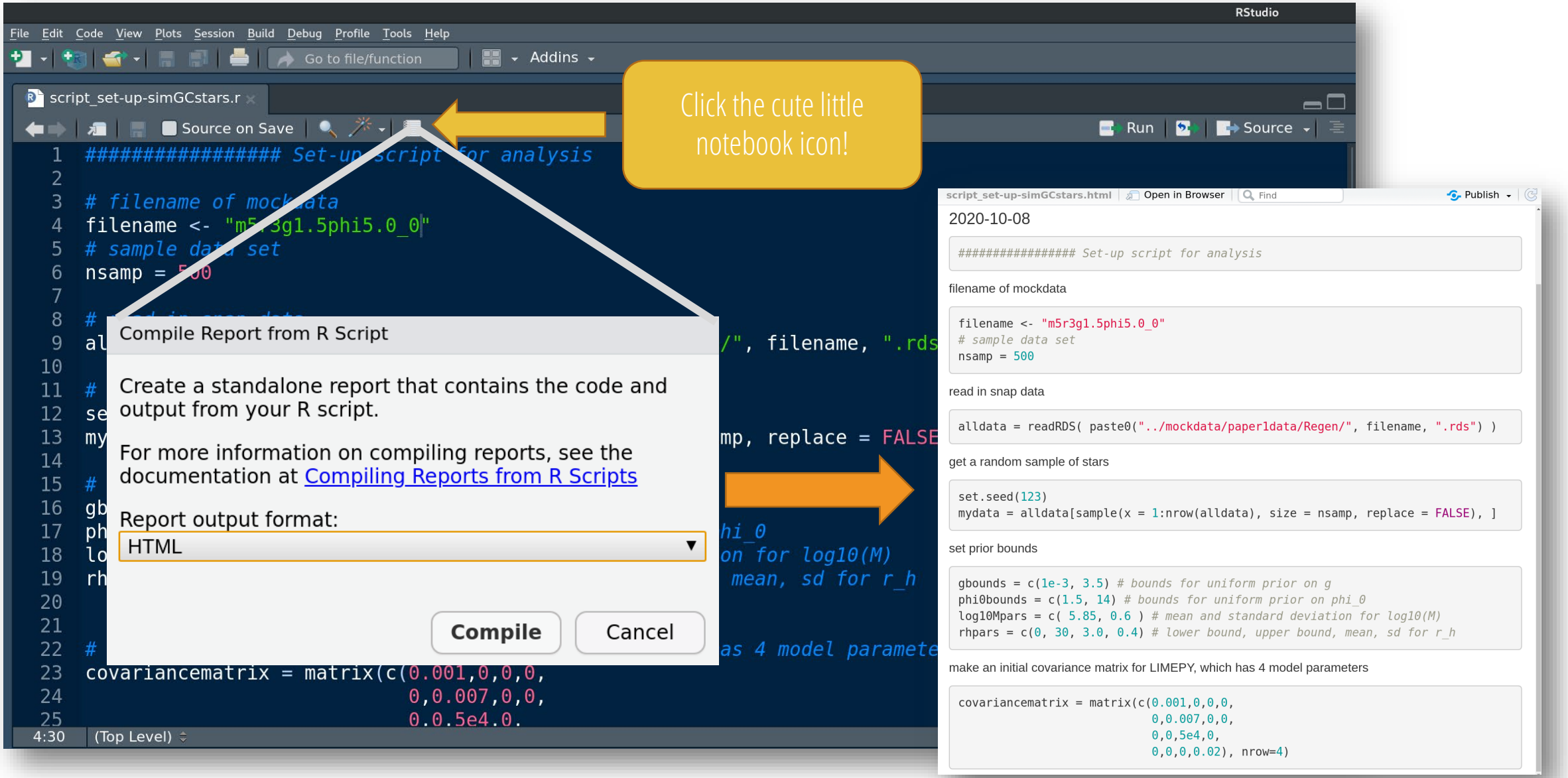



```
################ Set-up script for analysis

# filename of mockdata
filename <- "m5r3g1.5phi5.0_0"
# sample data set
nsamp = 500

# read in snap data
alldata = readRDS( paste0("../mockdata/paper1data/Regen/", filename, ".rds") )

# get a random sample of stars
set.seed(123)
mydata = alldata[sample(x = 1:nrow(alldata), size = nsamp, replace = FALSE), ]

# set prior bounds
gbounds = c(1e-3, 3.5) # bounds for uniform prior on g
phi0bounds = c(1.5, 14) # bounds for uniform prior on phi_0
log10Mpars = c( 5.85, 0.6 ) # mean and standard deviation for log10(M)
rhpars = c(0, 30, 3.0, 0.4) # lower bound, upper bound, mean, sd for r_h


# make an initial covariance matrix for LIMEPY, which has 4 model parameters
covariancematrix = matrix(c(0.001,0,0,0,
                            0,0.007,0,0,
                            0,0,5e4,0,
```

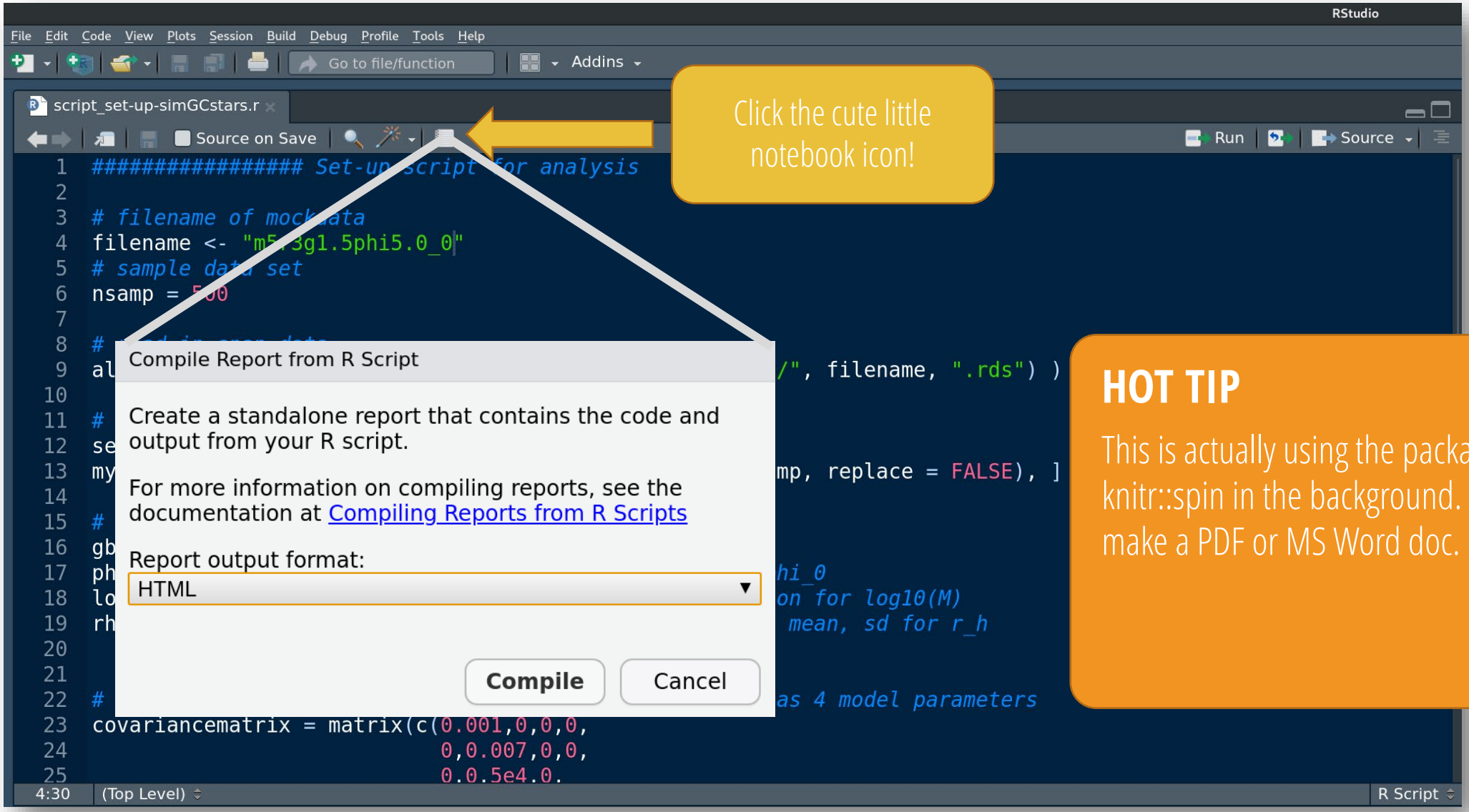
Going from R-script to R markdown notebook



Going from R-script to R markdown notebook



Going from R-script to R markdown notebook



Click the cute little notebook icon!

HOT TIP

This is actually using the package knitr::spin in the background. It can also make a PDF or MS Word doc.

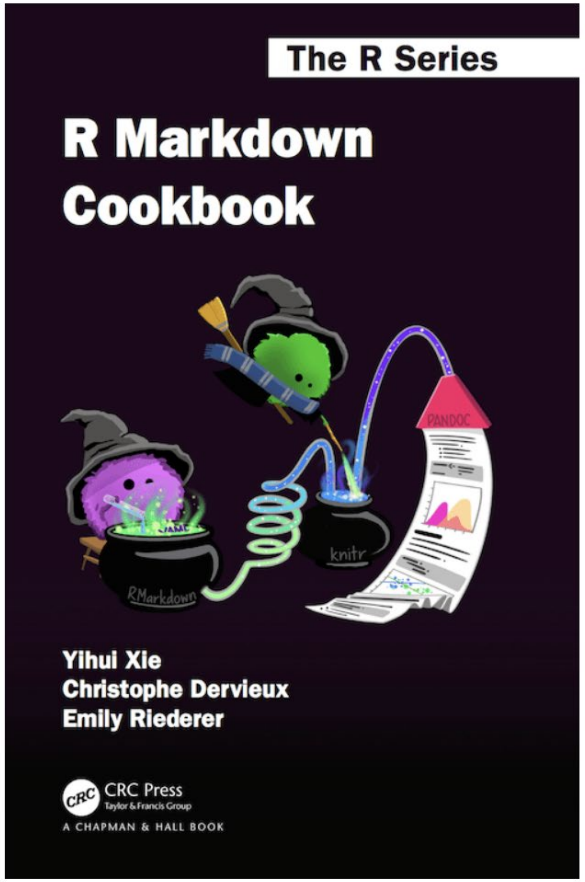
You can do the reverse too!

- Situation:
  - you wrote an R Markdown document while developing some code and sharing the steps with your supervisor/collaborators.
  - The code in the R Markdown doc is working great, so now you want to make only the code portions into R script(s) (maybe to prepare things for an R package!)

- Solution:
  - Use the function purl() in the knitr package (**knitr::purl**)
  - Purl grabs the R chunks from an R Markdown document, and puts these into an R Script

See the R Markdown Cookbook for more info:
https://bookdown.org/yihui/rmarkdown-cookbook/purl.html

Note: This book is to be published by Chapman & Hall/CRC. The online version of this book is free to read here (thanks to Chapman & Hall/CRC), and licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. If you have any feedback, please feel free to file an issue on GitHub. Thank you!



The R Markdown Cookbook

These are other helpful tips about R Markdown can be found here:

https://bookdown.org/yihui/rmarkdown-cookbook/

There are lots of other tutorials, discussions (stackoverflow, etc), available online. Google searching is your friend!

Packagifying your Code

What is a python package? Python code that you can import. A great way to re-use your functions.

Simplest possible python functions: a single python file!

Use __init__.py files to make your life easier.

Can build more complex structures to enable more complex features.

Jo has a full mini-course he's taught on the details of python processing: https://pythonpackaging.info/

Packagifying your Code in R

R packages

Packages are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sample data. In this book you'll learn how to turn your code into packages that others can easily download and use. Writing a package can seem overwhelming at first. So start with the basics and improve it over time. It doesn't matter if your first version isn't perfect as long as the next version is better.

This is where we are developing the 2nd edition of this book. The 1st edition remains available at http://r-pkgs.had.co.nz/.

https://r-pkgs.org/

Packagifying your Code in R

- There's a package for that! Check out **devtools** https://devtools.r-lib.org/
- **devtools** has a cheatsheet: https://rawgit.com/rstudio/cheatsheets/master/package-development.pdf

Package Development: : CHEAT SHEET

devtools

Package Structure

A package is a convention for organizing files into directories.

This sheet shows how to work with the 7 most common parts of an R package:

Package	
DESCRIPTION	SETUP
R/	WRITE CODE
tests/	TEST
man/	DOCUMENT
vignettes/	TEACH
data/	ADD DATA
NAMESPACE	ORGANIZE

The contents of a package can be stored on disk as a:

- **source** - a directory with sub-directories (as above)
- **bundle** - a single compressed file (*.tar.gz*)
- **binary** - a single compressed file optimized for a specific OS

Or installed into an R library (loaded into memory during an R session) or archived online in a repository. Use the functions below to move between these states.

Setup (DESCRIPTION)

The DESCRIPTION file describes your work, sets up how your package will work with other packages, and applies a copyright.

- You must have a DESCRIPTION file
- Add the packages that yours relies on with devtools::**use_package()** Adds a package to the Imports or Suggests field

CC0	MIT	GPL-2
No strings attached.	MIT license applies to your code if re-shared.	GPL-2 license applies to your code, *and all code anyone bundles with it*, if re-shared.

```
Package: mypackage
Title: Title of Package
Version: 0.1.0
Authors@R: person("Hadley", "Wickham", email =
    "hadley@me.com", role = c("aut", "cre"))
Description: What the package does (one paragraph)
Depends: R (>= 3.1.0)
License: GPL-2
LazyData: true
Imports:
    dplyr (>= 0.4.0),
    ggvis (>= 0.2)
Suggests:
    knitr (>= 0.1.0)
```

Import packages that your package *must have* to work. R will install them when it installs your package.

Suggest packages that are not very essential to yours. Users can install them manually, or not, as they like.

Write Code (R/)

All of the R code in your package goes in R/. A package with just an R/ directory is still a very useful package.

- Create a new package project with devtools::**create("path/to/name")** Create a template to develop into a package.
- Save your code in R/ as scripts (extension .R)

Test (tests/)

Use tests/ to store tests that will alert you if your code breaks.

- Add a **tests/** directory
- Import **testthat** with devtools::**use_testthat()**, which sets up package to use automated tests with testthat
- Write tests with **context()**, **test()**, and expect statements

I'll never

BREAK your heart

Debugging

A computer only does exactly and precisely what you tell it to do.

Types of Bugs

Syntax Errors: Code doesn't run. Interpreter/Compiler doesn't know what you mean
(Includes: typos, problems with indents, mismatched brackets)

Unexpected Behaviour: Code runs (or runs to a point), but doesn't do what you expect it to.
(Includes incorrect data types, missing variables, problems with flow control)

Incorrect Result: Code runs without error, produces the incorrect result.
(Includes mathematical errors, precision problems, logic errors)

Poor Optimization: Code runs without error, produces the correct result, but takes much longer than it should.

Types of Bugs

Syntax Errors: Code doesn't run. Interpreter/Compiler doesn't know what you mean
(Includes: typos, problems with indents, mismatched brackets)

Unexpected Behaviour: Code runs (or runs to a point), but doesn't do what you expect it to.
(Includes incorrect data types, missing variables, problems with flow control)

Incorrect Result: Code runs without er
(Includes mathematical error

Poor Optimization: Code runs without er
much longer than it s

COOL CATCH

For languages that you compile (e.g. C, Fortran), when something compiles, it doesn't mean that it runs. So for the first type of errors, check both that it compiles correctly AND runs/executes.

Strategies for how to deal with bugs

Syntax Errors: Code doesn't run. Interpreter/Compiler doesn't know what you mean
(Includes: typos, problems with indents, mismatched brackets)

Unexpected Behaviour: Code runs (or runs to a point), but doesn't do what you expect it to.
(Includes incorrect data types, missing variables, problems with flow control)

Incorrect Result: Code runs without error, produces the incorrect result.
(Includes mathematical errors, precision problems, logic errors)

Poor Optimization: Code runs without error, produces the correct result, but takes much longer than it should.

Using Linters, Formatters, Bracket Highlighters, and Reading Error Messages

Strategies for how to deal with bugs

Syntax Errors: Code doesn't run, Interpreter/Compiler doesn't know what you mean.
(Includes: typos, problems with indents, mismatched brackets)

Reading Error Messages, Running subset of code tests, checking variables during execution

Unexpected Behaviour: Code runs (or runs to a point), but doesn't do what you expect it to.
(Includes incorrect data types, missing variables, problems with flow control)

Incorrect Result: Code runs without error, produces the incorrect result.
(Includes mathematical errors, precision problems, logic errors)

Poor Optimization: Code runs without error, produces the correct result, but takes much longer than it should.

Strategies for how to deal with bugs

Syntax Errors: Code doesn't run. Interpreter/Compiler doesn't know what you mean
(Includes: typos, problems with indents, mismatched brackets)

Running with test data/parameters with known expectations to determine where the problem is

Unexpected Behaviour: Code runs (without error or crash) but does not do what you expect it to.
(Includes incorrect data types, missing variables, problems with flow control)

Incorrect Result: Code runs without error, produces the incorrect result.
(Includes mathematical errors, precision problems, logic errors)

Poor Optimization: Code runs without error, produces the correct result, but takes much longer than it should.

Strategies for how to deal with bugs

Syntax Errors: Code doesn't run. Interpreter/Compiler doesn't know what you mean
(Includes: typos, problems with indents, mismatched brackets)

Unexpected Behaviour: Code runs (or runs to a point), but doesn't do what you expect it to
(Includes incorrect data types, misuse of variables, problems with flow control)

Incorrect Result: Code runs without error, produces the incorrect result.
(Includes mathematical errors, precision problems, logic errors)

Poor Optimization: Code runs without error, produces the correct result, but takes much longer than it should.

Breaking code into smaller steps to see where slowdowns are occurring. Profiling Code.

What's that Error Message?

This is a traceback, it is your friend. Your talkative, hyper-detailed friend, but your friend nonetheless:

```
In [19]: two_dim_loop(3)
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-19-7e3574312995> in <module>
----> 1 two_dim_loop(3)

<ipython-input-18-defffd49cb25> in two_dim_loop(val)
      2     for i in range(val):
      3         for j in range(val):
----> 4             multiply_by_inverted_number(i, j)
      5

<ipython-input-17-eab3f38f874e> in multiply_by_inverted_number(num1, num2)
      1 def multiply_by_inverted_number(num1, num2):
----> 2     return num1 * divide_by_number(num2)
      3

<ipython-input-16-bcba7c1d330c> in divide_by_number(num)
      1 def divide_by_number(num):
----> 2     return 1.0/num
      3

ZeroDivisionError: float division by zero
```

What's that Error Message?

This is a traceback, it is your friend. Your talkative, hyper-detailed friend, but your friend nonetheless:

What type of error it is

```
In [19]: two_dim_loop(3)
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-19-7e3574312995> in <module>
----> 1 two_dim_loop(3)

<ipython-input-18-defffd49cb25> in two_dim_loop(val)
      2     for i in range(val):
      3         for j in range(val):
----> 4             multiply_by_inverted_number(i, j)
      5

<ipython-input-17-eab3f38f874e> in multiply_by_inverted_number(num1, num2)
      1 def multiply_by_inverted_number(num1, num2):
----> 2     return num1 * divide_by_number(num2)
      3

<ipython-input-16-bcba7cfd330c> in divide_by_number(num)
      1 def divide_by_number(num):
----> 2     return 1.0/num
      3

ZeroDivisionError: float division by zero
```

What's that Error Message?

This is a traceback, it is your friend. Your talkative, hyper-detailed friend, but your friend nonetheless:

The highest level function where the error happened

```
In [19]: two_dim_loop(3)
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-19-7e3574312995> in <module>
----> 1 two_dim_loop(3)

<ipython-input-18-defffd49cb25> in two_dim_loop(val)
      2     for i in range(val):
      3         for j in range(val):
----> 4             multiply_by_inverted_number(i, j)
      5

<ipython-input-17-eab3f38f874e> in multiply_by_inverted_number(num1, num2)
      1 def multiply_by_inverted_number(num1, num2):
----> 2     return num1 * divide_by_number(num2)
      3

<ipython-input-16-bcba7c1d330c> in divide_by_number(num)
      1 def divide_by_number(num):
----> 2     return 1.0/num
      3

ZeroDivisionError: float division by zero
```

The root of where the error was caused

What's that Error Message?

This is a traceback, it is your friend. Your talkative, hyper-detailed friend, but your friend nonetheless:

The name of the file/input where the code was written

```
In [19]: two_dim_loop(3)
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-19-7e3574312995> in <module>
----> 1 two_dim_loop(3)

<ipython-input-18-defffd49cb25> in two_dim_loop(val)
      2     for i in range(val):
      3         for j in range(val):
----> 4             multiply_by_inverted_number(i, j)
      5

<ipython-input-17-eab3f38f874e> in multiply_by_inverted_number(num1, num2)
      1 def multiply_by_inverted_number(num1, num2):
----> 2     return num1 * divide_by_number(num2)
      3

<ipython-input-16-bcba7c1d330c> in divide_by_number(num)
      1 def divide_by_number(num):
----> 2     return 1.0/num
      3

ZeroDivisionError: float division by zero
```

The line number where the problem is

Minimal Reproducible Code Example

All the code you need to recreate the error.

If you have multiple functions, truncate what you need

Provide the data/values that cause the exception

Unit Tests

Typically, when you write code, you'll write it sequentially:

Unit Tests

Typically, when you write code, you'll write it sequentially:

It's often useful to think of it as a bunch of connected units. Separating out these units allows you to reproduce them

Unit Tests

Typically, when you write code, you'll write it sequentially:

It's often useful to think of it as a bunch of connected units. Separating out these units allows you to reproduce them

You want to build units that you can test thoroughly and ensure they work – this allows you to reuse them and allows you to code faster

pyTest

PyTest is a framework to allow you to build tests around each of your units.

```python
import pytest
import numpy as np

import loggingfile

def test_take_sum_of_numbers():
    x = 2
    y = 3
    z = loggingfile.take_sum_of_numbers(x, y)

    assert z == 6, "Test Failed"
```

```
pytest
========================================================== test ses
platform linux -- Python 3.7.6, pytest-5.3.5, py-1.8.1, pluggy-0.13.1
rootdir: /mnt/c/Users/mubdi/OneDrive/Documents/GitHub/exercises_week5
plugins: hypothesis-5.5.4, doctestplus-0.5.0, openfiles-0.4.0, astrop
collected 1 item

test_loggingfile.py F                                                    [100%]

================================================================= FAILURES =================================================================
_____________________________________________________ test_take_sum_of_numbers _____________________________________________________

    def test_take_sum_of_numbers():
        x = 2
        y = 3
        z = loggingfile.take_sum_of_numbers(x, y)

>       assert z == 6, "Test Failed"
E       AssertionError: Test Failed
E       assert 5 == 6

test_loggingfile.py:11: AssertionError
========================================================== 1 failed in 0.25s ==========================================================
```

pyTest

PyTest is a framework to allow you to build tests around each of your units.

```python
import pytest
import numpy as np

import loggingfile

def test_take_sum_of_numbers():
    x = 2
    y = 3
    z = loggingfile.take_sum_of_numbers(x, y)

    assert z == 6, "Test Failed"
```

```
pytest
========================================================== test ses
platform linux -- Python 3.7.6, pytest-5.3.5, py-1.8.1, pluggy-0.13.1
rootdir: /mnt/c/Users/mubdi/OneDrive/Documents/GitHub/exercises_week5
plugins: hypothesis-5.5.4, doctestplus-0.5.0, openfiles-0.4.0, astrop
collected 1 item

test_loggingfile.py F

================================================================= FAILURES =================================================
_____________________________________________________ test_take_sum_of_numbers _____________________________________________

    def test_take_sum_of_numbers():
        x = 2
        y = 3
        z = loggingfile.take_sum_of_numbers(x, y)

>       assert z == 6, "Test Failed"
E       AssertionError: Test Failed
E       assert 5 == 6

test_loggingfile.py:11: AssertionError
========================================================== 1 failed in 0.25s ===========================================
```

HOT TIP

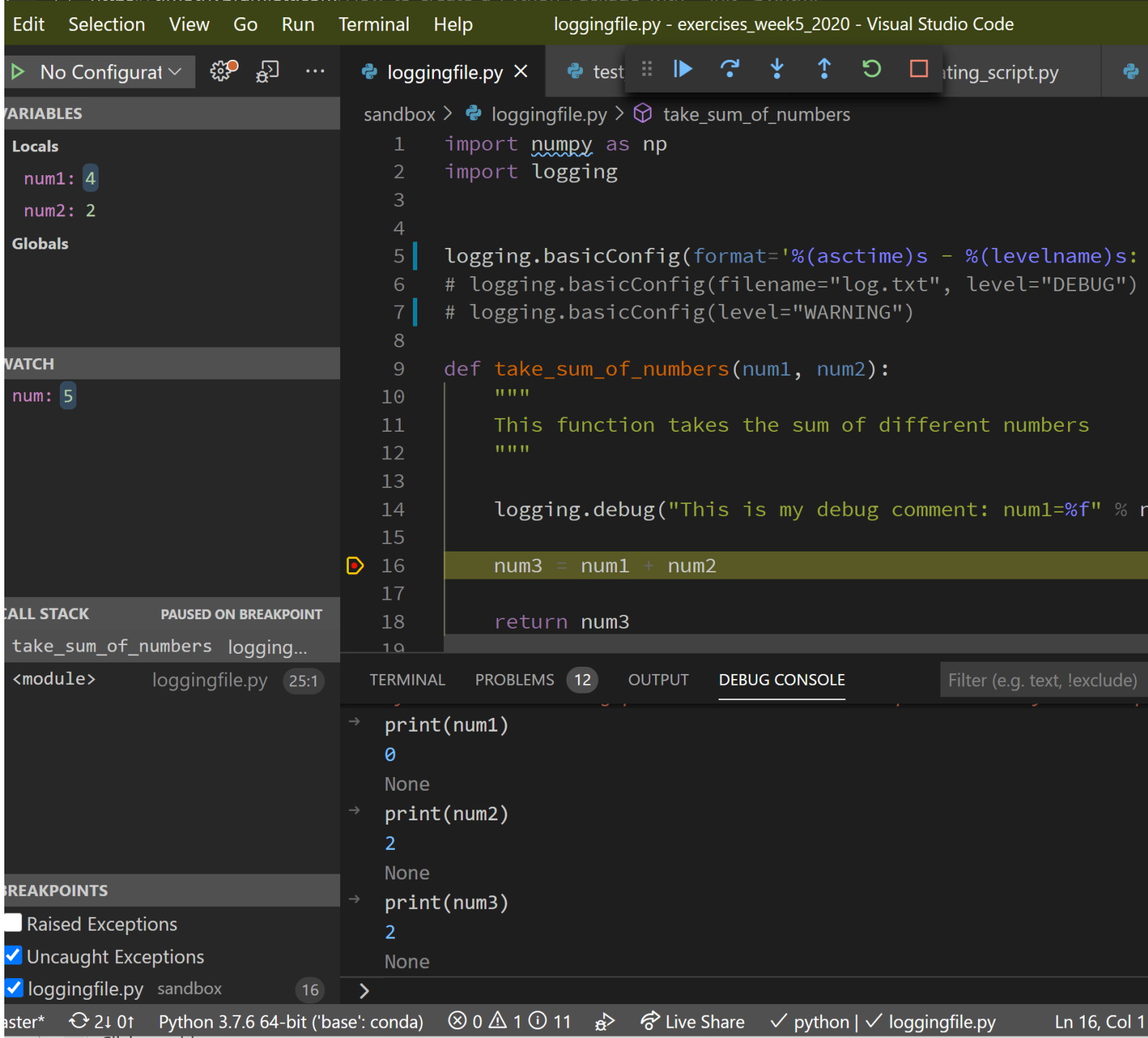
In R, the equivalent package is called `testthat`

Debugging via VS Code

Breakpoints

Conditional Breakpoints

Logging

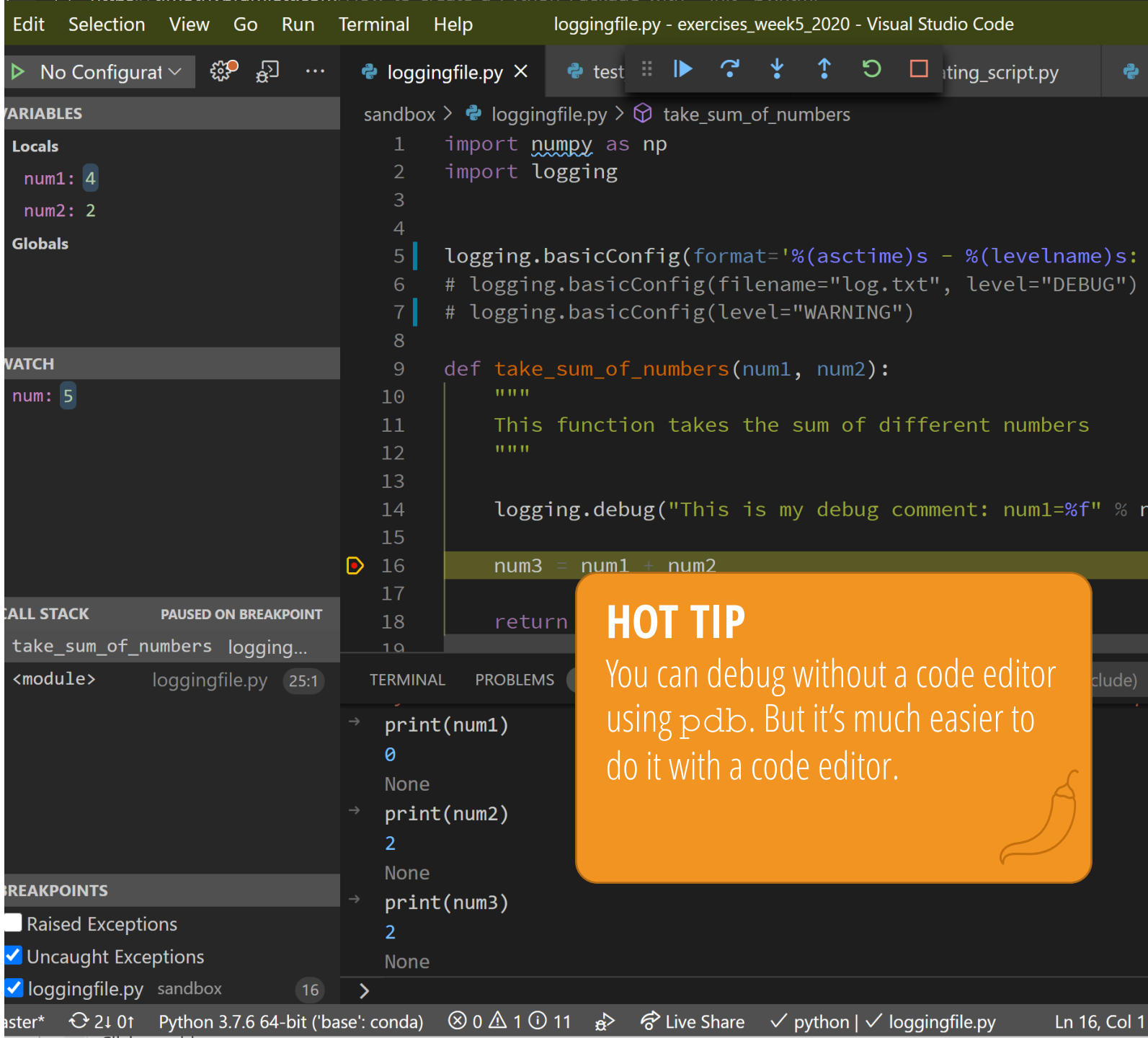


Debugging via VS Code

Breakpoints

Conditional Breakpoints

Logging



HOT TIP

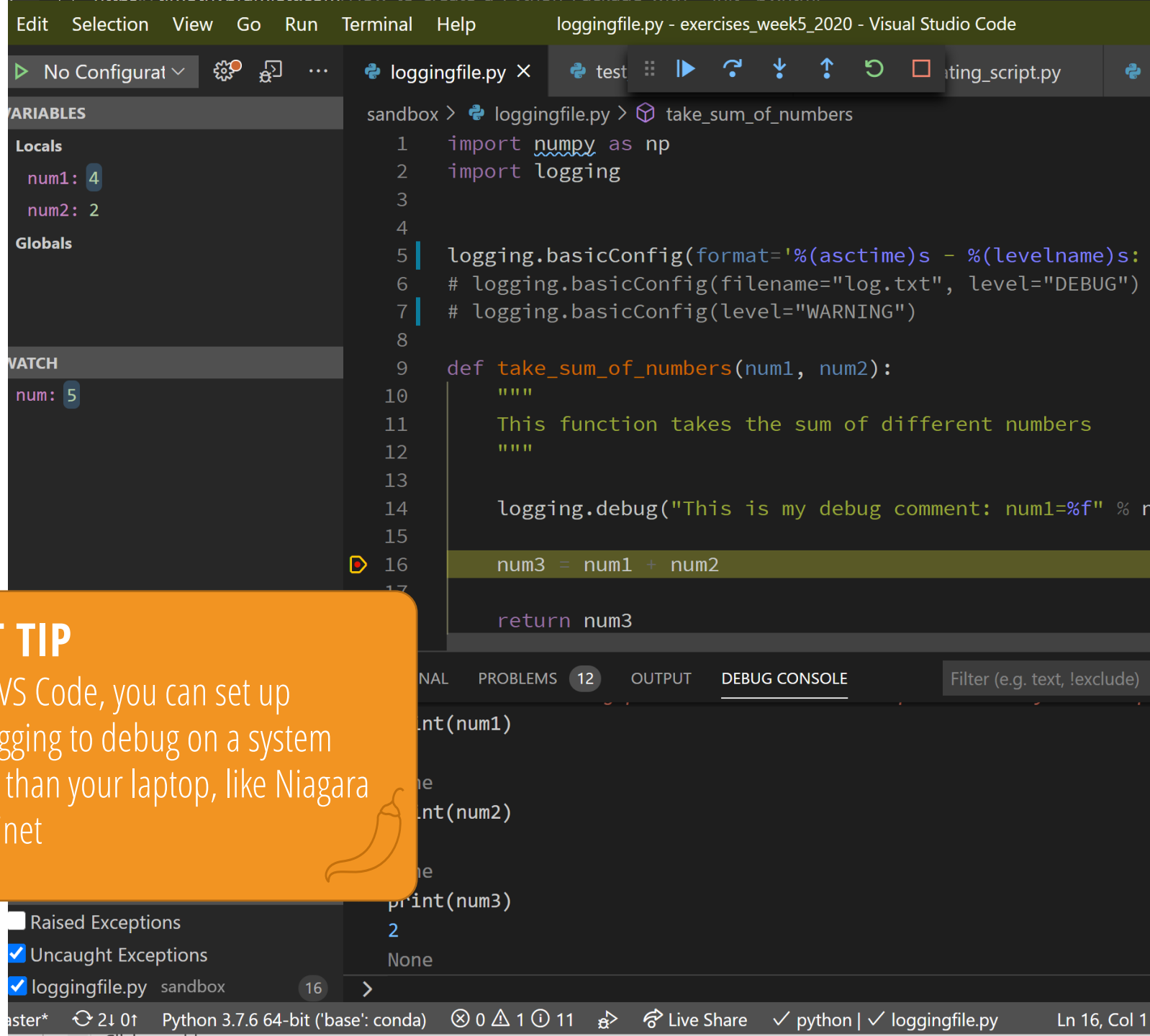
You can debug without a code editor using `pdb`. But it's much easier to do it with a code editor.

Debugging via VS Code

Breakpoints

Conditional Breakpoints

Logging



HOT TIP

With VS Code, you can set up debugging to debug on a system other than your laptop, like Niagara on scinet

Debugging in R

- `traceback()`
  - Same idea as in python
  - Shows you the environments that were being used when error thrown
- `options(error=recover)`
  - The `options` function allows you to change how R behaves
  - Setting the `error` option to `recover`, tells R to call the recover function when an error happens
- `browser()`

```
> boot_coefs( SDSS_quasar, n = 10 )
Error in `[.data.frame`(x, sample.int(nrow(x))) :
  undefined columns selected

Enter a frame number, or 0 to exit

1: boot_coefs(SDSS_quasar, n = 10)
2: #6: x[sample.int(nrow(x))]
3: #6: `[.data.frame`(x, sample.int(nrow(x)))

Selection:
```

Exercise

Teams of 2-3 (Distribute Parts)

Using the SDSS Web Interface, grab all objects between 29.75 < dec < 30 and 180.75 < RA < 181 (download them as CSV files) from the following tables:

1. the PhotoObj Table left joined with the SpecObj table (joined on objid and bestobjid respectively), grabbing the columns objid, ra, dec, u, g, r, i, z from PhotoObj, and z and class from SpecObj
2. the tmassxsc Table left joined with the PhotoObj table (joined on objid from both tables), grabbing the columns ra, dec, J_M_K20FE, H_M_K20FE, K_M_K20FE from tmassxsc, and objid, ra, dec, u, g, r, i, z from PhotoObj
3. the FIRST Table left joined with the PhotoObj table (joined on the objid from both tables) grabbing the columns ra, dec, and integr from FIRST, and objid, ra, dec, u, g, r, i, z from PhotoObj

Exercise

Take a Jupyter Notebook we give you (**`Get_SDSS_spectra.ipynb`**) and turn it into a production script.

Optional Exercise 1

Take the code from **`error_generating_script.py`**, follow the traceback, and make a minimal reproducible example

Exercise

(Thanks Dr. Aaron Springford for making this example!)

- Open the week5_exercise.R in R studio
- Read the comments and work through the code, running the lines
- Try the different debugging tools, and fix the bugs

Optional Exercise 2

Run through a debugging session to fix an issue in **`error_generating_script.py`**