Session 4: Practical Coding and Statistics in Astronomy

SQL, DEBUGGING, & DATAVIS!

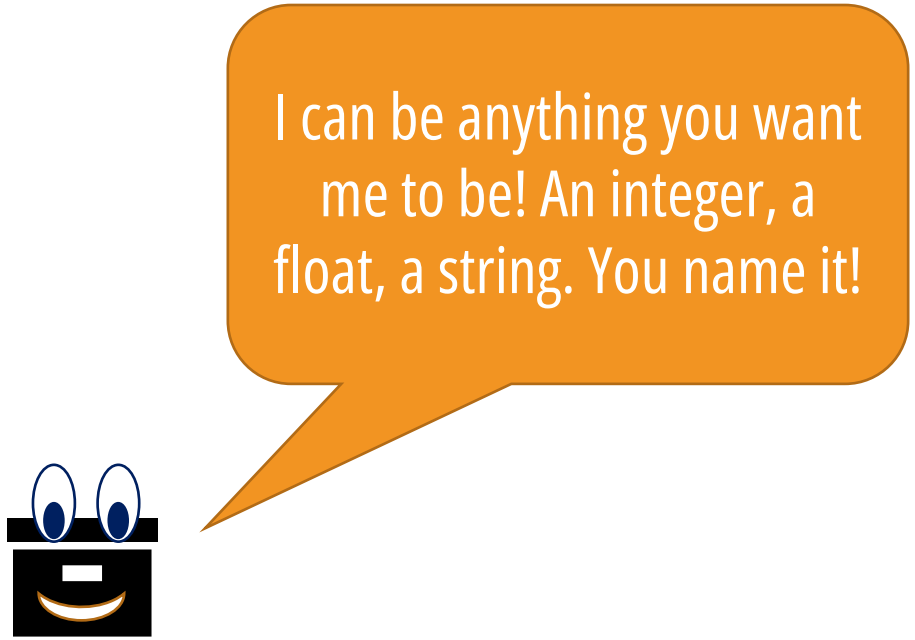
STARFISH SCHOOL 2022

Databases and their jargon

How do you solve a problem like data?

How do you solve a problem like data?

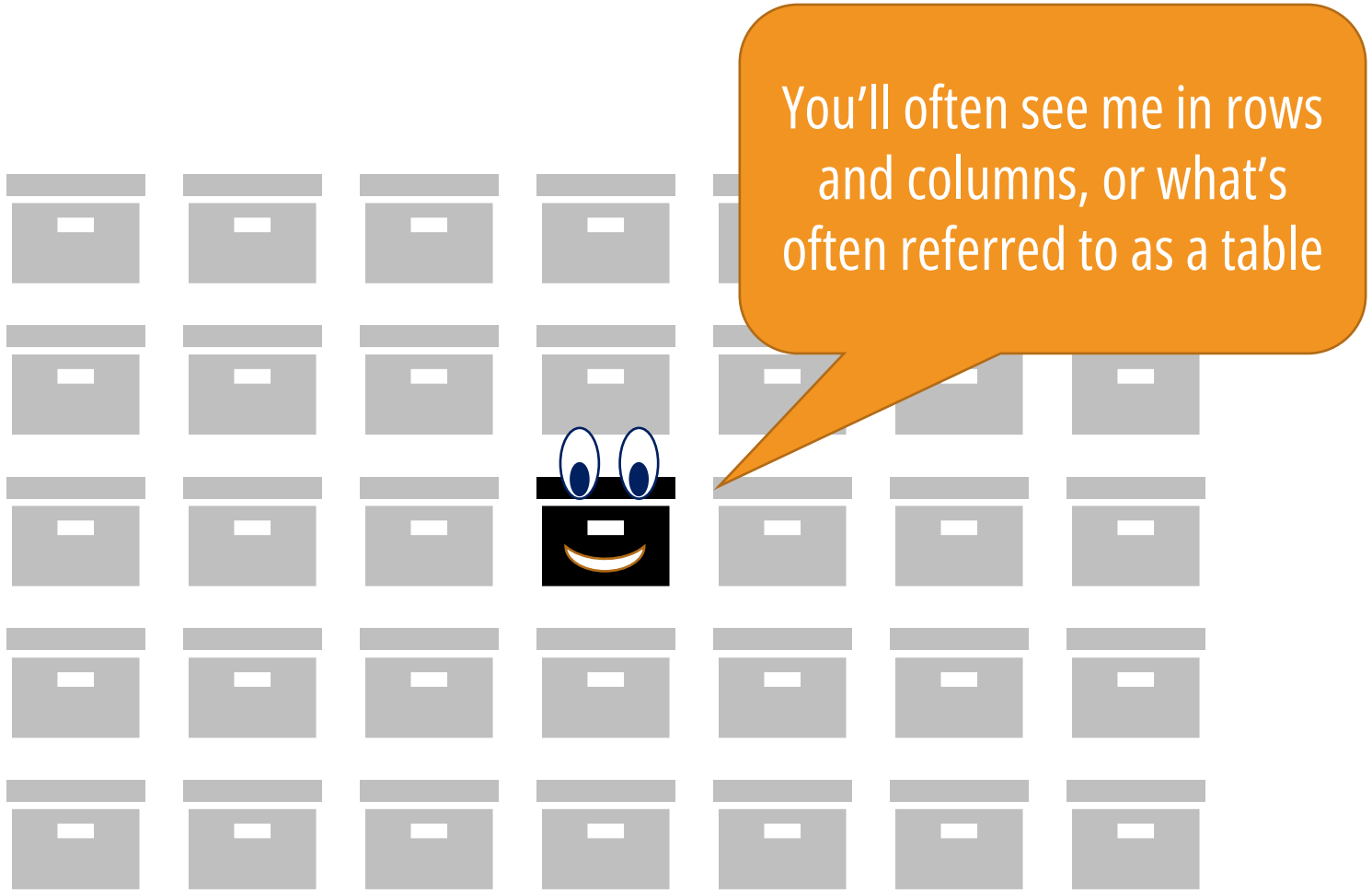
How do you solve a problem like data?



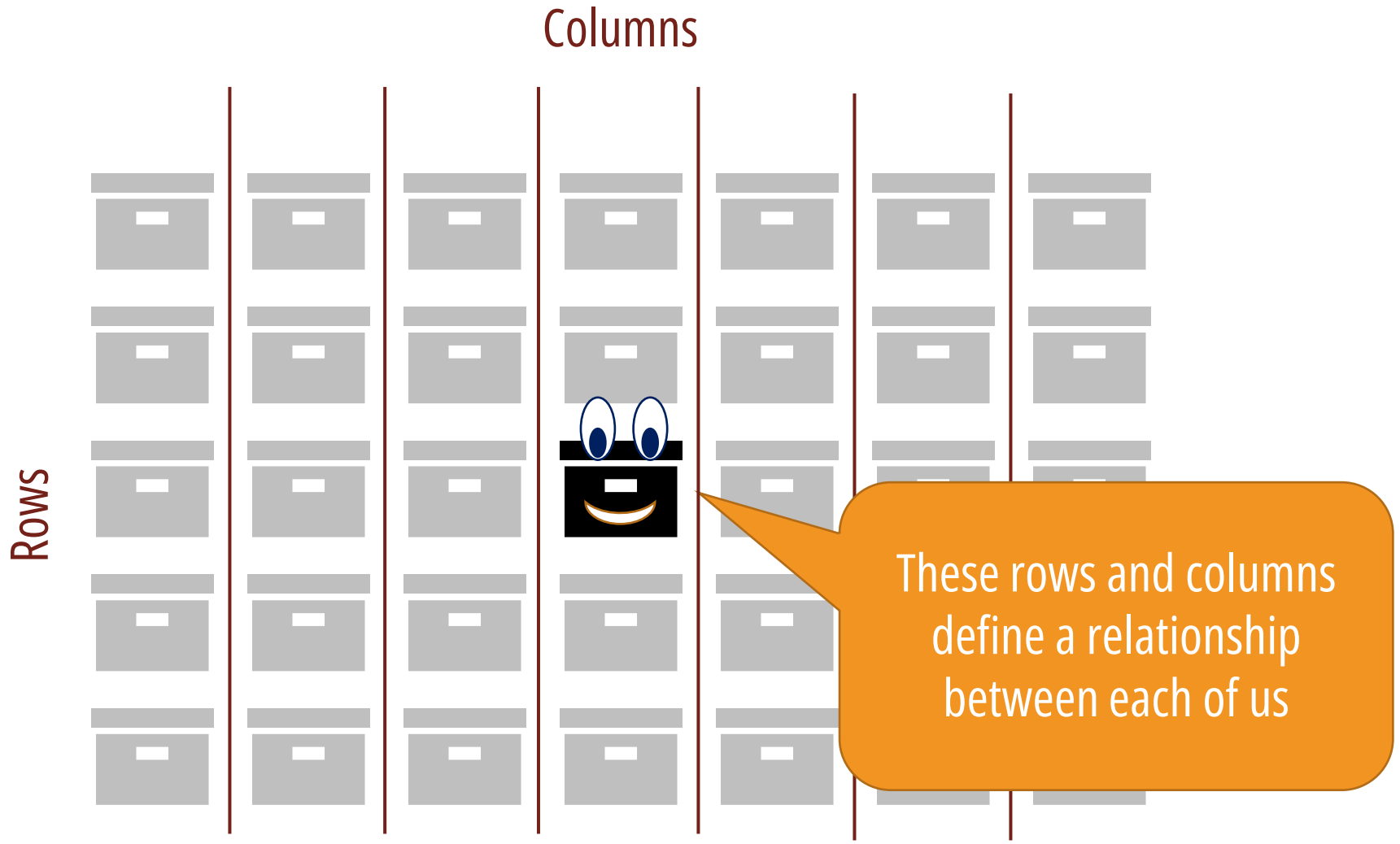
How do you solve a problem like data?

How do you solve a problem like data?

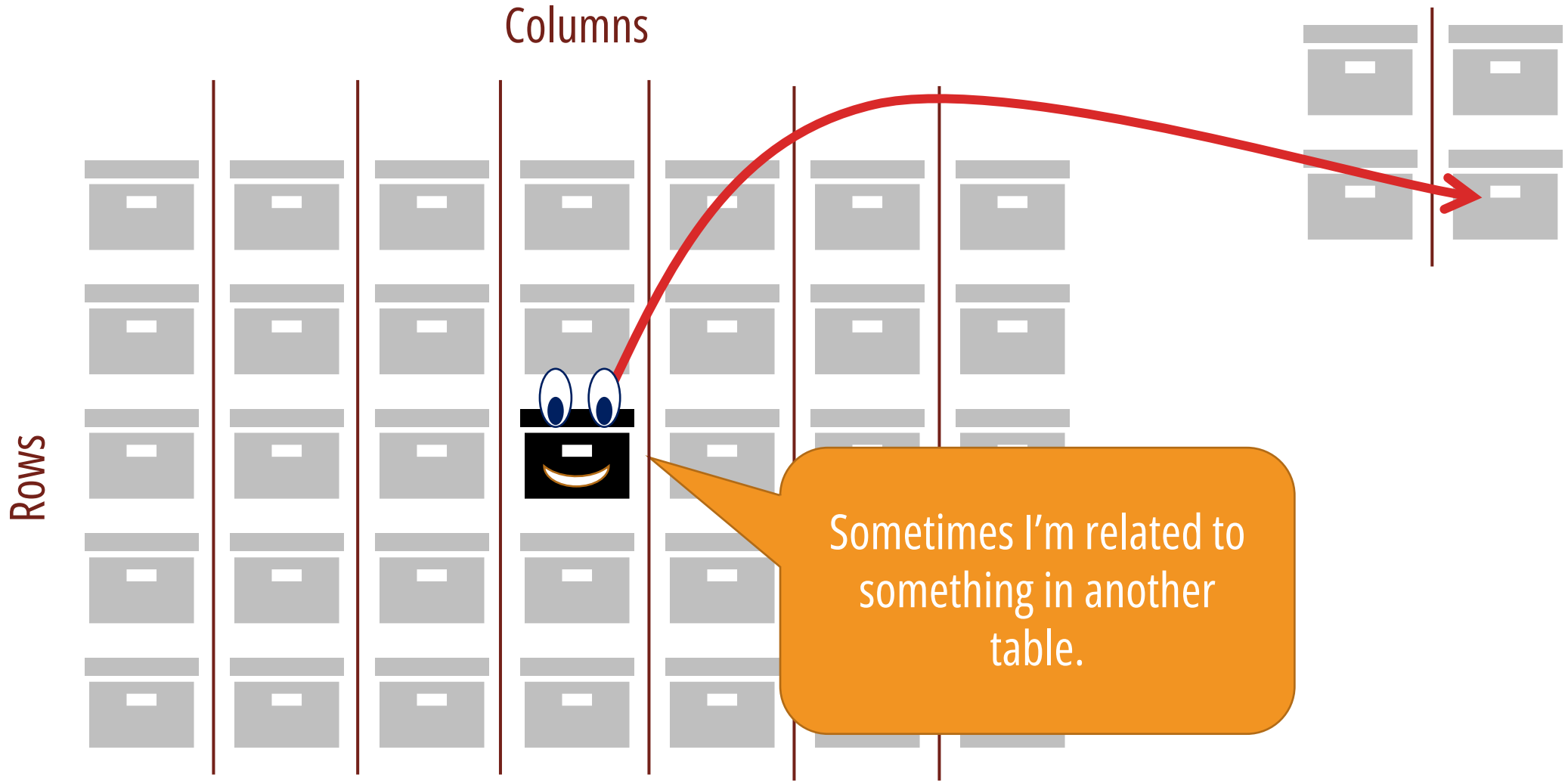
How do you solve a problem like data?



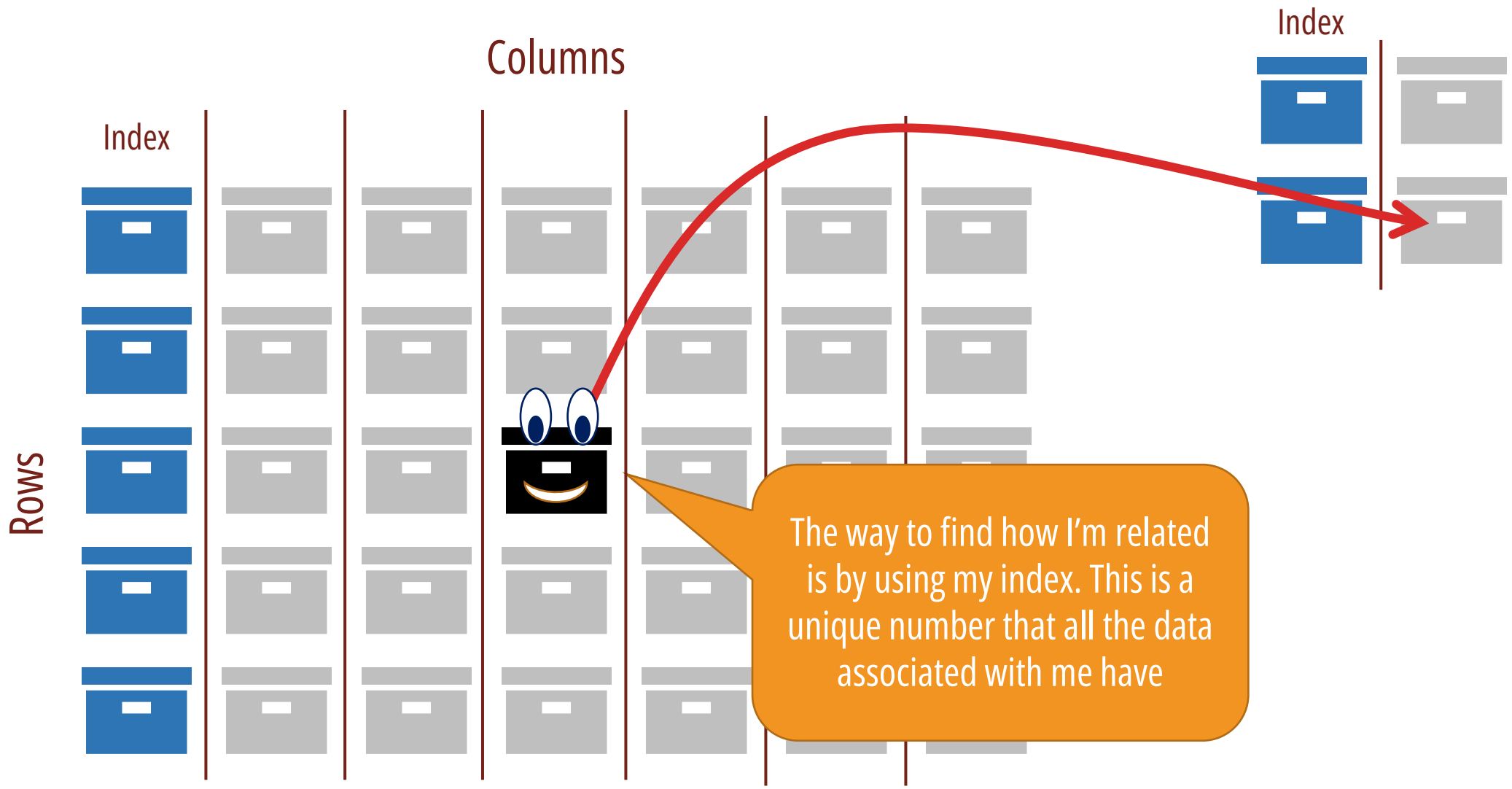
How do you solve a problem like data?



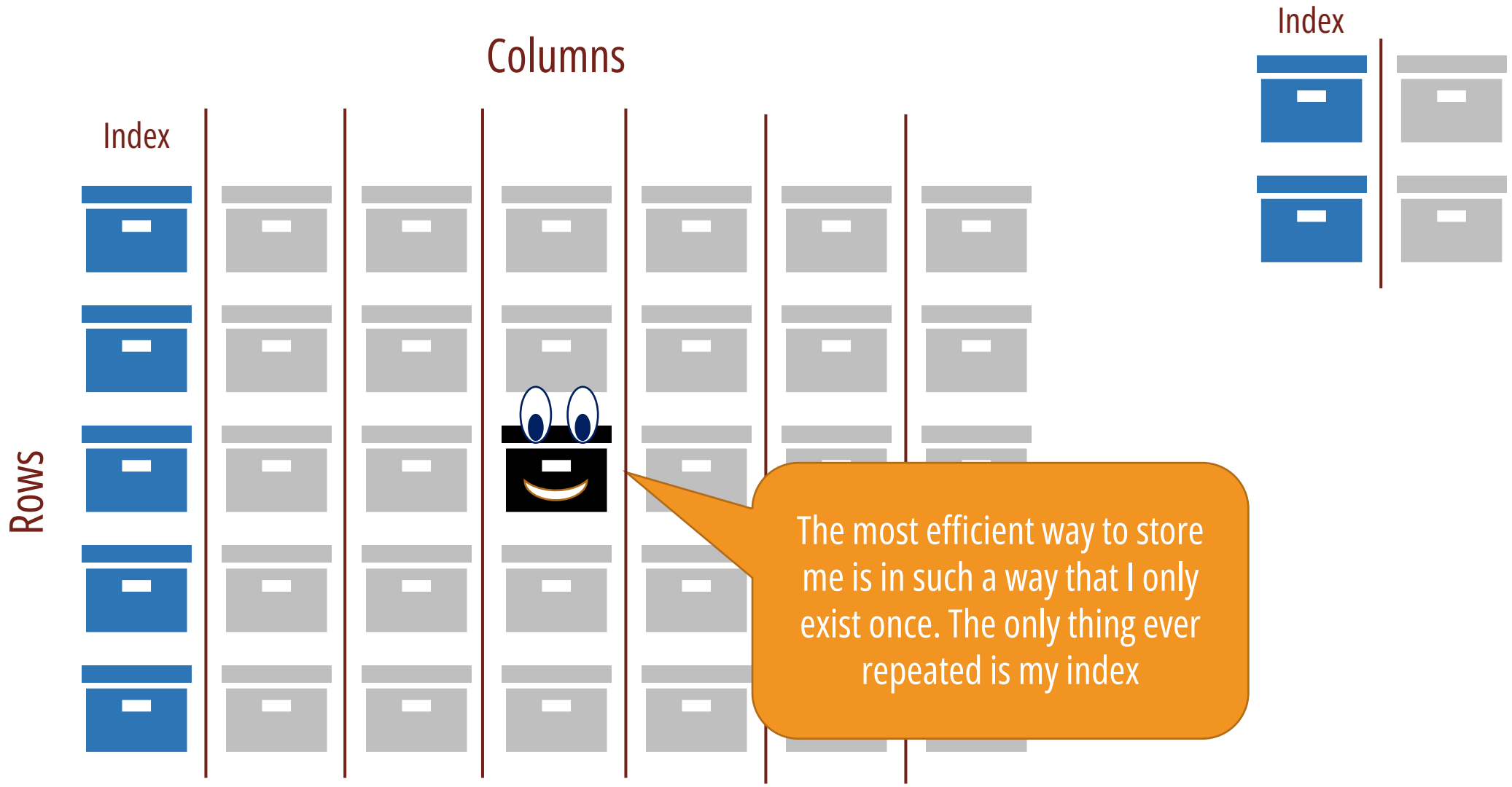
How do you solve a problem like data?



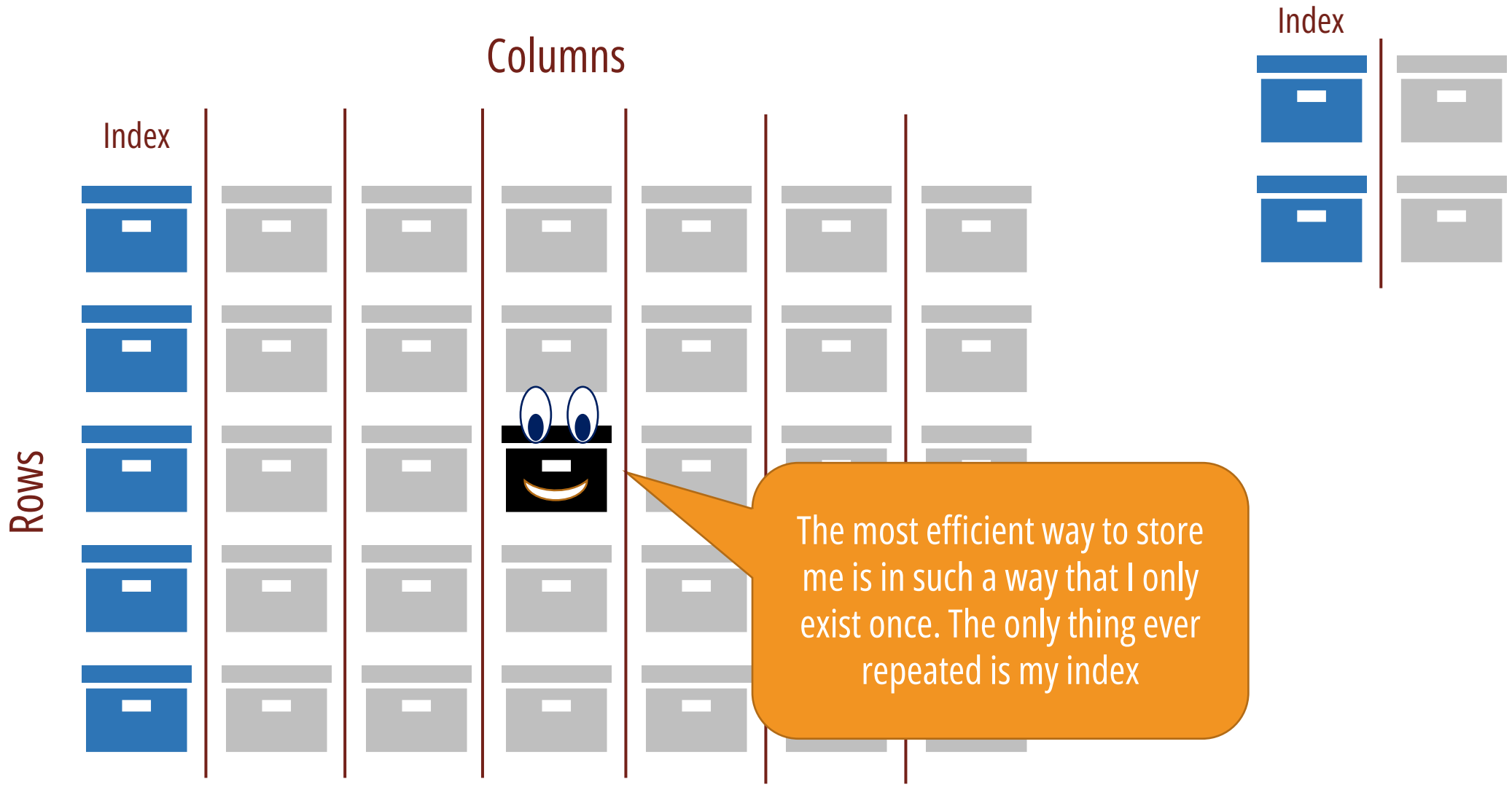
How do you solve a problem like data?



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

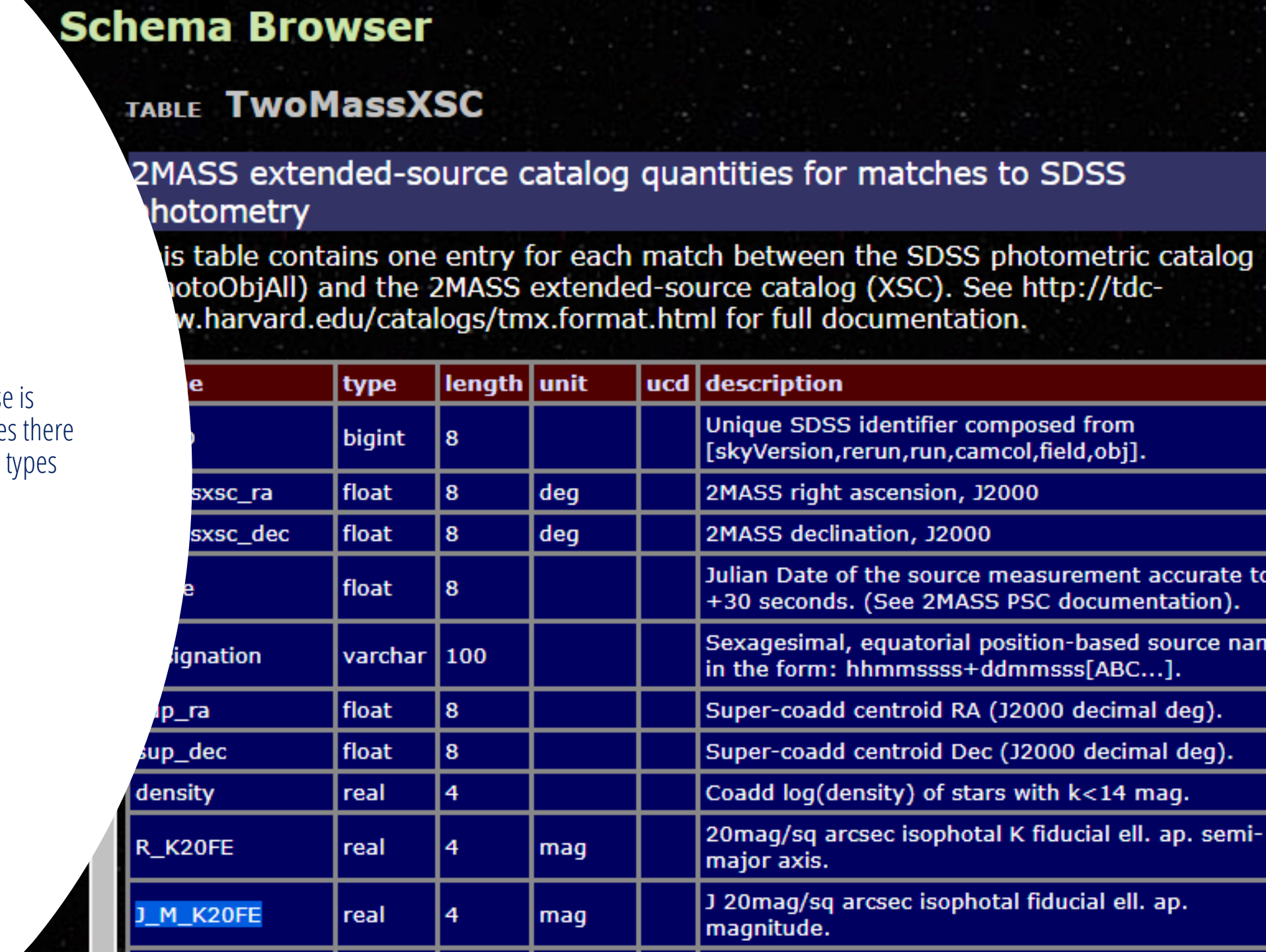
The SELECT command is how to grab data from the database. The simplest query of all:

```
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.

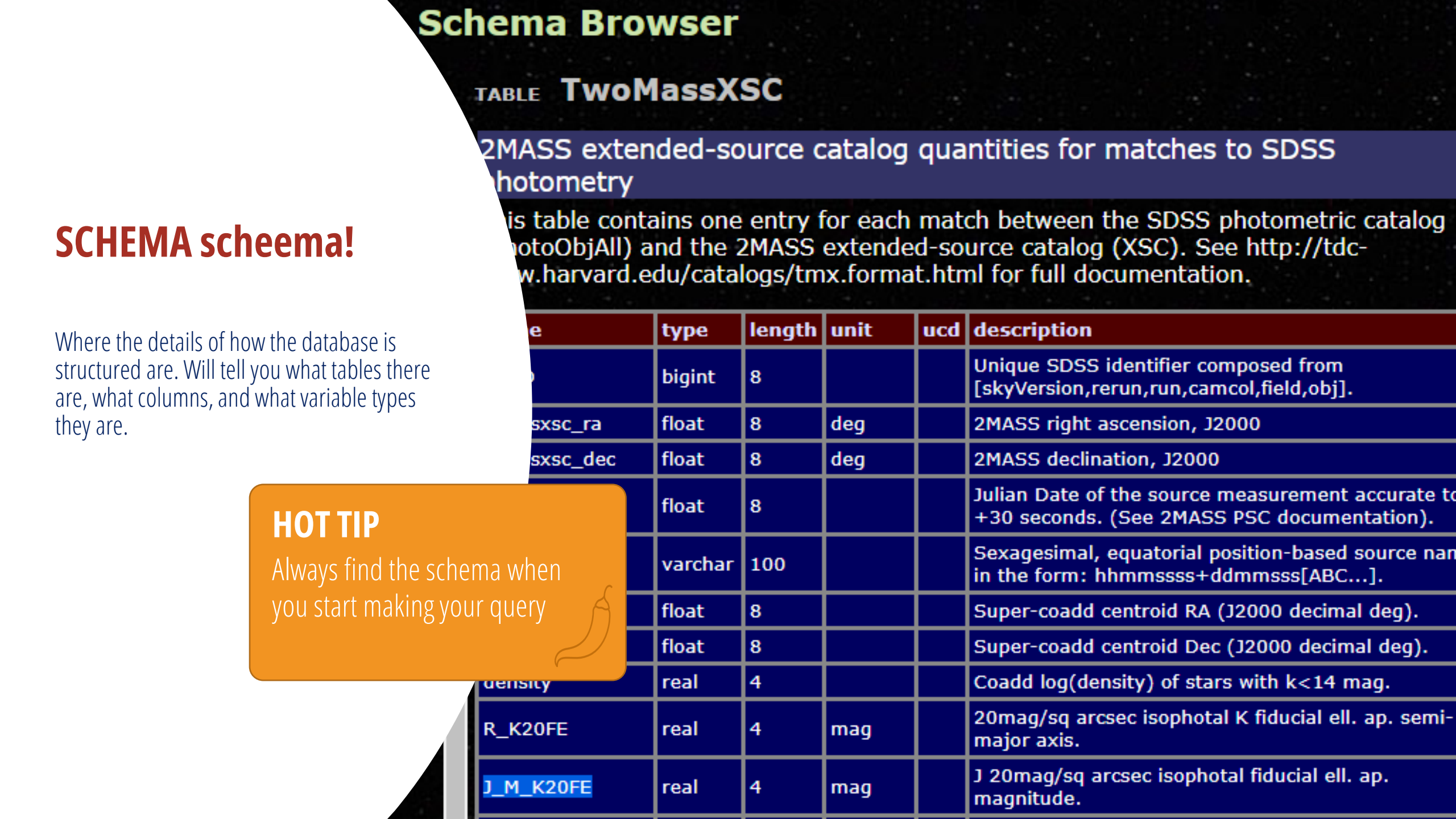


SCHEMA scheema!

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

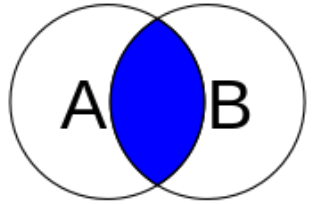
The real power from databases comes from taking information from multiple tables and processing it together. Let's say there's table1 and table2 that have different data, but both have the unique identifier in a column called "ID", you can get all data merged from the two tables:

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

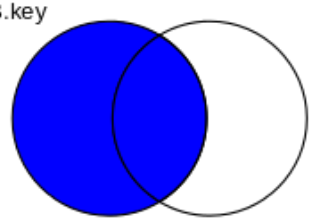
COOL CATCH

All of your joins should be done before your WHEREs.

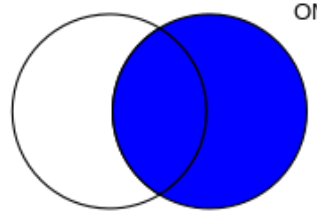
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



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

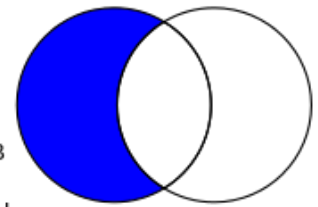


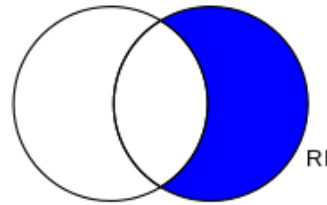
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key



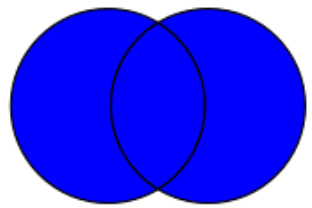
SQL JOINS

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

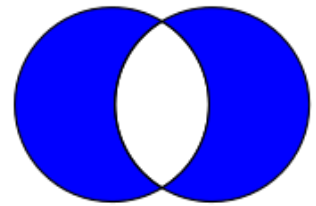




SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE a.key IS NULL



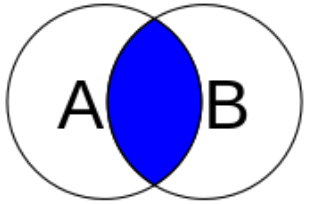
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key



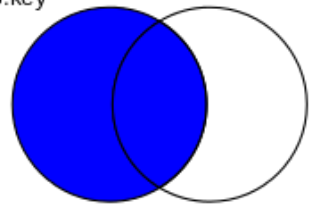
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key iIS NULL

SQL JOINS

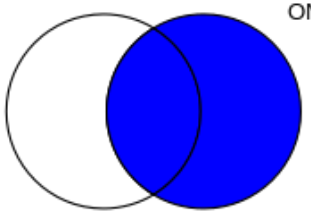
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



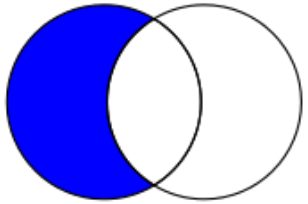
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key



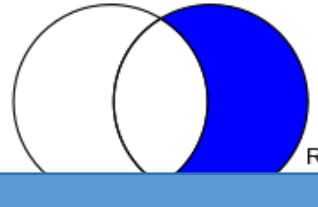
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key



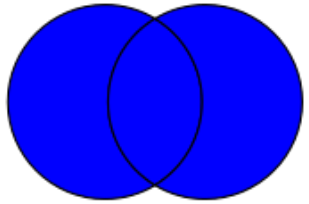
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL



SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B



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

COOL CATCH

By default, all joins in SQL are inner joins

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.

SQL via Python!
Example with Astroquery and Gaia

In [2]:
```
from astroquery.gaia import Gaia

query_text = '''SELECT TOP 4096 ra, dec, parallax, pmra, pmdec, radial_velocity,
phot_g_mean_mag, phot_bp_mean_mag, phot_rp_mean_mag
FROM gaiadr3.gaia_source
WHERE parallax_over_error > 10 AND
    parallax > 10 AND
    radial_velocity BETWEEN -200 AND 200
'''
```

In [3]:
```
job = Gaia.launch_job(query_text)
gaia_data = job.get_results()
```

In [4]:
```
gaia_data
```

Out[4]: *96*

ra	dec	parallax	pmra	pmdec	radial_velocity	phot_g_mean_mag	phot_bp_mean_mag	phot_rp_mean_m
deg	deg	mas	mas / yr	mas / yr	km / s	mag	mag	m
at64	float64	float64	float64	float64	float32	float32	float32	floa
)922	-50.63863518231249	13.398257319108815	-49.35844067323052	-84.67277009494555	12.315269	13.470652	14.585335	12.4197
9386	-50.568383922784385	18.395337998280908	-20.821239369836658	-76.53943156160437	-29.955275	14.576204	16.034027	13.3565
3107	-51.477673785228106	19.588654431566606	11.454067954459212	131.33292721367764	45.4255	6.22464	6.3795276	5.9394
5151	-51.606571251985955	10.368954647575999	5.946151060742354	-40.65596429190273	7.1170692	15.446747	16.958754	14.2442
4702	-51.59898526057091	10.326265936660176	6.066630513795433	-40.46666568311353	7.0510607	13.695807	14.8268175	12.6357

SQL via Python
Example with NOIRLab Datalab

In [17]:
```
# Data Lab
from dl import queryClient as qc
```

In [18]:
```
ra = 189.58
dec = -40.89

# Create the query string; SQL keyword capitalized for clarity

query = """SELECT ra,dec,mag_auto_g,mag_auto_r,mag_auto_i
   FROM delve_dr1.objects
   WHERE q3c_radial_query(ra, dec, {0}, {1}, 0.5) AND
         mag_auto_g BETWEEN 14 AND 25 AND
         (mag_auto_g - mag_auto_r) BETWEEN -0.4 AND 0.4""".format(ra, dec)
print(query)
```

```
SELECT ra,dec,mag_auto_g,mag_auto_r,mag_auto_i
   FROM delve_dr1.objects
   WHERE q3c_radial_query(ra, dec, 189.58, -40.89, 0.5) AND
         mag_auto_g BETWEEN 14 AND 25 AND
         (mag_auto_g - mag_auto_r) BETWEEN -0.4 AND 0.4
```

In [19]:
```
data = qc.query(sql=query,fmt='pandas') # data is a pandas DataFrame
```

In [20]:
```
data
```

Out[20]:

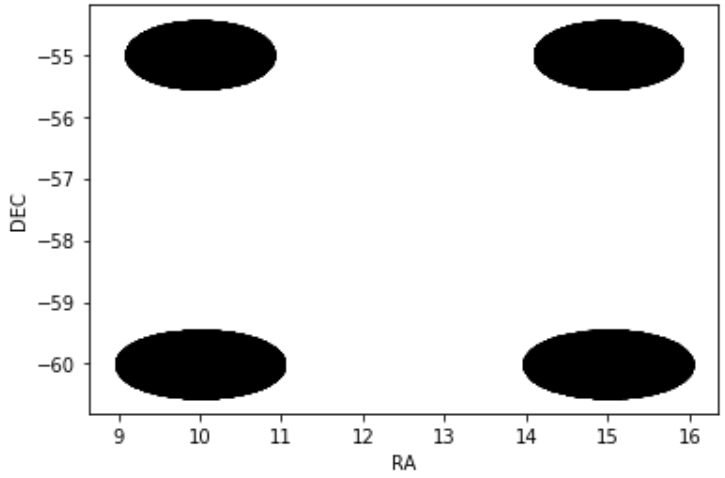
	ra	dec	mag_auto_g	mag_auto_r	mag_auto_i
0	189.403377	-41.348853	23.556286	23.566725	23.279278
1	189.307646	-41.324232	22.742039	22.548280	22.021933
2	189.443498	-41.374310	21.653559	21.291859	21.216507
3	189.484940	-41.369037	20.421825	20.113430	20.205090

SQL via Python
Example with NOIRLab Datalab

In [42]:
```
import numpy as np
from dl import queryClient as qc
import matplotlib.pyplot as plt

def getData(ra,dec,radius,columns='*'):

    query_template =\
    """SELECT {0} FROM des_dr1.main
       WHERE q3c_radial_query(ra,dec,{1},{2},{3})"""

    query = query_template.format(columns,ra,dec,radius)

    result = qc.query(sql=query) # by default the result is a CSV formatted string

    df = convert(result) # convert to Pandas dataframe object

    return df
```

In [43]:
```
columns = '''ra,dec,mag_auto_g,mag_auto_i,mag_auto_r,mag_auto_z,flags_g,flags_i,flags_r,flags_z,
    flux_auto_g,fluxerr_auto_g,flux_auto_i,fluxerr_auto_i,spread_model_g,spread_model_i,
    spread_model_r,spread_model_z,class_star_g,class_star_i,class_star_r,class_star_z,
    kron_radius,tilename'''

radius = 0.5

ra_list = np.arange(10,20,5)
dec_list = np.arange(-60,-50,5)
```

In [44]:
```
for ra in ra_list:
    for dec in dec_list:
        plt.plot(ra, dec,'x')
        tmp = getData(ra,dec,radius, columns)
        plt.plot(tmp['ra'],tmp['dec'],'.k')
plt.xlabel('RA')
plt.ylabel('DEC')
```

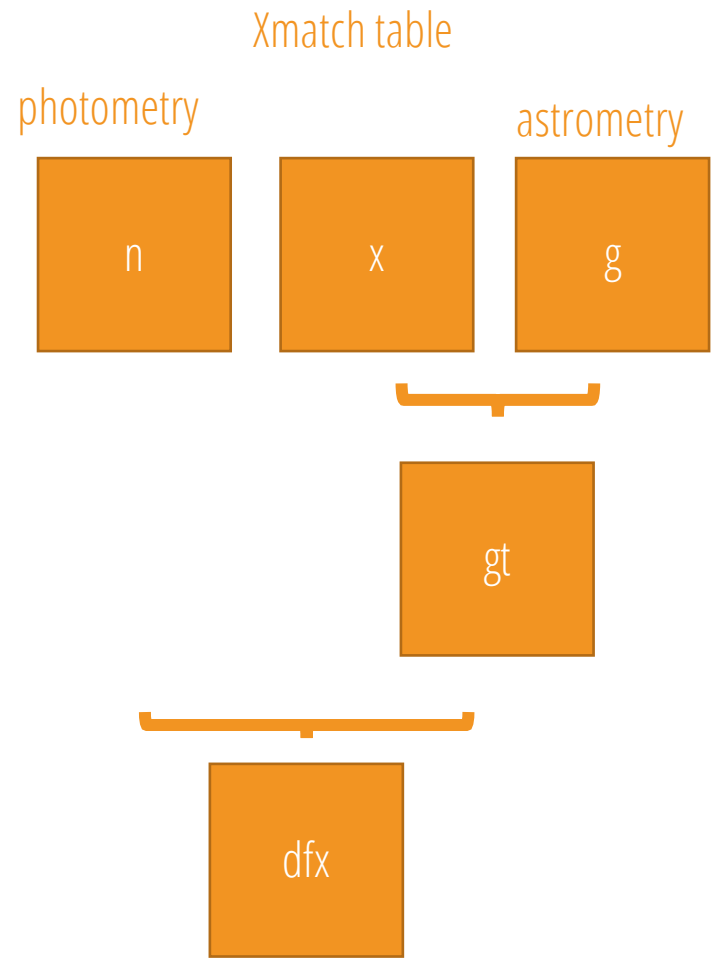


SQL via Python
Example with NOIRLab Datalab

In [50]:
```
from astropy.coordinates import name_resolve
from dl import queryClient as qc

c = name_resolve.get_icrs_coordinates('Palomar 5')
radius=2.0
query2="""
SELECT
    n.id, n.ra, n.dec, n.mjd, n.umag, n.uerr, n.gmag, n.gerr, n.rmag, n.rerr,
    n.imag, n.ierr, n.zmag, n.zerr, gt.*
FROM
    nsc_dr2.object AS n
LEFT JOIN (
    SELECT
        x.id1 as gaia_id, x.id2 as nsc_id, x.ra1 as gaia_ra, x.dec1 as gaia_dec, x.distance as dist,
        g.pmra as gaia_pmra, g.pmra_error as gaia_pmra_err,
        g.pmdec as gaia_pmdec, g.pmdec_error as gaia_pmdec_err,
        g.parallax, g.parallax_error
    FROM
        gaia_dr3.gaia_source AS g
    INNER JOIN
        gaia_dr3.x1p5__gaia_source__nsc_dr2__object AS x
    ON
        g.source_id = x.id1
    WHERE
        (q3c_radial_query(g.ra,g.dec,{0},{1},{2}))
) AS gt
ON
    n.id = gt.nsc_id
WHERE
    (q3c_radial_query(n.ra,n.dec,{0},{1},{2}))
""".format(c.ra.value,c.dec.value,radius)
```

In [51]:
```
%%time
dfx=qc.query(sql=query2,fmt='pandas')
print(f"Rows in resulting table: {len(dfx)}")
```

```
Rows in resulting table: 1083986
CPU times: user 3.49 s, sys: 781 ms, total: 4.27 s
Wall time: 25.3 s
```

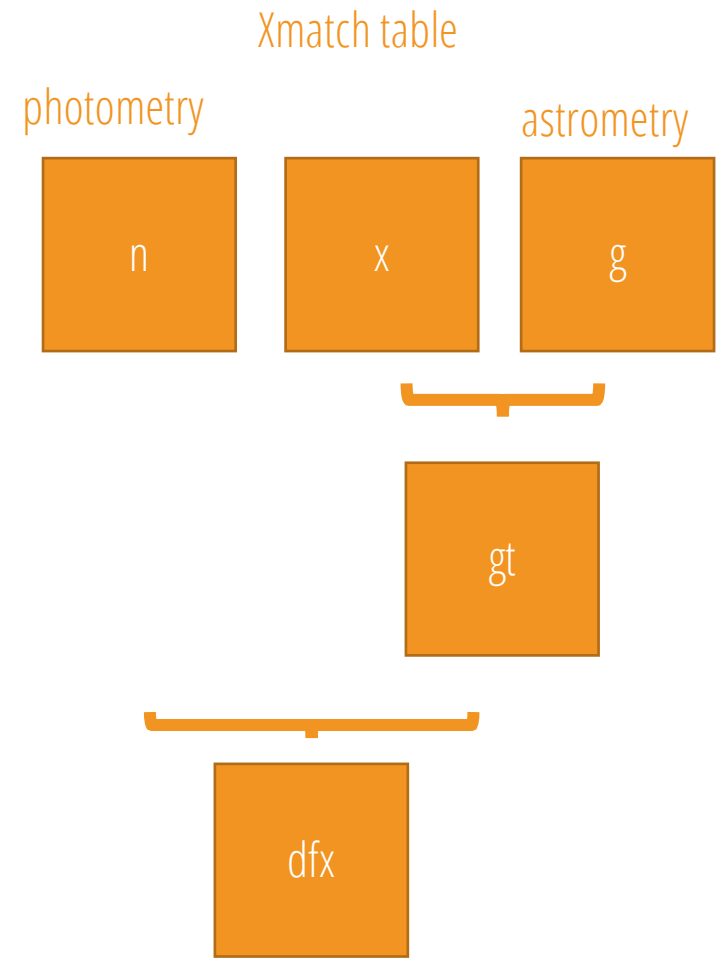


SQL via Python
Example with NOIRLab Datalab

```
In [73]: dfx['id','ra','dec','gmag','rmag','gaia_id','gaia_pmra','gaia_pmdec']
```

Out[73]: *Table length=1083986*

id	ra	dec	gmag	rmag	gaia_id	gaia_pmra	gaia_pmdec
str12	float64	float64	float64	float64	float64	float64	float64
100932_4011	227.7597927444488	-1.661063612401518	16.892656	16.580189	4.4181418083845335e+18	-6.420351761303554	-6.538589286338878
100932_3941	227.77669782804327	-1.661369626621191	22.232037	21.829666	--	--	--
100932_21517	227.77697345090672	-1.6584101713965873	21.328125	99.99	--	--	--
100932_17737	227.7799224207096	-1.666327862284693	99.99	24.253906	--	--	--
100932_16047	227.7816845582915	-1.6623432360245642	99.99	23.913057	--	--	--
100932_17738	227.78255724704024	-1.6624007634486564	99.99	24.09375	--	--	--
100932_11107	227.7790512767691	-1.6579044696491856	24.227655	23.795732	--	--	--
100932_18144	227.78323409615825	-1.6582002448409552	99.99	24.15625	--	--	--
100932_19753	227.79630420135416	-1.6901435643869969	99.99	21.03125	--	--	--
100932_11575	227.81450286116416	-1.6584341166750436	23.06798	23.049318	--	--	--
...	...	...	...	...	...	...	...
97352_14726	230.90978123468284	0.5443098601099531	99.99	23.979742	--	--	--
97352_16199	230.9087660878228	0.5465201697274901	99.99	99.99	--	--	--
97352_16261	230.91151785435108	0.5435519478799269	99.99	99.99	--	--	--
97352_6860	230.908133155737	0.5508180754677512	99.99	24.282377	--	--	--
97352_12846	230.90992763917865	0.5488119963645425	24.616777	24.085495	--	--	--
97352_13277	230.90707479380373	0.5532871646298039	23.943844	23.816998	--	--	--
100419_5179	227.4378705498005	-1.327959847796134	20.62322	19.278862	4.4183530821199027e+18	-58.30517061556114	19.142560985262616
100419_10934	227.43921843407568	-1.327951333856278	23.846416	23.256102	--	--	--
100419_8772	227.43889829427644	-1.3250560232813309	23.933619	23.763311	--	--	--
100419_5183	227.4377050729523	-1.324957310197574	21.971382	21.260548	--	--	--



Notebooks -> Production Code -> Packages

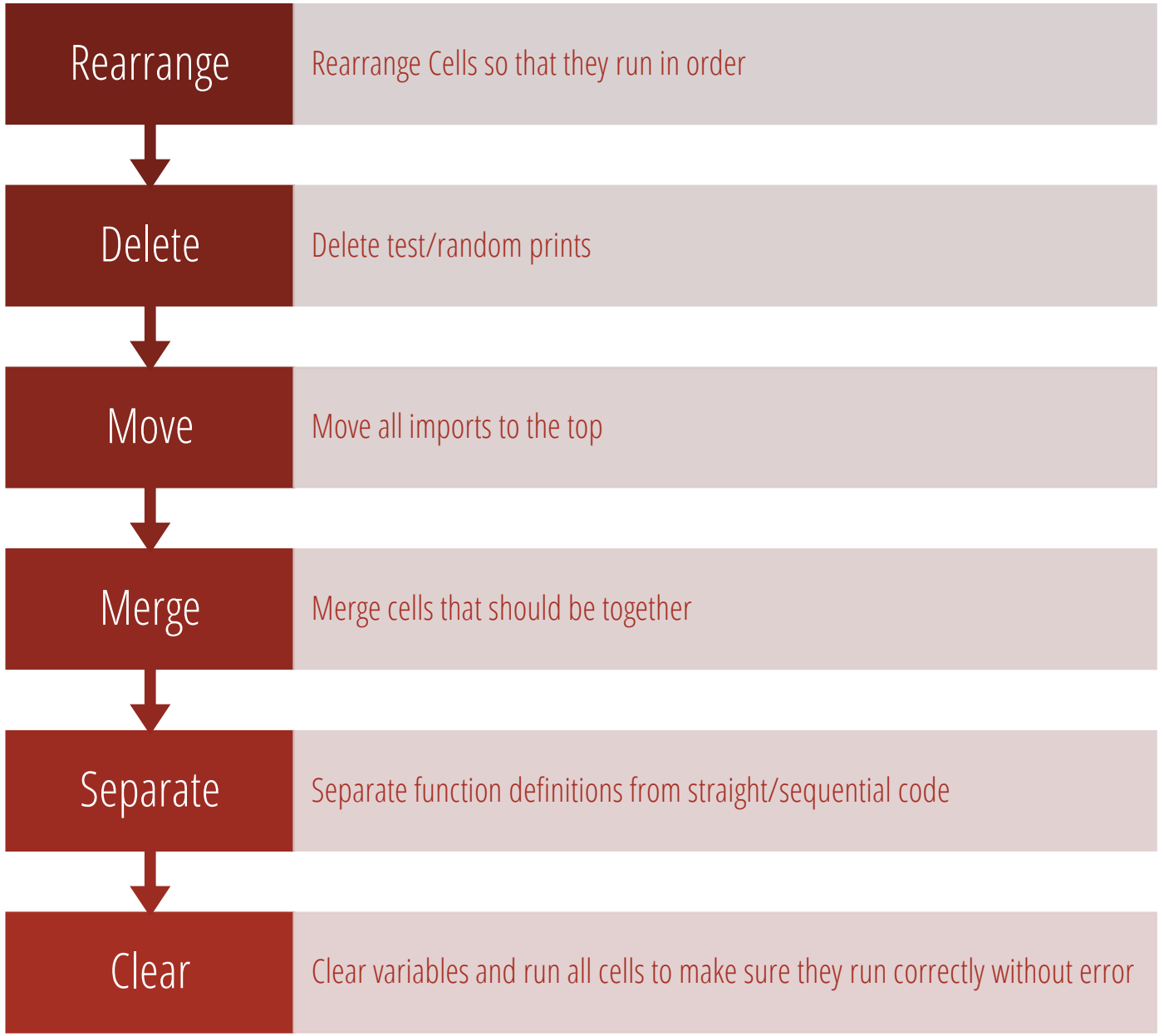
Why Notebooks aren't everything

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

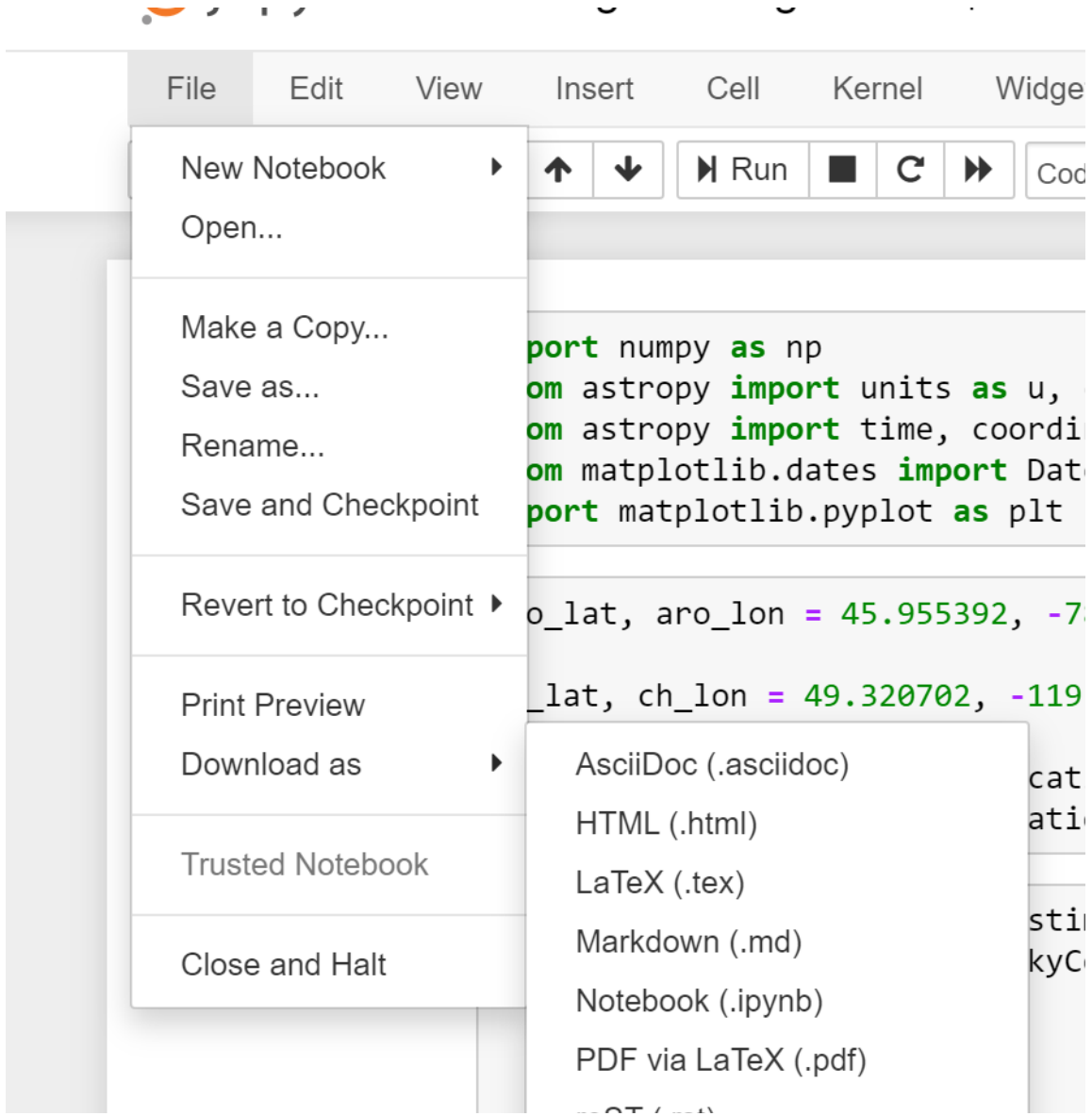
- Not reusable
- Difficult to Automate
- Missing debugging and advanced development tools
- Difficult to Scale

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

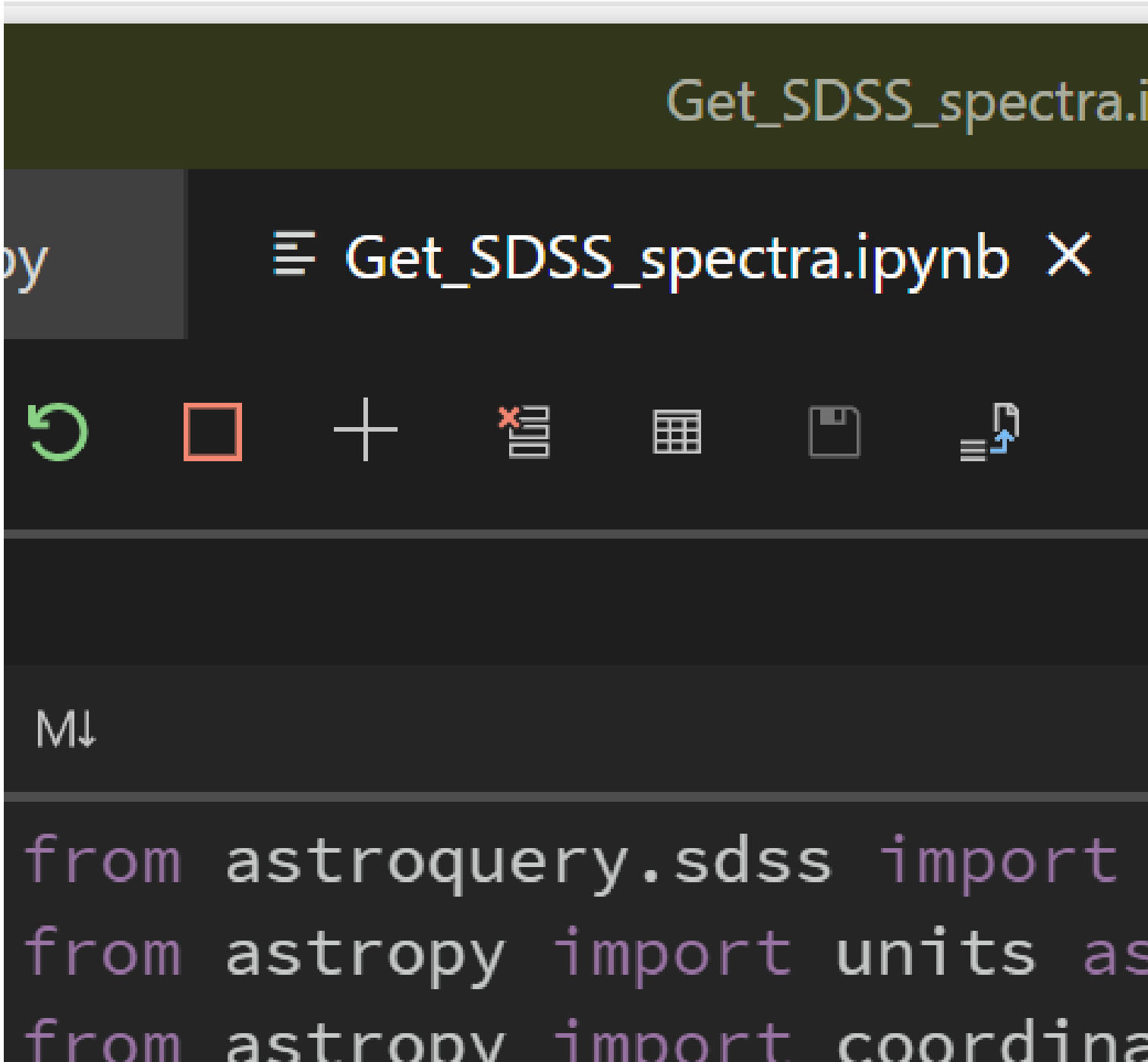
Getting a notebook ready for conversion



Converting a Notebook to a Script (via Jupyter)



Converting a Notebook to a Script (via VS Code)

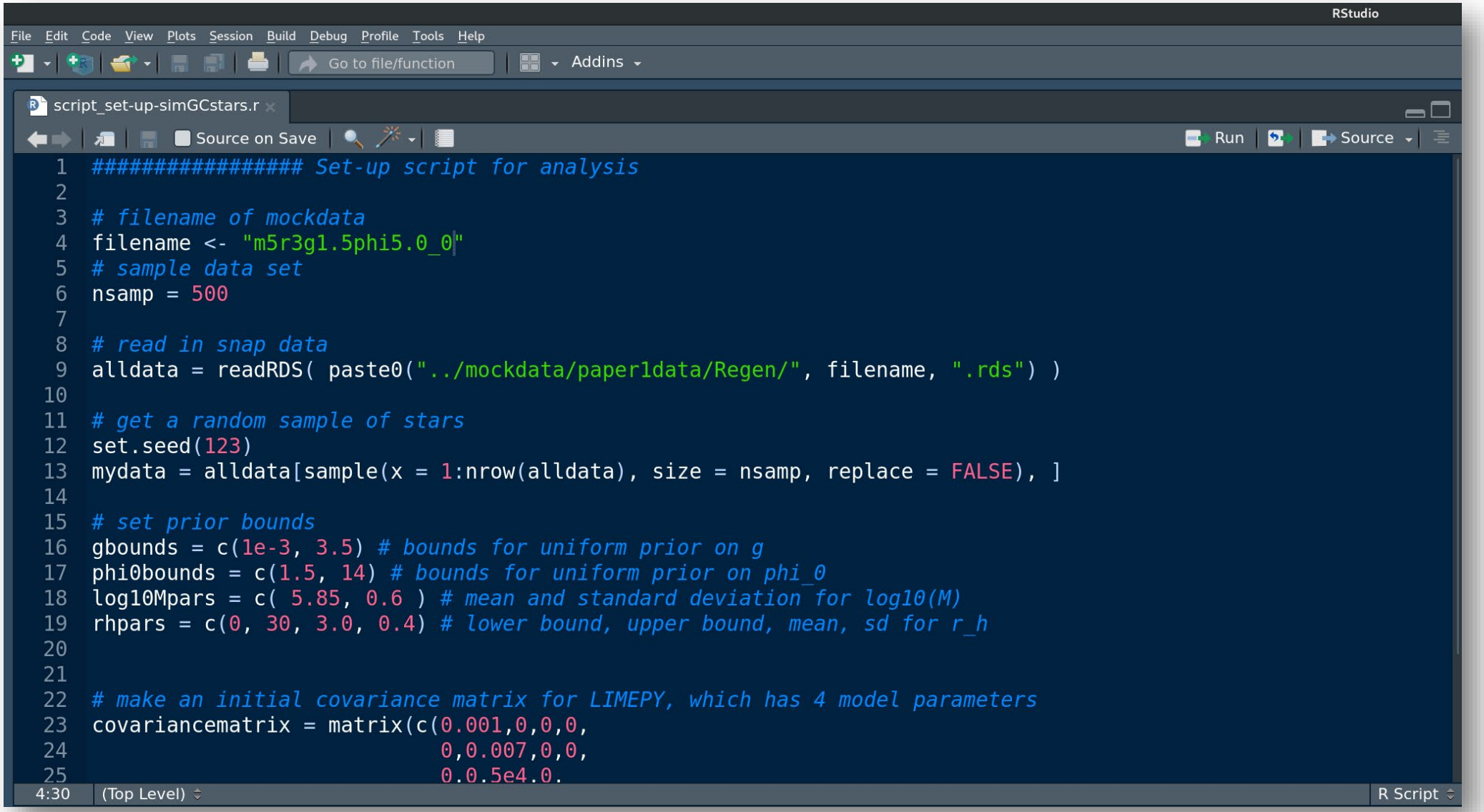


Going from R-script to R markdown notebook

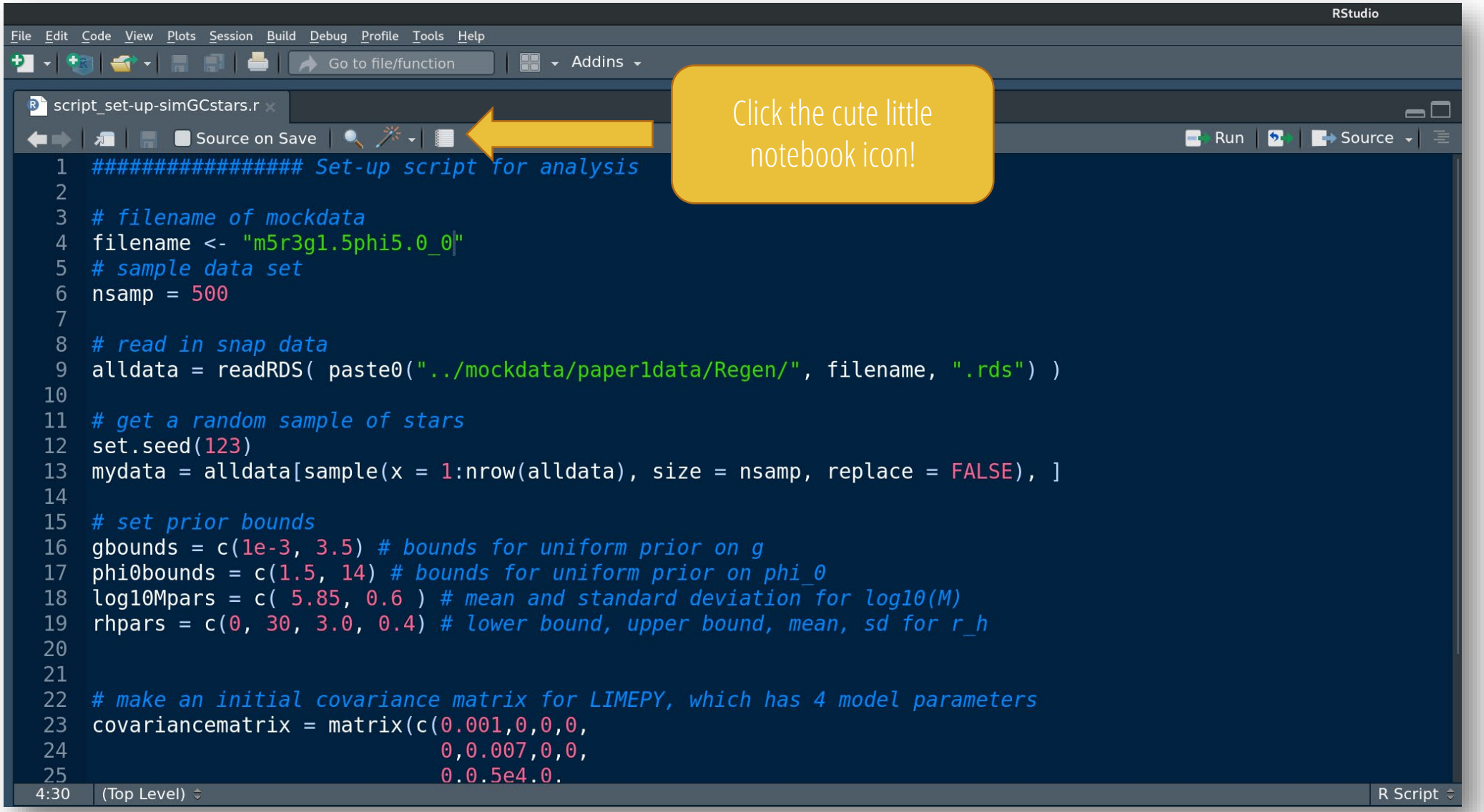
- Situation:
  - You wrote an R script to do some stuff and make a plot, and your supervisor/collaborator wants to understand it quickly.
  - You want to give them an easy-to-read "report" in html, pdf, etc.

- 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.

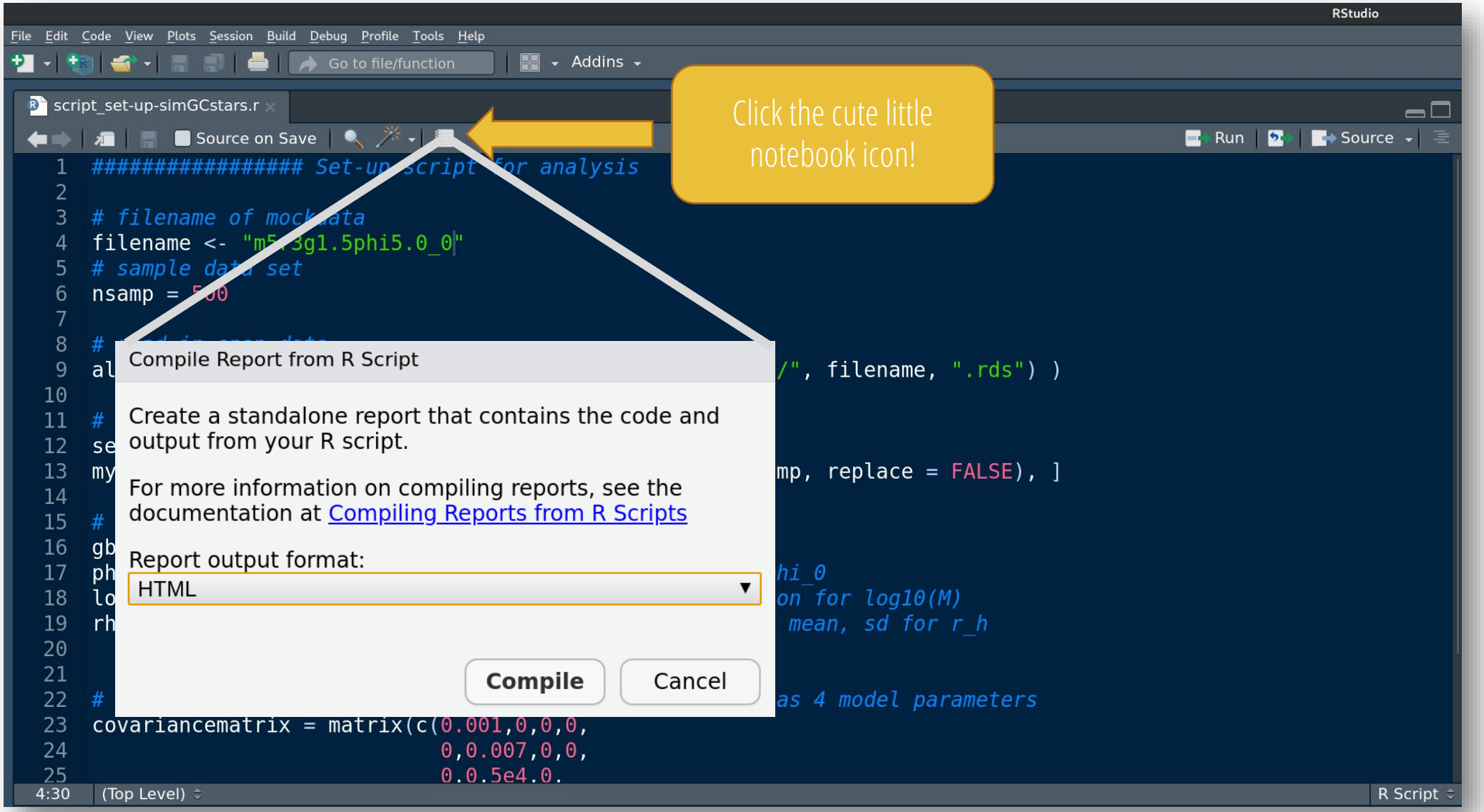
Going from R-script to R markdown notebook



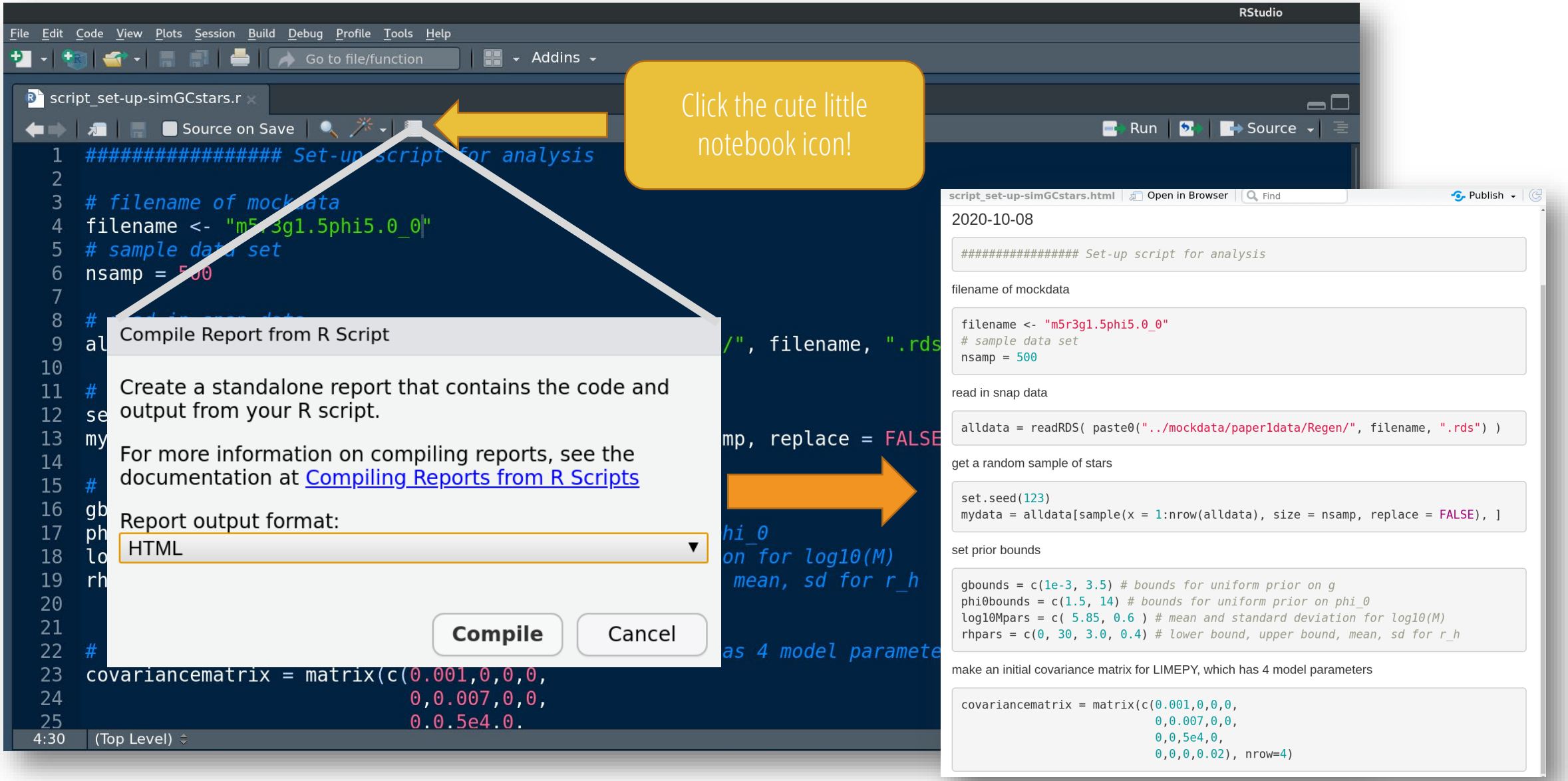
Going from R-script to R markdown notebook



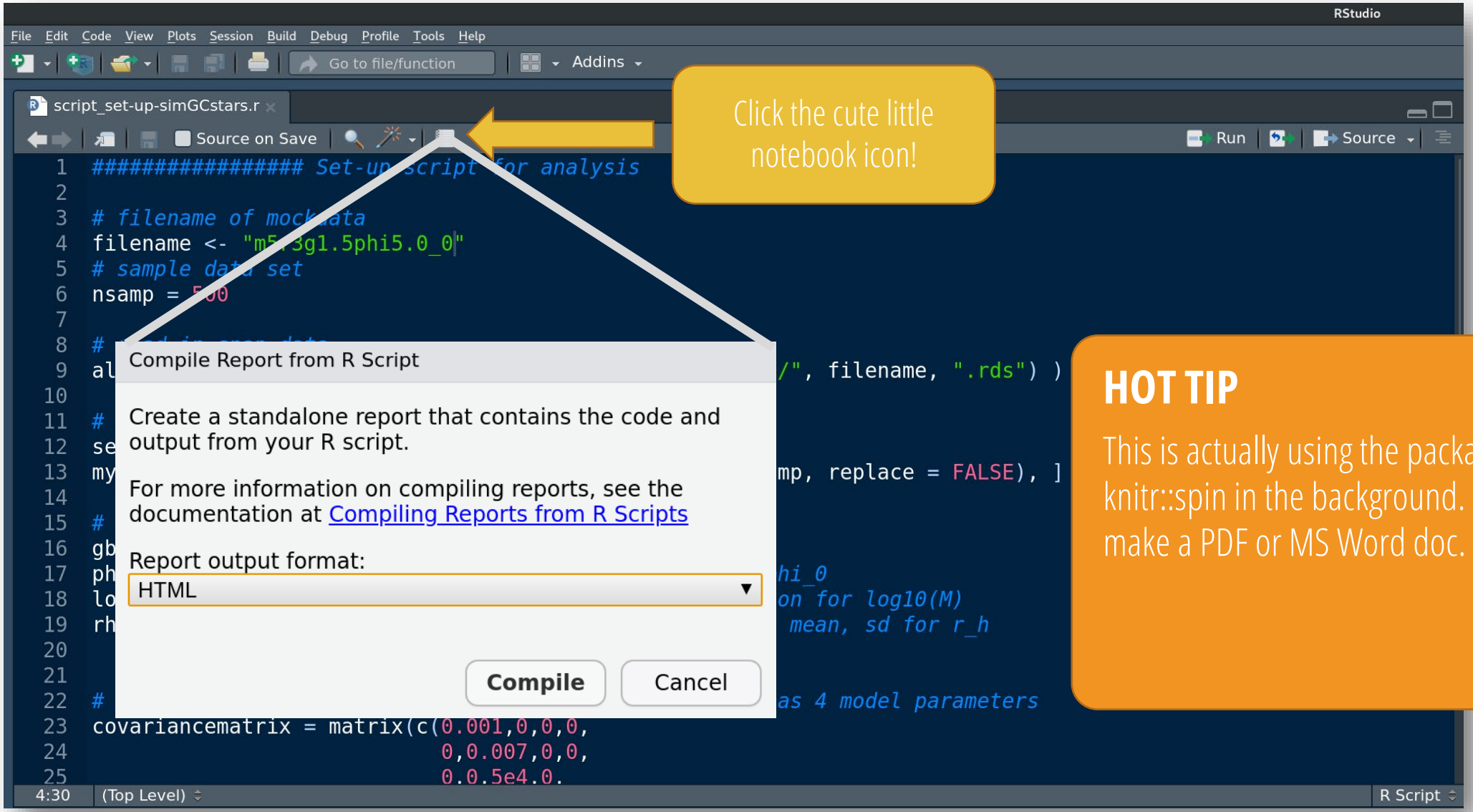
Going from R-script to R markdown notebook



Going from R-script to R markdown notebook



Going from R-script to R markdown notebook



HOT TIP

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

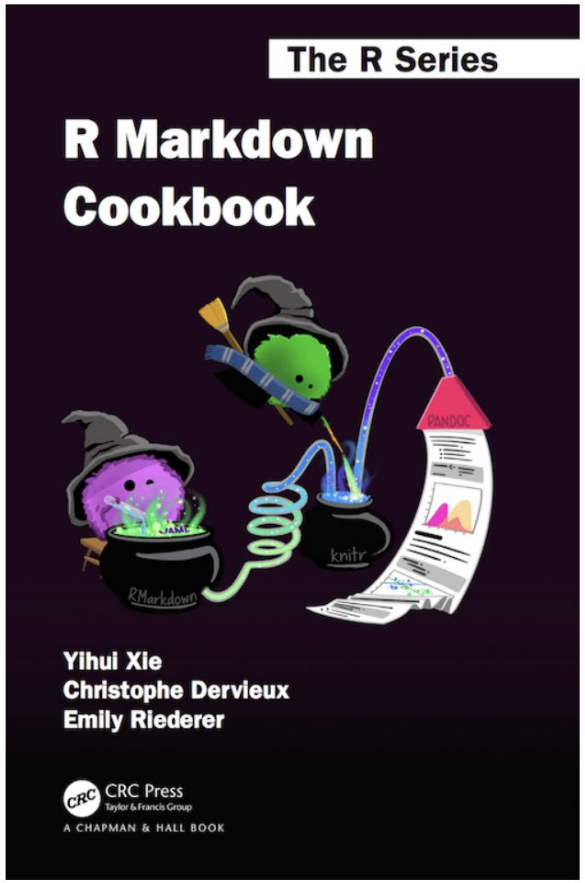
You can do the reverse too!

- 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

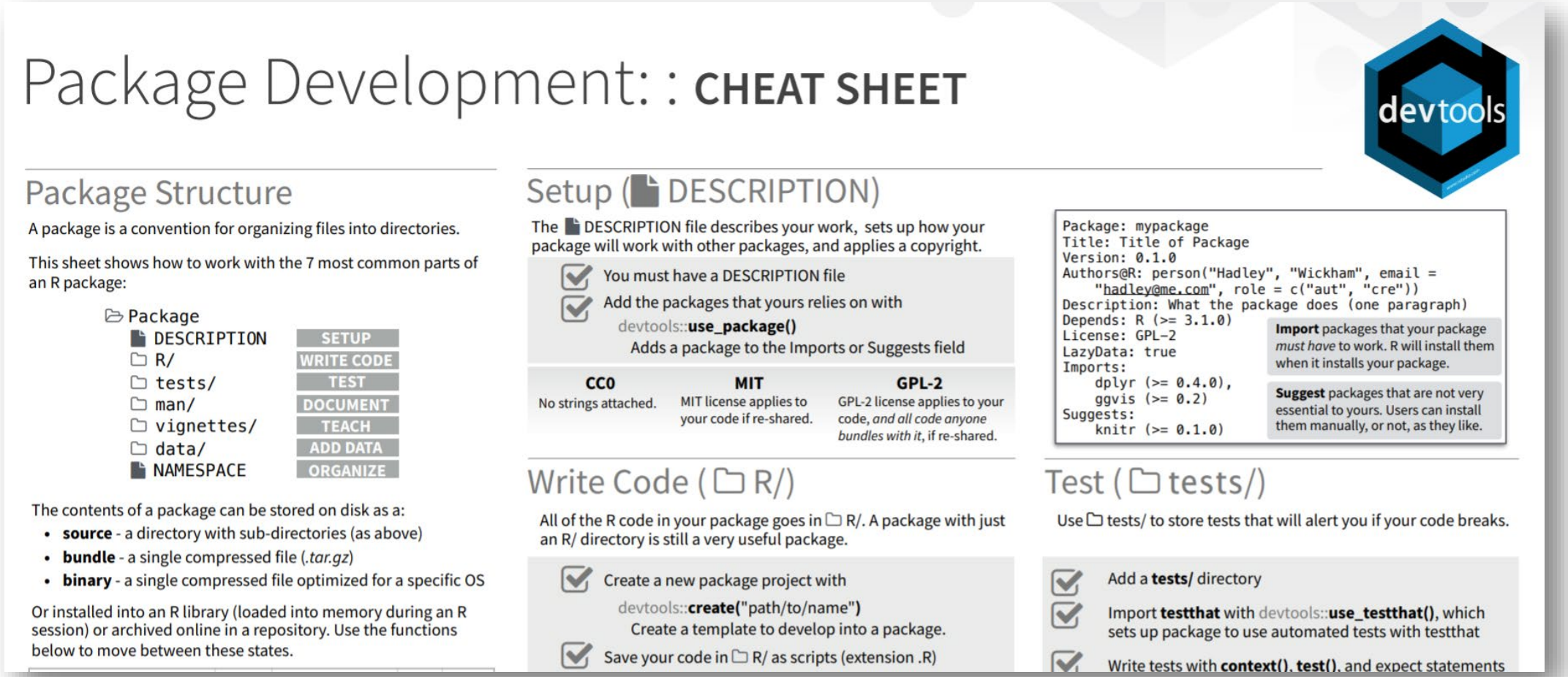
Packages are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sample data. In this book you'll learn how to turn your code into packages that others can easily download and use. Writing a package can seem overwhelming 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)

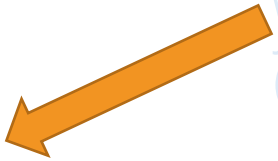
Unexpected Behaviour: Code runs (or runs to a point), but doesn't do what you expect it to.
(Includes incorrect data types, missing variables, problems with flow control)

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

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



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

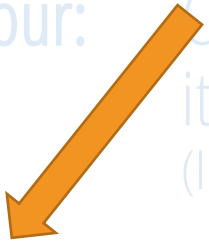
Unexpected Behaviour: Code runs (or runs to a point), but doesn't do what you expect it to.
(Includes incorrect data types, missing variables, problems with flow control)

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 (sometimes crashes), but does something that you expect it to.
(Includes incorrect data types, missing variables, problems with flow control)

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



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, missing variables, problems with flow control)

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

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.

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-bcba7c[illegible]d330c> in divide_by_number(num)
      1 def divide_by_number(num):
----> 2     return 1./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

The line number where the problem 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-bcba7c1d330c> in divide_by_number(num)
      1 def divide_by_number(num):
----> 2     return 1.0/num
      3

ZeroDivisionError: float division by zero
```

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

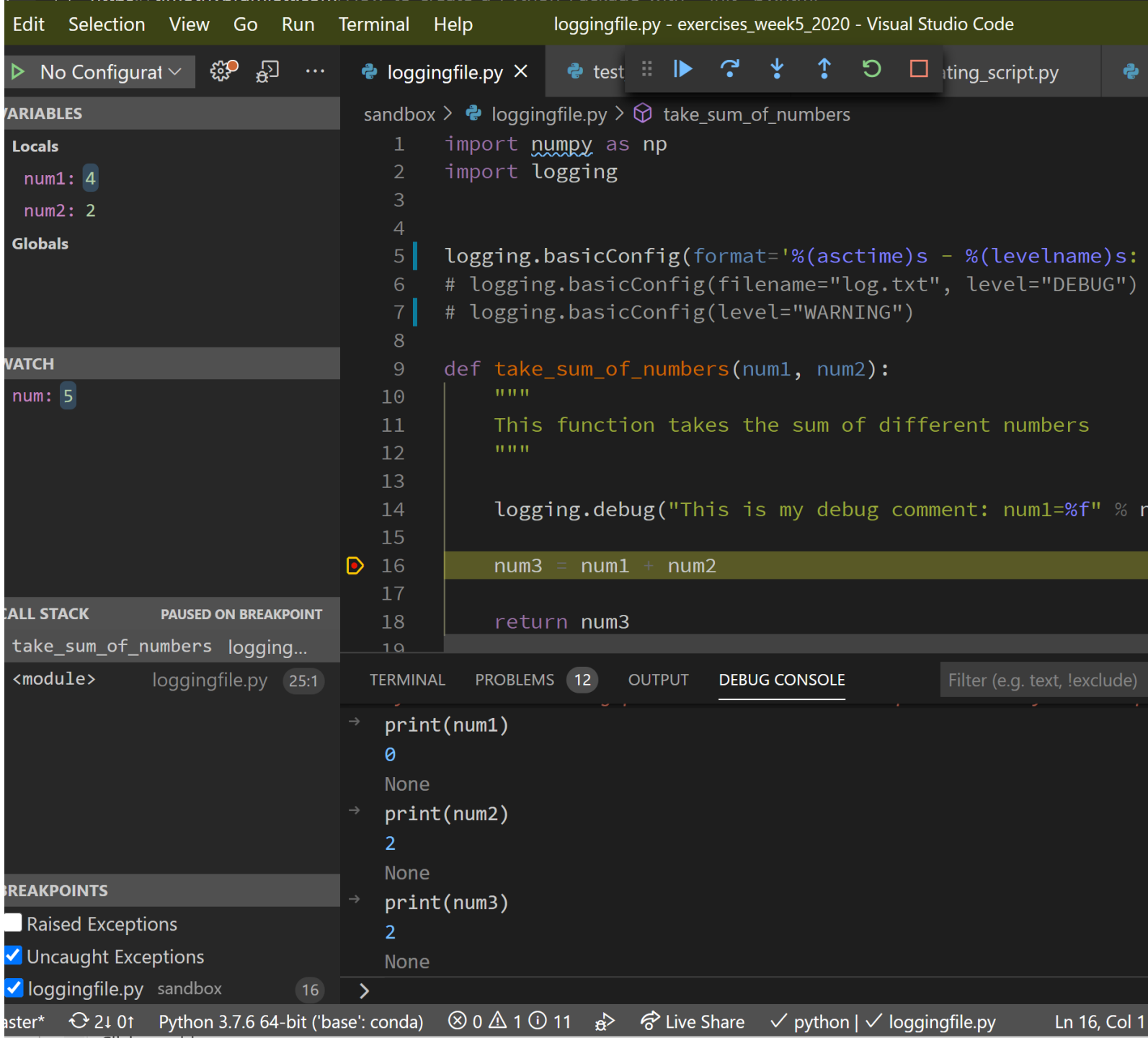
In R, the equivalent package is called `testthat`

Debugging via VS Code

Breakpoints

Conditional Breakpoints

Logging

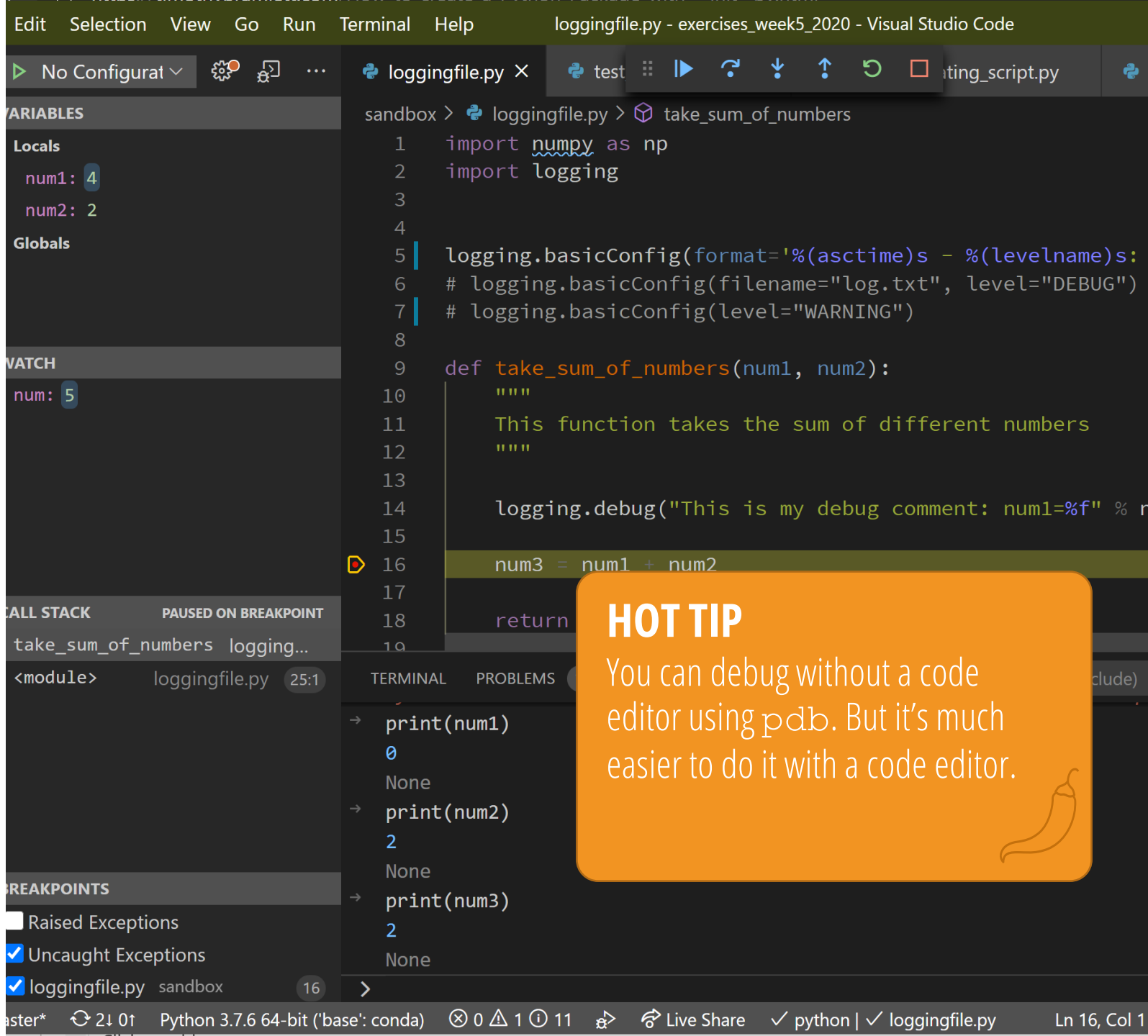


Debugging via VS Code

Breakpoints

Conditional Breakpoints

Logging



Debugging via VS Code

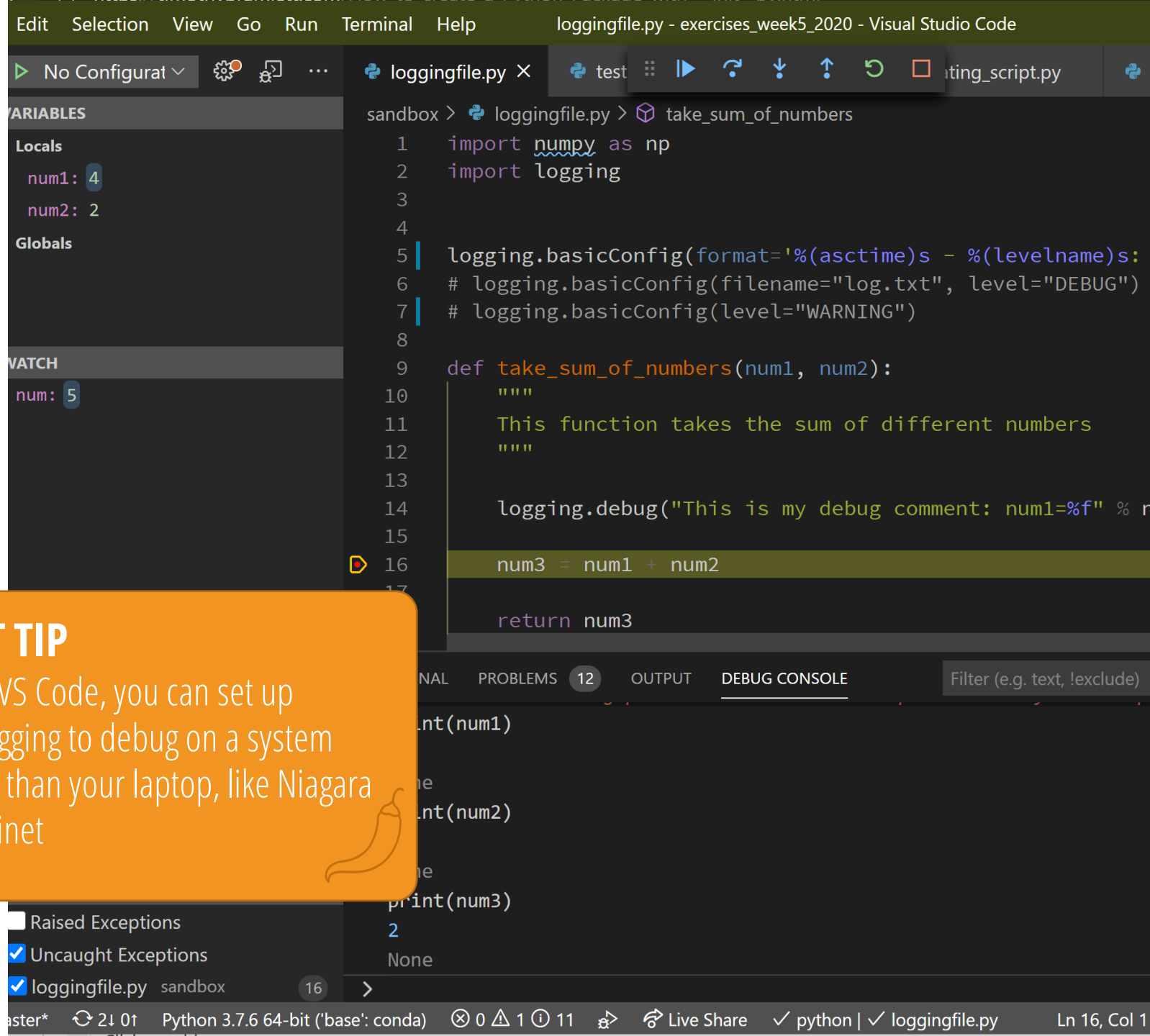
Breakpoints

Conditional Breakpoints

Logging

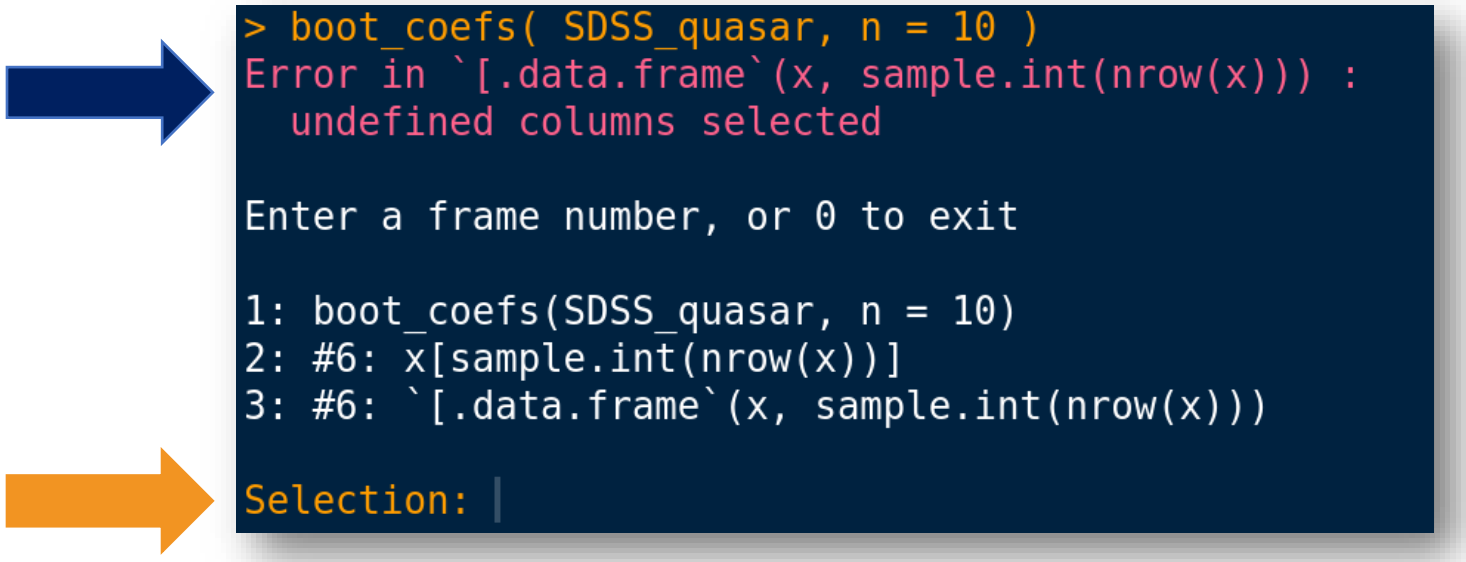
HOT TIP

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()`



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

Query Gaia data with astroquery in Jupyter Notebook

- pip install astroquery if you do not have it installed
- Select the first 100000 stars in Gaia DR3 (use gaiadr3.gaia_source table, see schema via this link) with the following requirement
  - parallax_over_error > 10 (so you select stars with parallax error less than 10%)
  - phot_g_mean_mag < 17 (so relative bright)
  - RA between 30 and 40
  - Dec between -50 and -40
- Convert parallax to distance, and then to distance modulus
- Make an HR diagram with M_G vs BP-RP

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 session4_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`

Visualizing your Data

You have your data, now what?

Ask yourself:

- Do I need to visually represent/plot this data?
- Is this plot answering the question I'm asking?
  - Is this looking for specific values of something, or is this some form of comparison?
  - Are the quantities that I'm plotting the right ones?
- Is this plot hiding something that I should be looking at?

Graphical Perception

Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods

WILLIAM S. CLEVELAND and ROBERT McGILL*

Source: *Journal of the American Statistical Association*, Sep., 1984, Vol. 79, No. 387 (Sep., 1984), pp. 531-554

Published by: Taylor & Francis, Ltd. on behalf of the American Statistical Association

Stable URL: https://www.jstor.org/stable/2288400

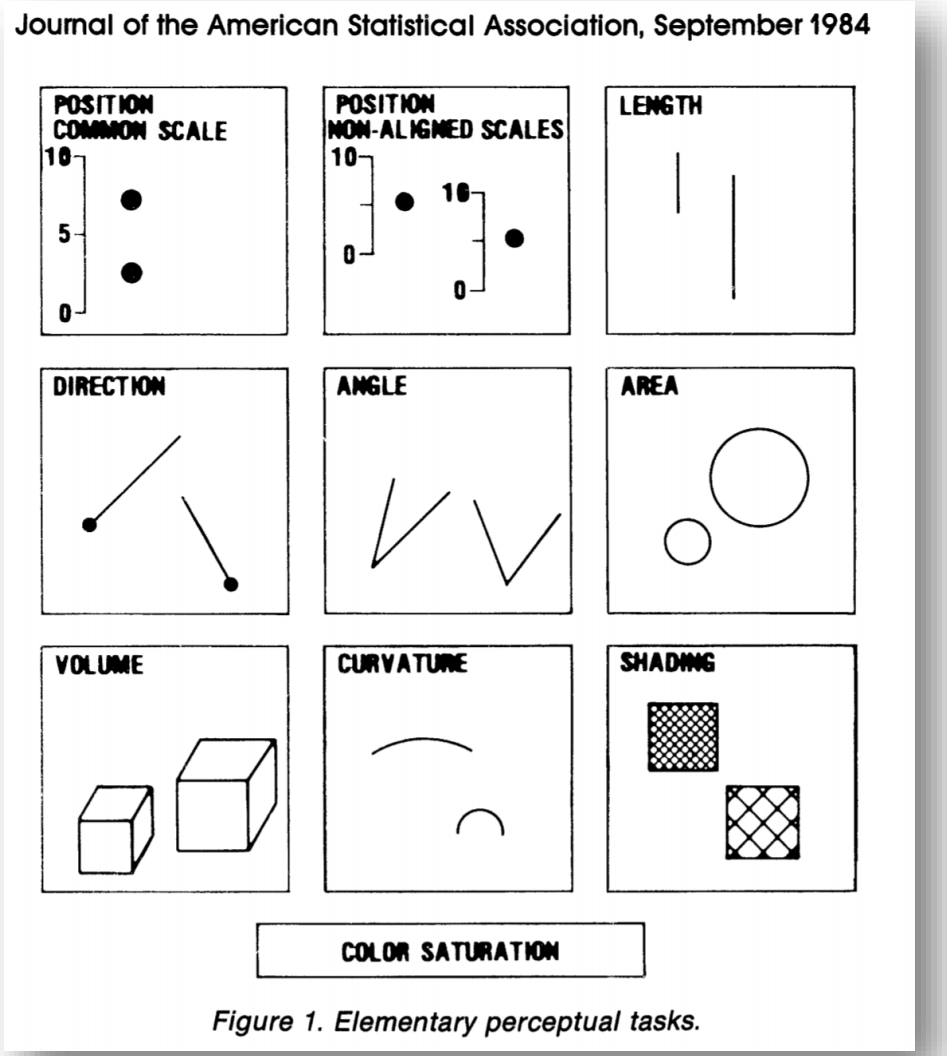


Figure 1. Elementary perceptual tasks.

Graphical Perception

Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods

WILLIAM S. CLEVELAND and ROBERT McGILL*

Source: *Journal of the American Statistical Association*, Sep., 1984, Vol. 79, No. 387 (Sep., 1984), pp. 531-554

Published by: Taylor & Francis, Ltd. on behalf of the American Statistical Association

Stable URL: https://www.jstor.org/stable/2288400

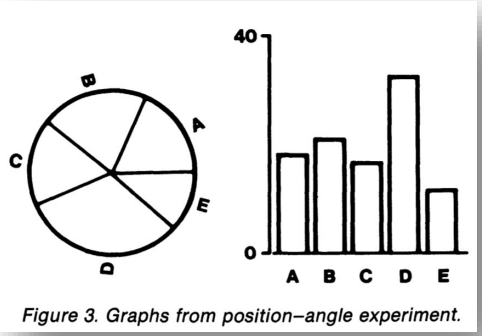


Figure 3. Graphs from position–angle experiment.

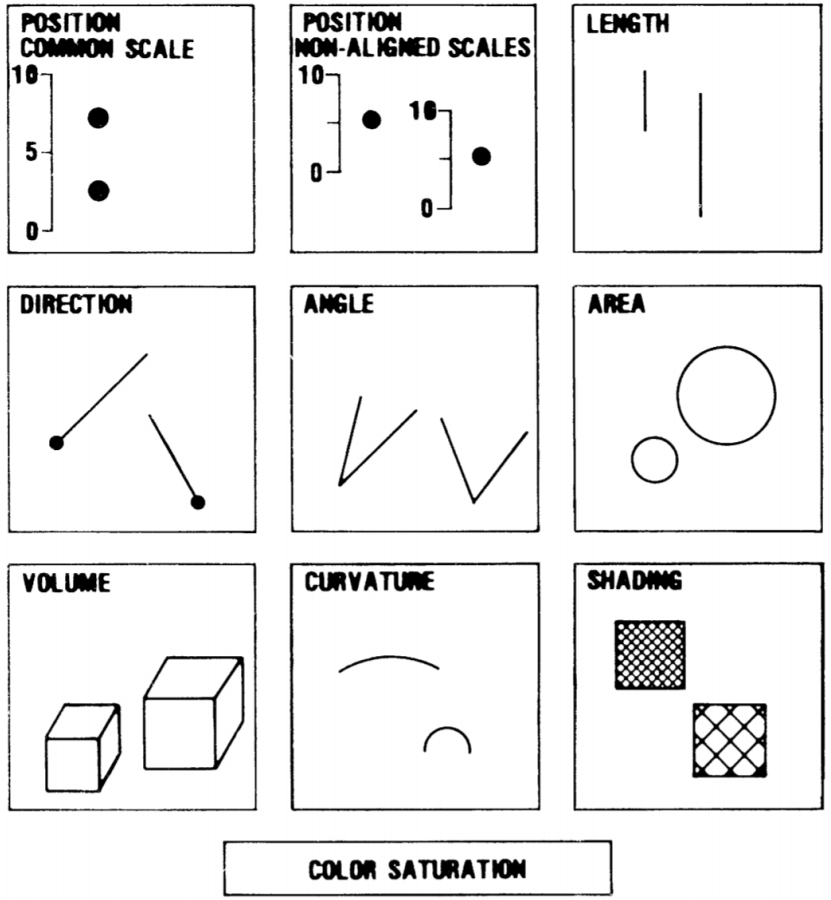


Figure 1. Elementary perceptual tasks.

Graphical Perception

Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods

WILLIAM S. CLEVELAND and ROBERT McGILL*

Source: *Journal of the American Statistical Association*, Sep., 1984, Vol. 79, No. 387 (Sep., 1984), pp. 531-554

Published by: Taylor & Francis, Ltd. on behalf of the American Statistical Association

Stable URL: https://www.jstor.org/stable/2288400

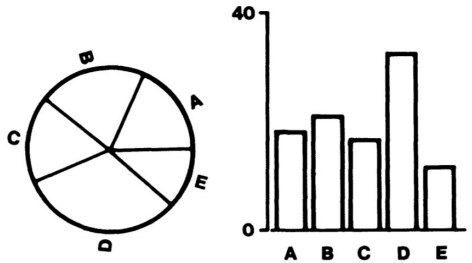


Figure 3. Graphs from position–angle experiment.

The following are the 10 elementary tasks in Figure 1, ordered from most to least accurate:

1. Position along a common scale
2. Positions along nonaligned scales
3. Length, direction, angle
4. Area
5. Volume, curvature
6. Shading, color saturation

Journal of the American Statistical Association, September 1984

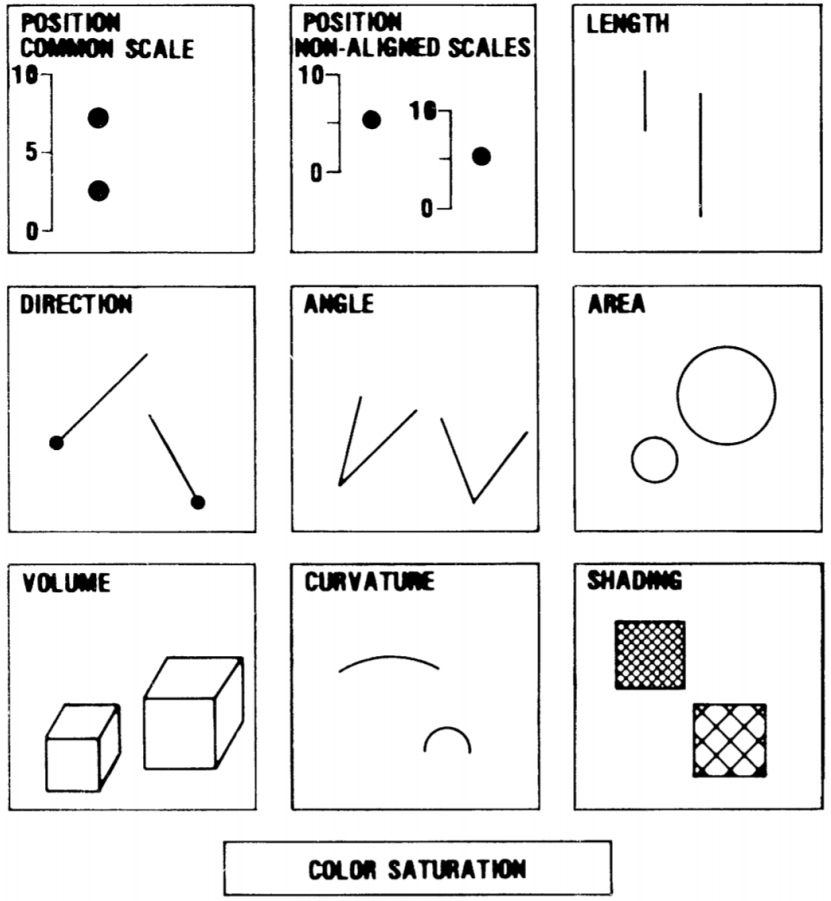


Figure 1. Elementary perceptual tasks.

The Grammar of Visual Elements

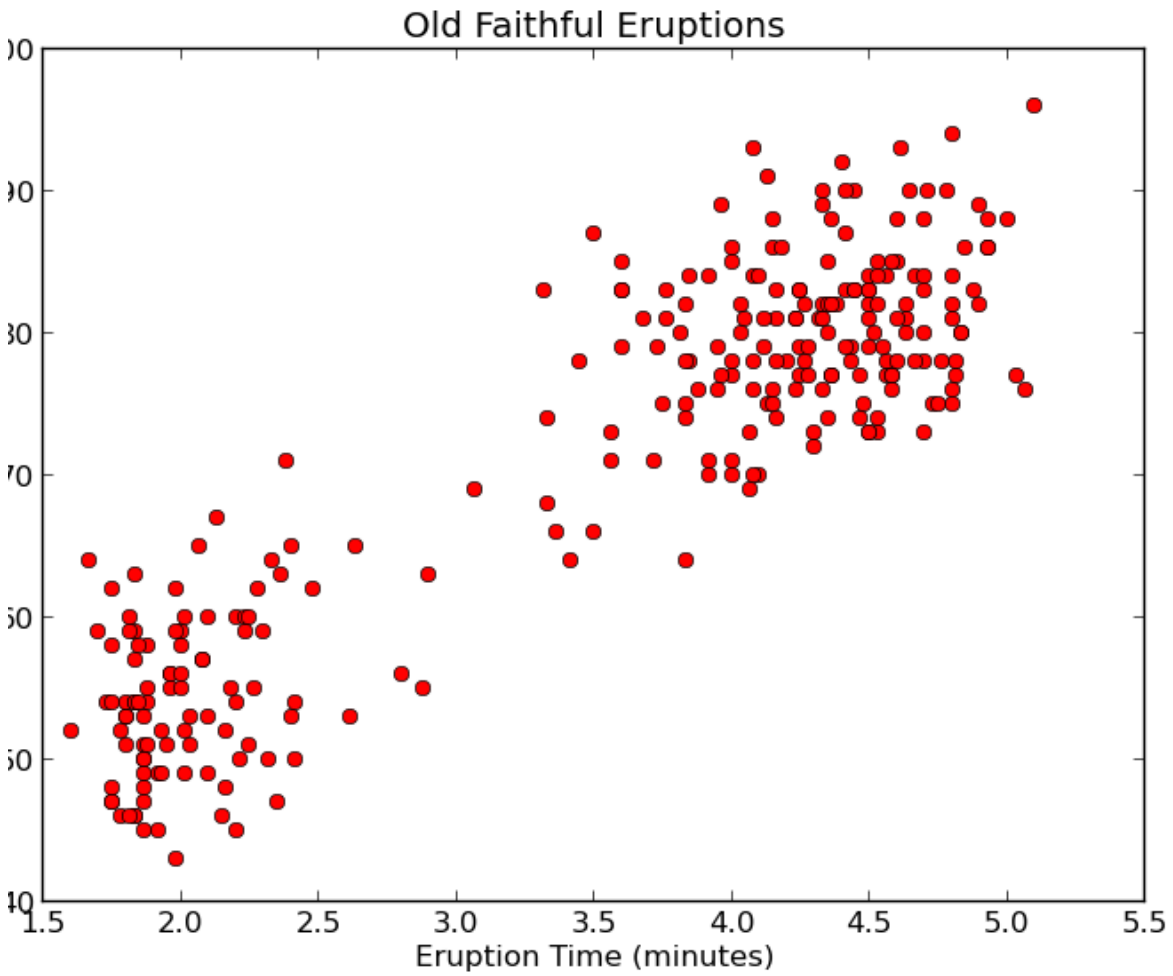
Ways of encoding information in visual form

- Position
- Size
- Shape
- Thickness
- Colour
- Opacity

Position

Representation of two dimensions easily

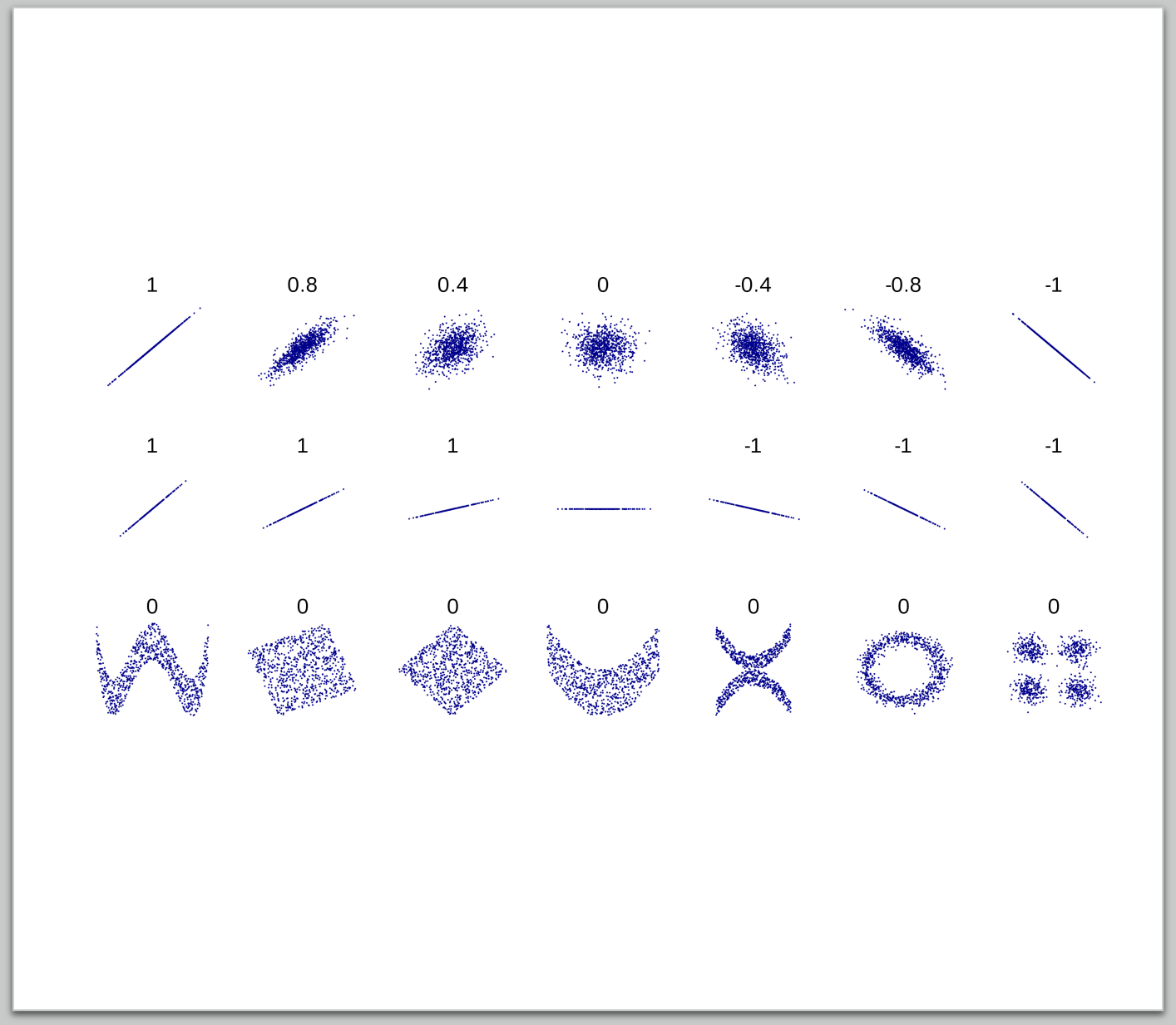
Strong cultural convention



Position

Representation of two dimensions easily

Strong cultural convention



Size

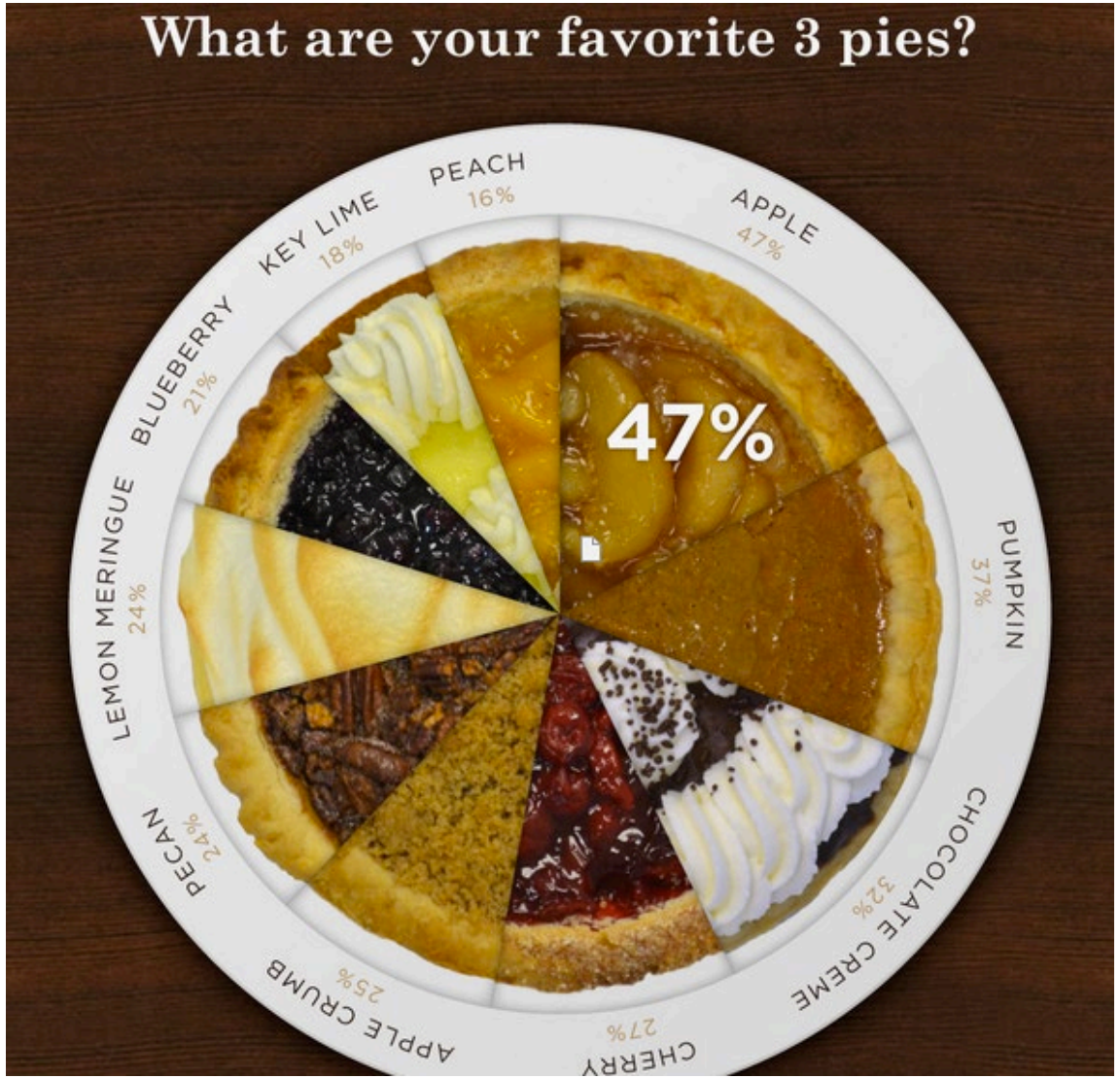
Area as an indication of magnitude

Easy Comparison between elements

Size

Area as an indication of magnitude

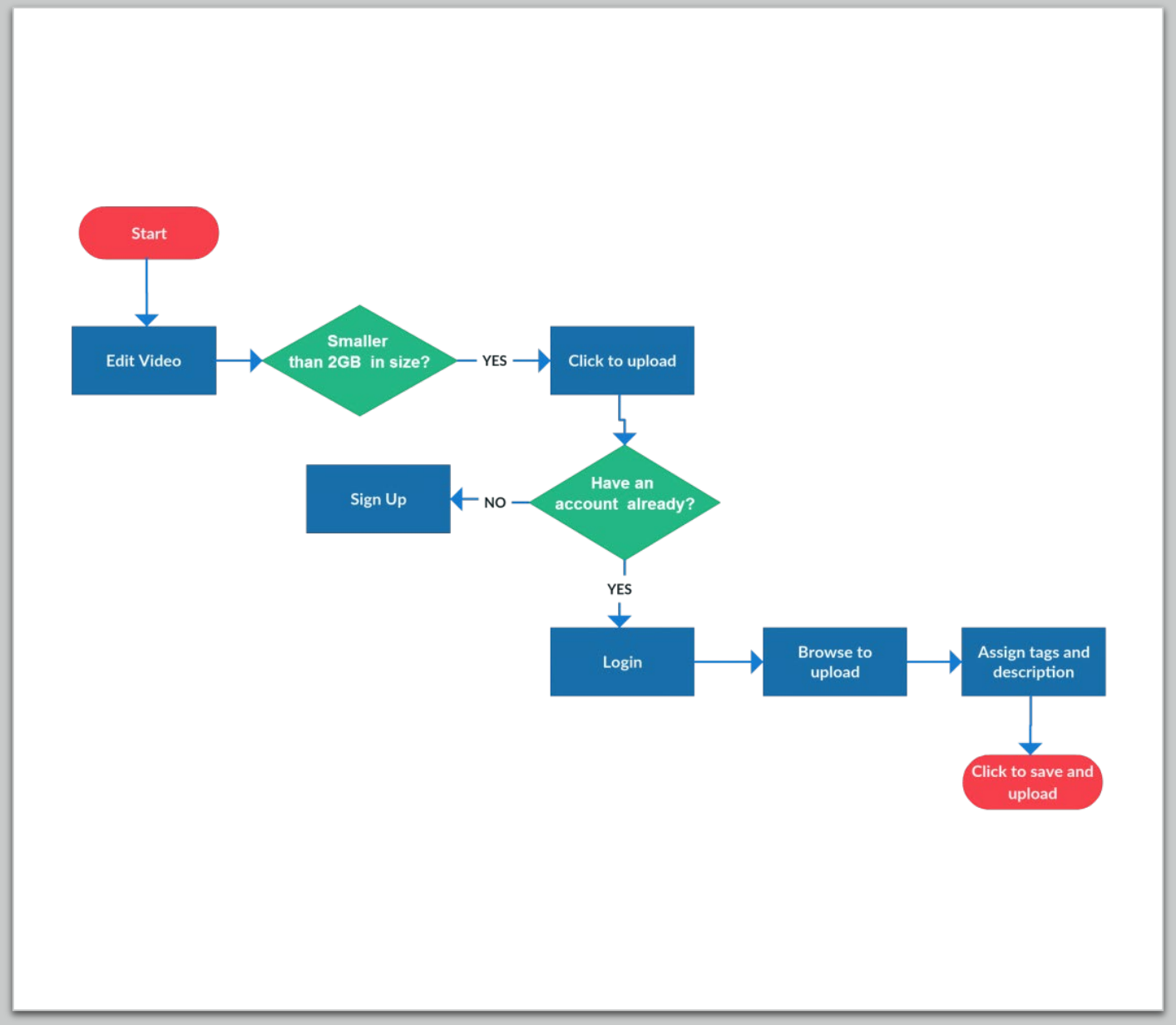
Easy Comparison between elements

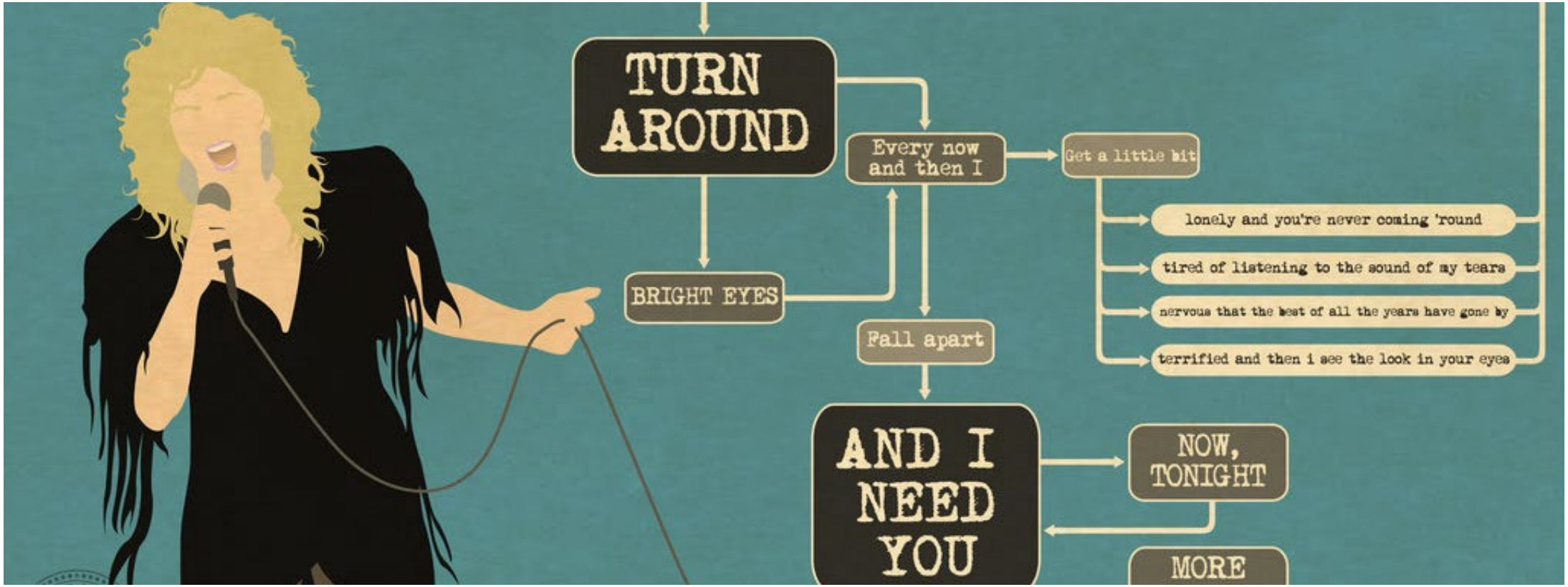


Shape

Can encode categorization information

Can use mappings or cultural conventions





Shape

Can encode categorization information

Can use mappings or cultural conventions

Thickness

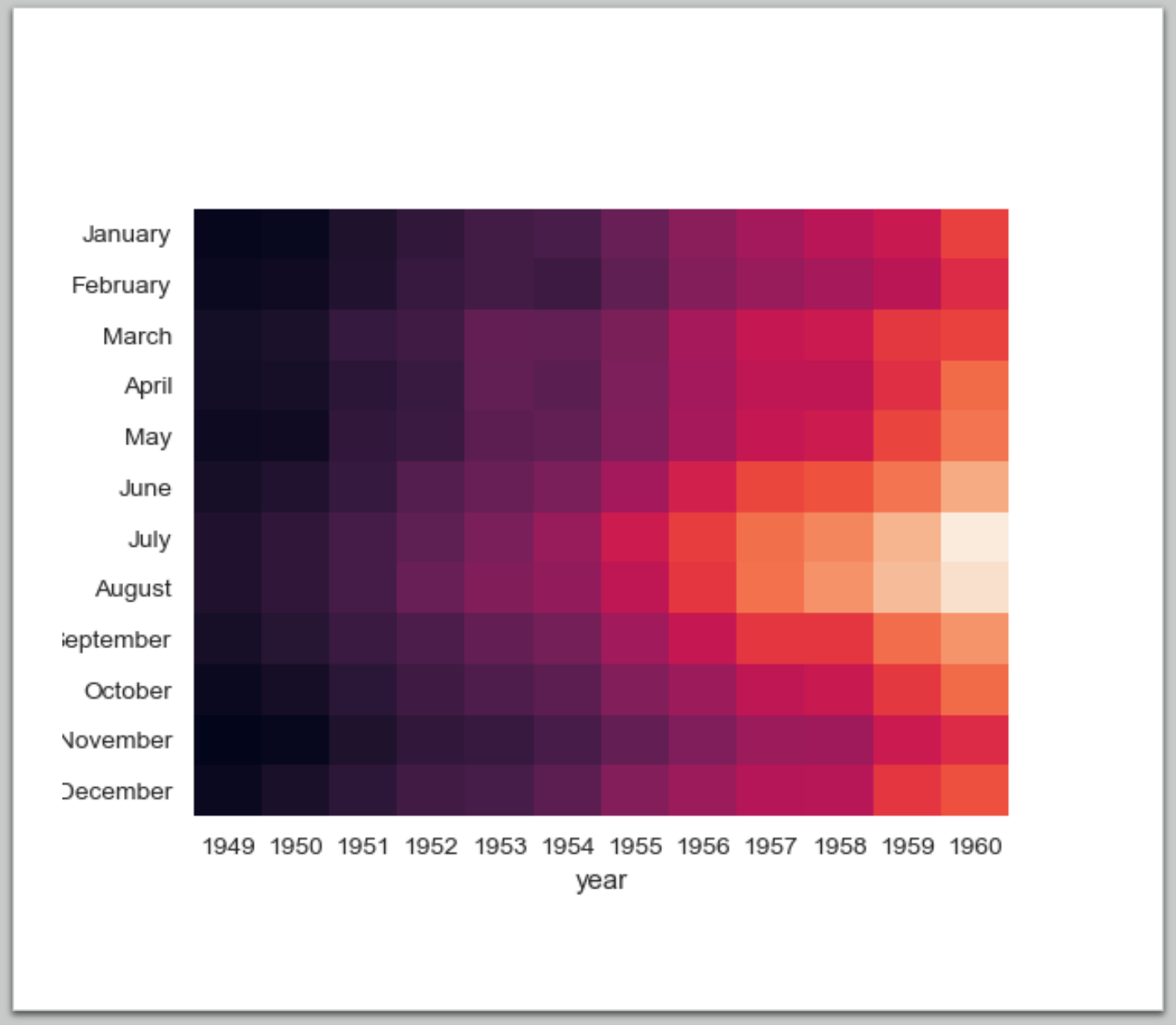
Underutilized but highly effective

Provide a means to categorize visual elements

Colour

Encodes magnitude with natural comparative properties

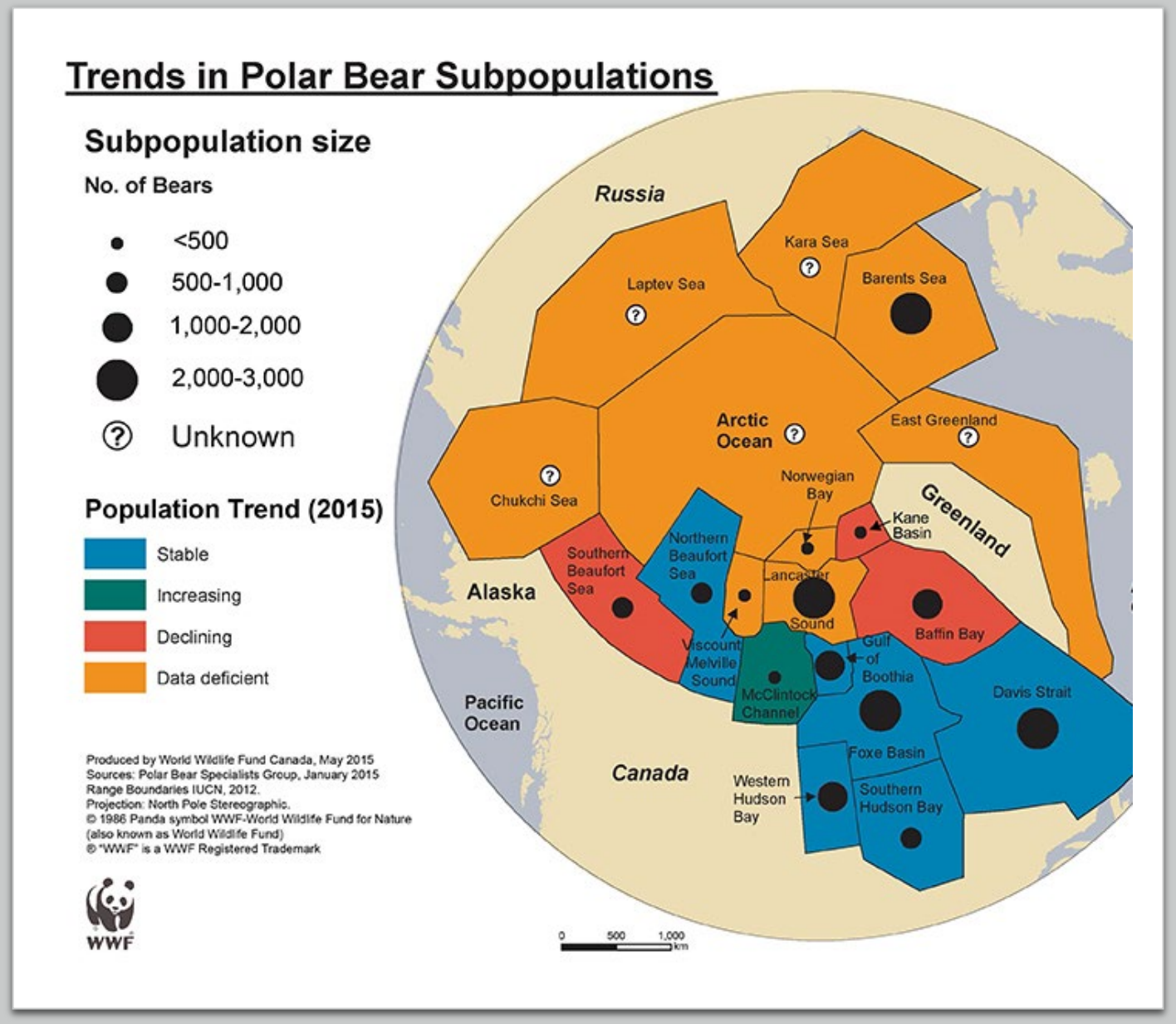
Potential to encode large ranges and types of variation



Colour

Encodes magnitude with natural comparative properties

Potential to encode large ranges and types of variation



Opacity

- Often used in conjunction with colour
- Provides crude information on magnitude

The Three Layers of a Plot

Highest Level: Getting the basic story out quickly

Intermediate Level: Justifying/confirming the story

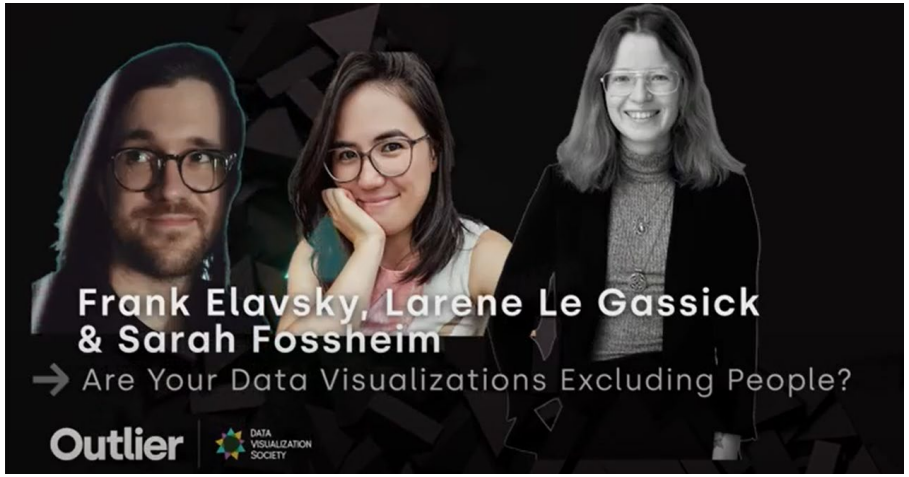
Deepest Level: Being able to reproduce the story

Visualization Accessibility

"When we don't intentionally include, we unintentionally exclude."

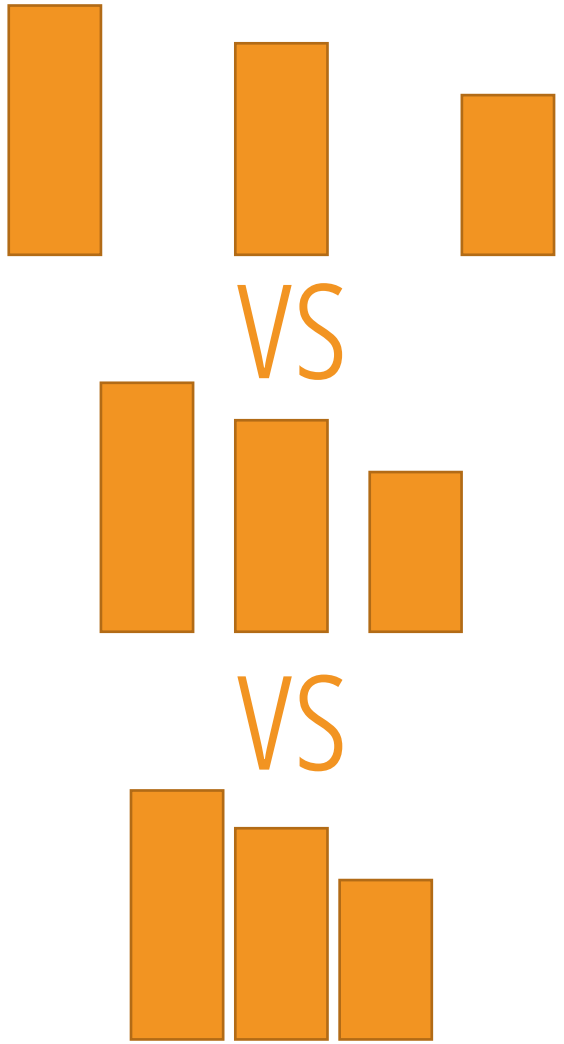
Visualization Accessibility

"When we don't intentionally include, we unintentionally exclude."

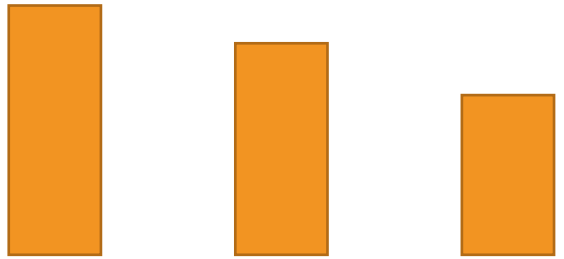


https://youtu.be/SWB-KLXN-Ok

Visualization Accessibility

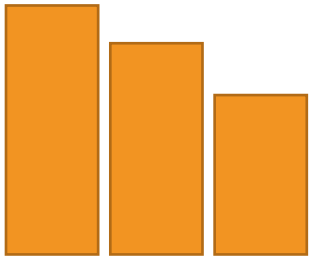


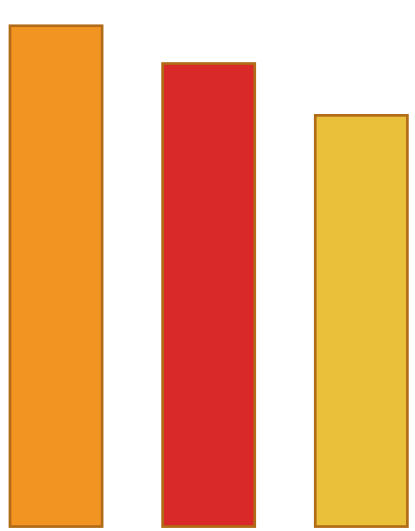
Visualization Accessibility



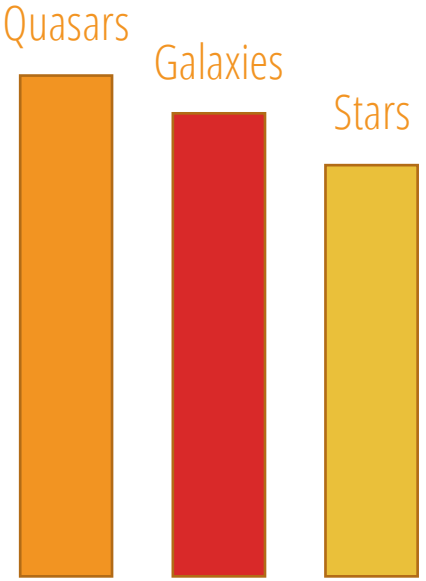
VS

VS

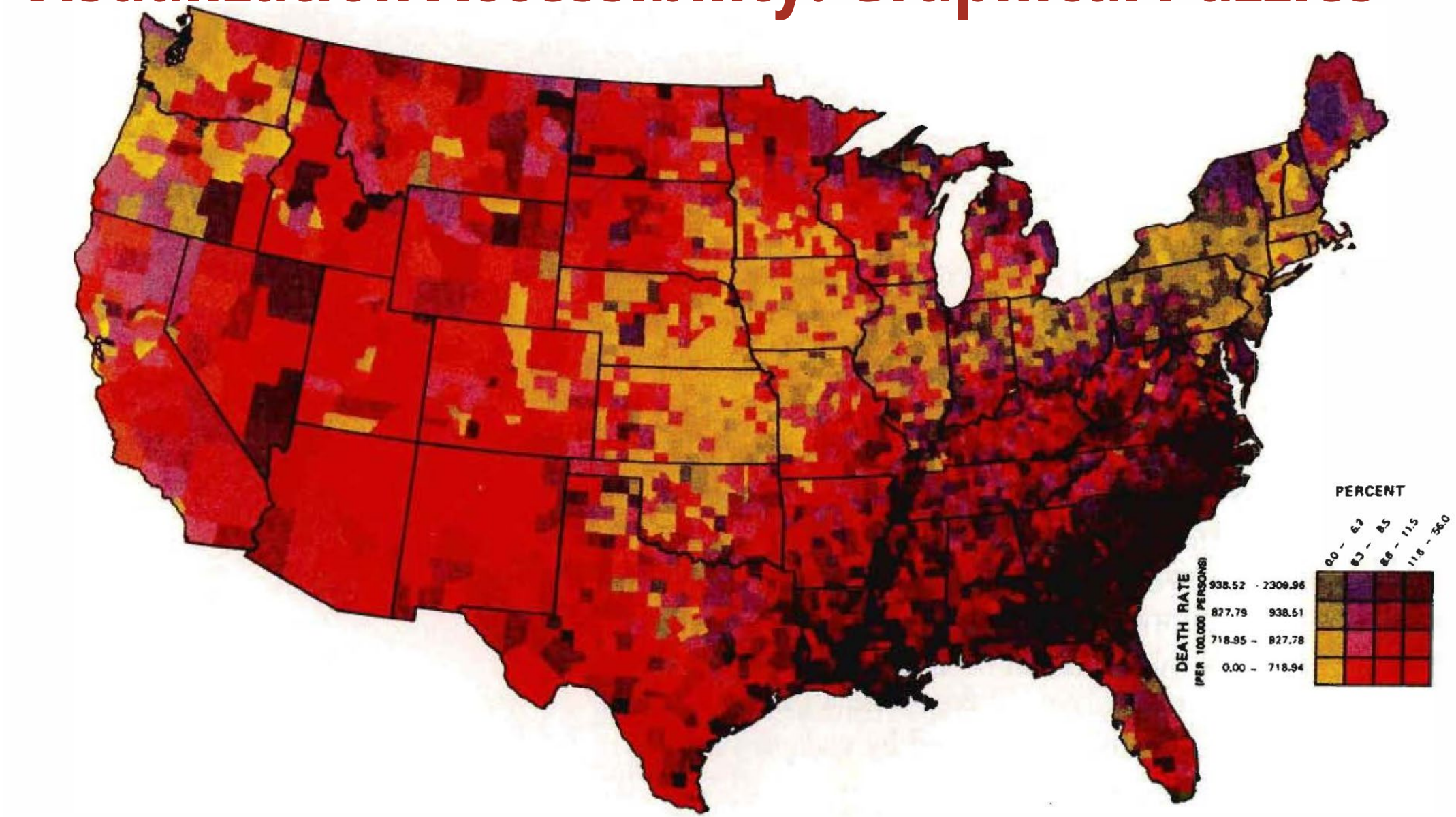


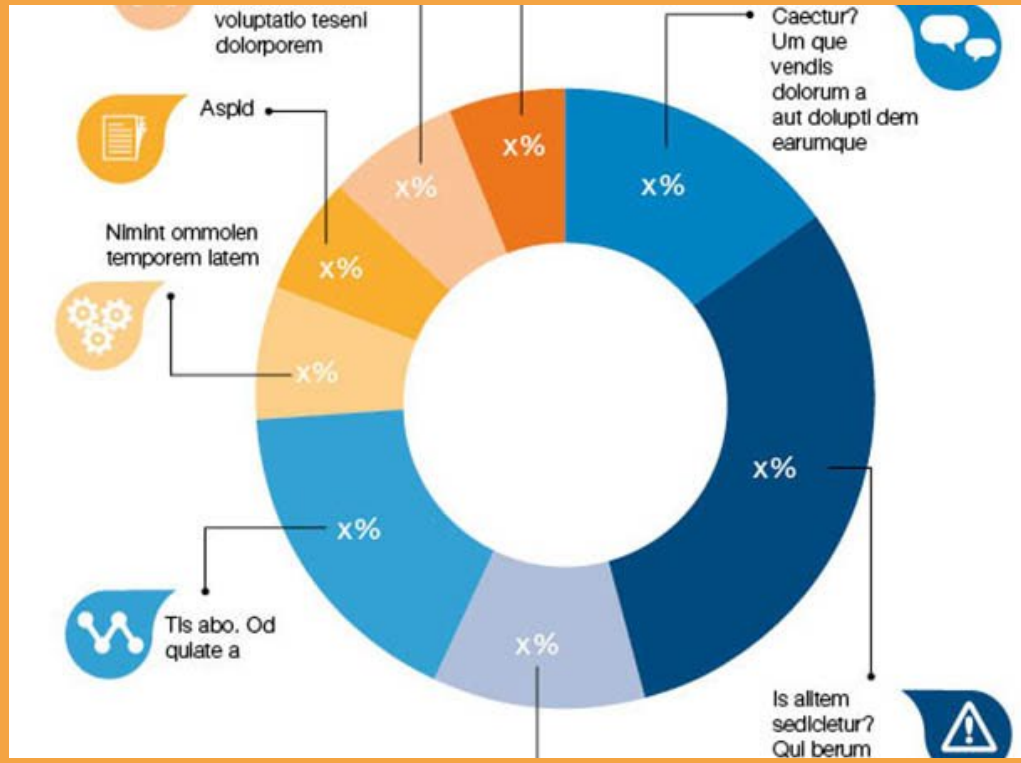


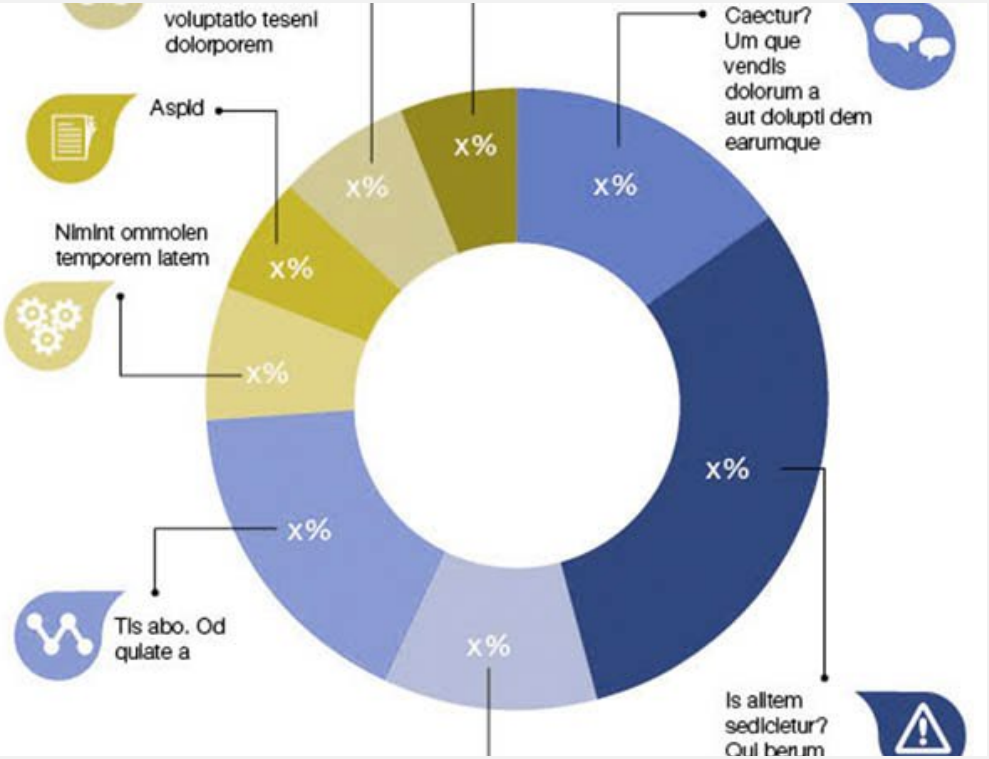
VS



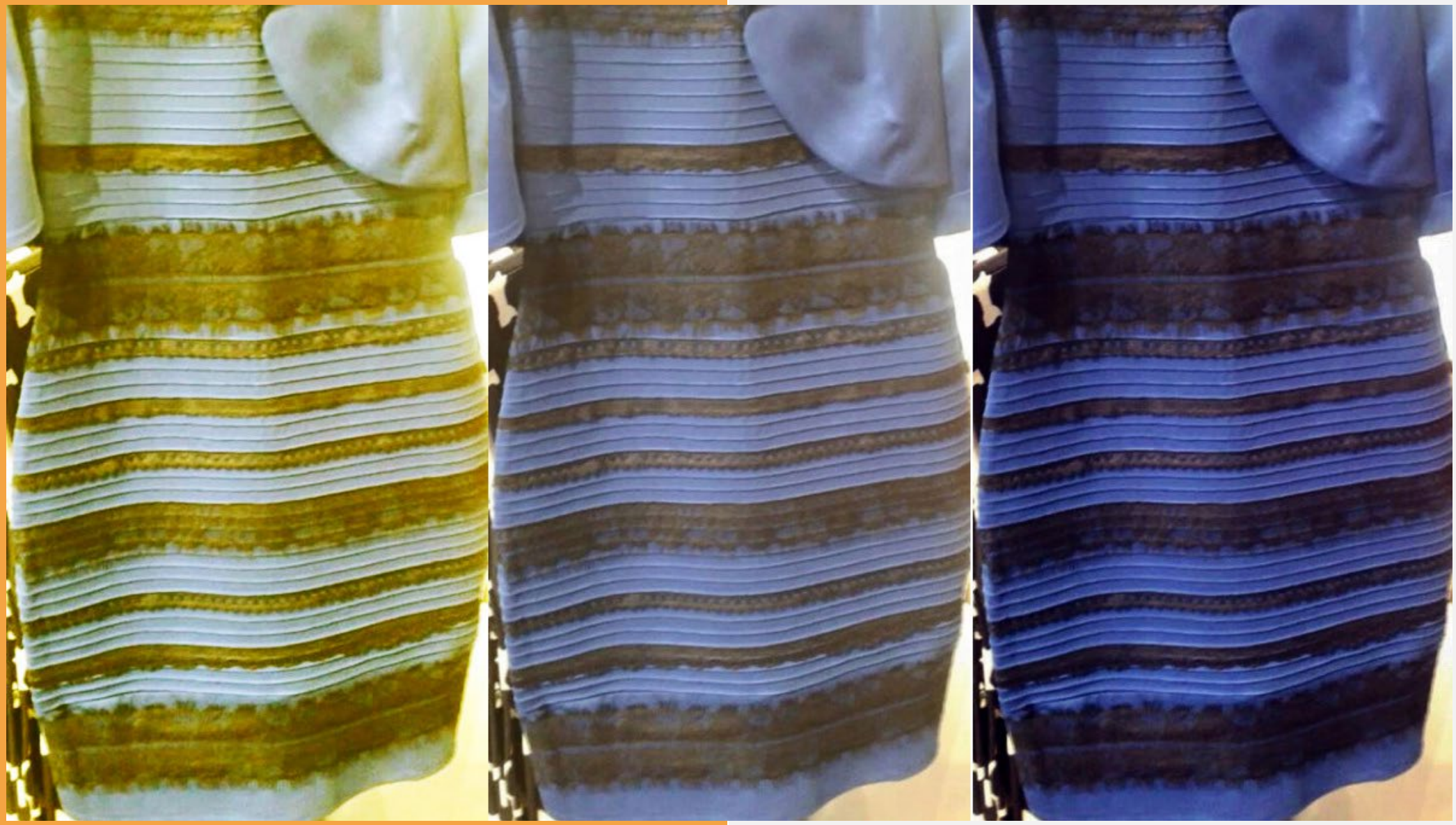
Visualization Accessibility: Graphical Puzzles





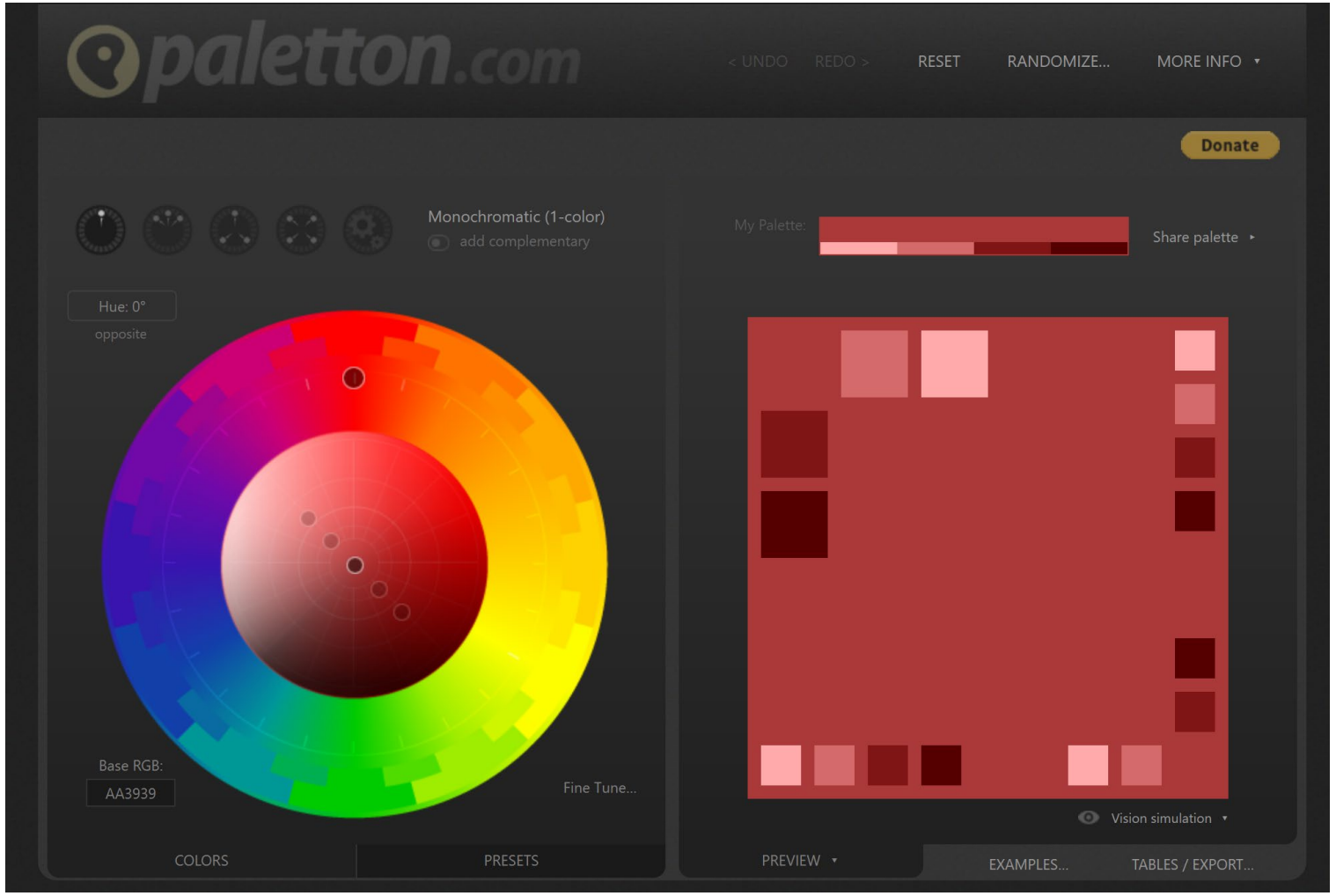


Visualization Accessibility: Colour Blindness

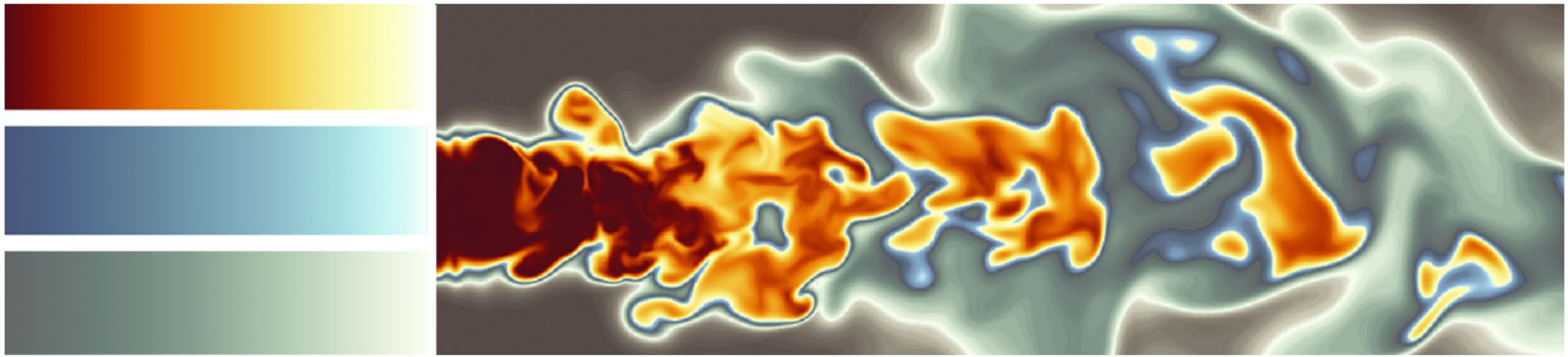


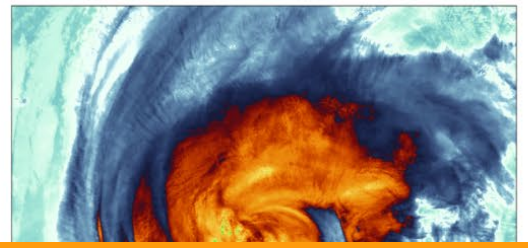
Colours are relative

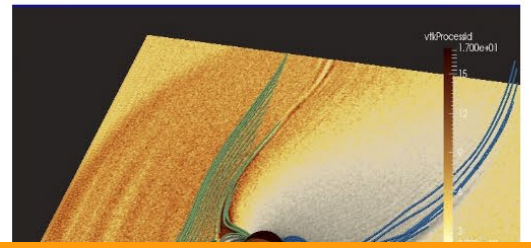
Choosing Harmonious Colours



SciVisColor: Color Tools and Strategies for Scientific Visualization





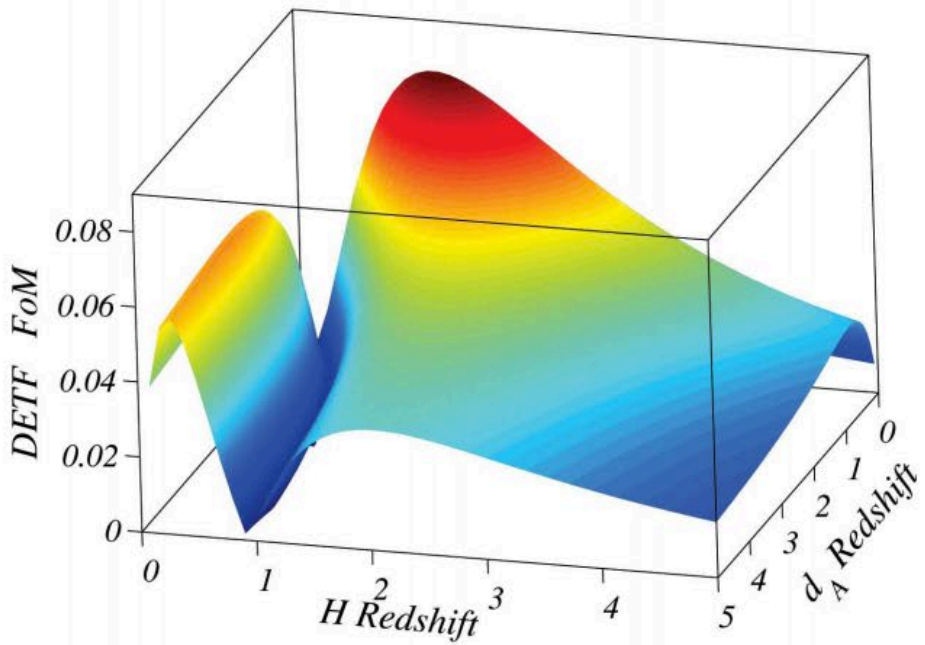


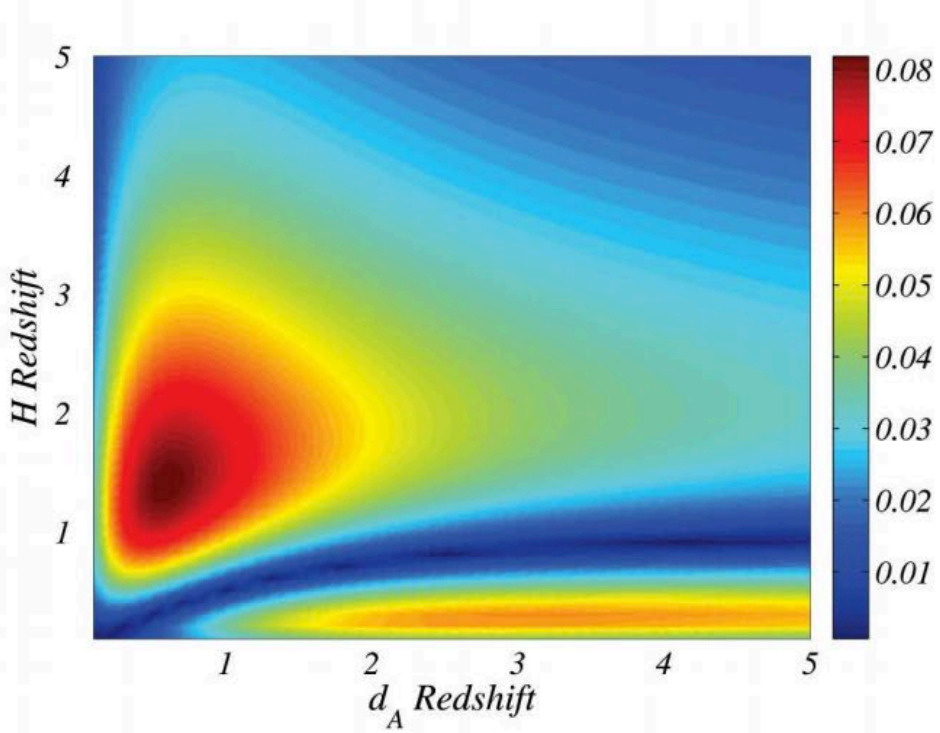
Colour Theory is complex

Which is brightest? Which is darkest?

Colour Theory is complex

Which is brightest? Which is darkest?



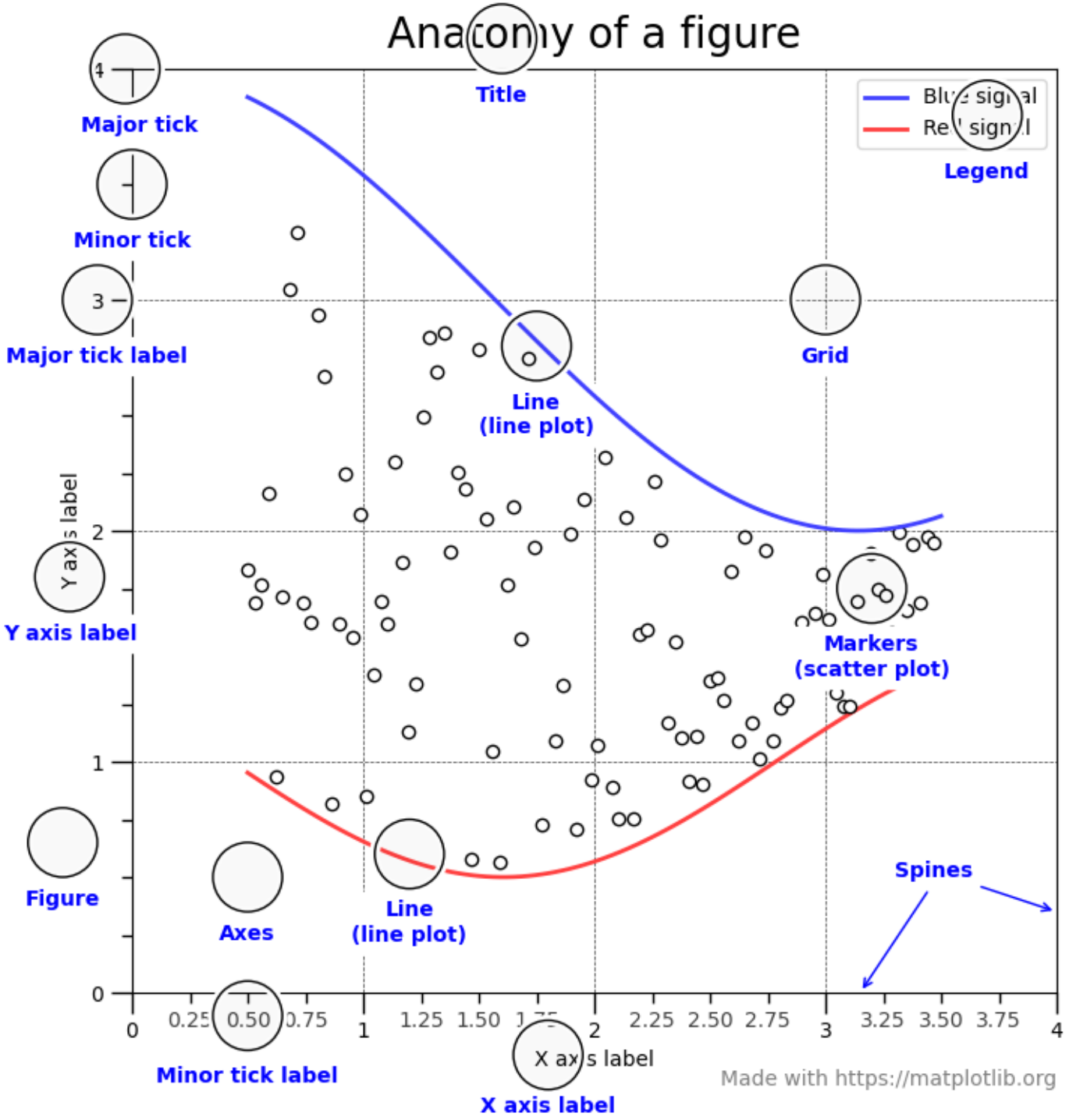


Just because you can, doesn't mean you should

- https://arxiv.org/pdf/0906.0993.pdf

Making Matplotlib do your bidding!

Just need to look up what the element is called, and you can edit it!



Matplotlib Styles

Create your own style file to make your own defaults:

1. Make a text file called **`<stylename>.mplstyle`** and put it in **`~/.config/matplotlib`**
2. Use it by calling **`plt.style.use("<stylename>")`**

You can also change your global defaults by creating/editing a file called **`matplotlibrc`**

Details: https://matplotlib.org/3.2.1/tutorials/introductory/customizing.html

Typography Matters

Critical Tool for visualization

At its best, provides a visual hierarchy and communicates

At its worst, adds chaos and confusion to the visualization

She Ji: The Journal of Design, Economics, and Innovation
Volume 2, Issue 1, Spring 2016, Pages 59-87

Using Typography to Expand the Design Space of Data Visualization

Richard Brath, Ebad Banissi
Show more

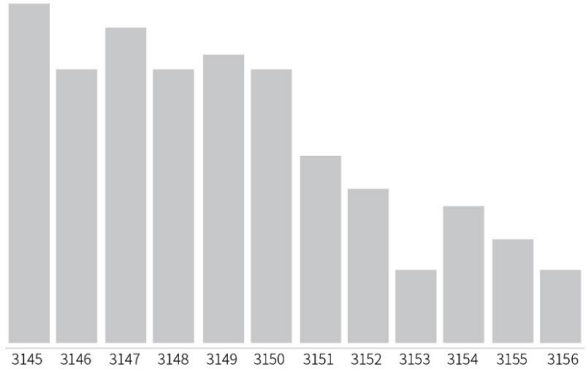
https://doi.org/10.1016/j.sheji.2016.05.003

Highlights

- The design space of data visualization can be expanded with

Lorem Ipsum Dolor

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.



Notes: wikipedia and census.org

Header Text
PT Sans, 22px/24px, bold

Subheader Text
Noto Sans, 14px/20px, light

Callout Text
Noto Sans, 14px/20px, light

Label/Axis Text
Noto Sans, 14px/20px, light

Notes/Sources Text
Noto Sans, 14px/20px, light

National Geographic/ Cartography

They are masters at using and executing visual hierarchies with font faces

Good Open Fonts to use for Visualization

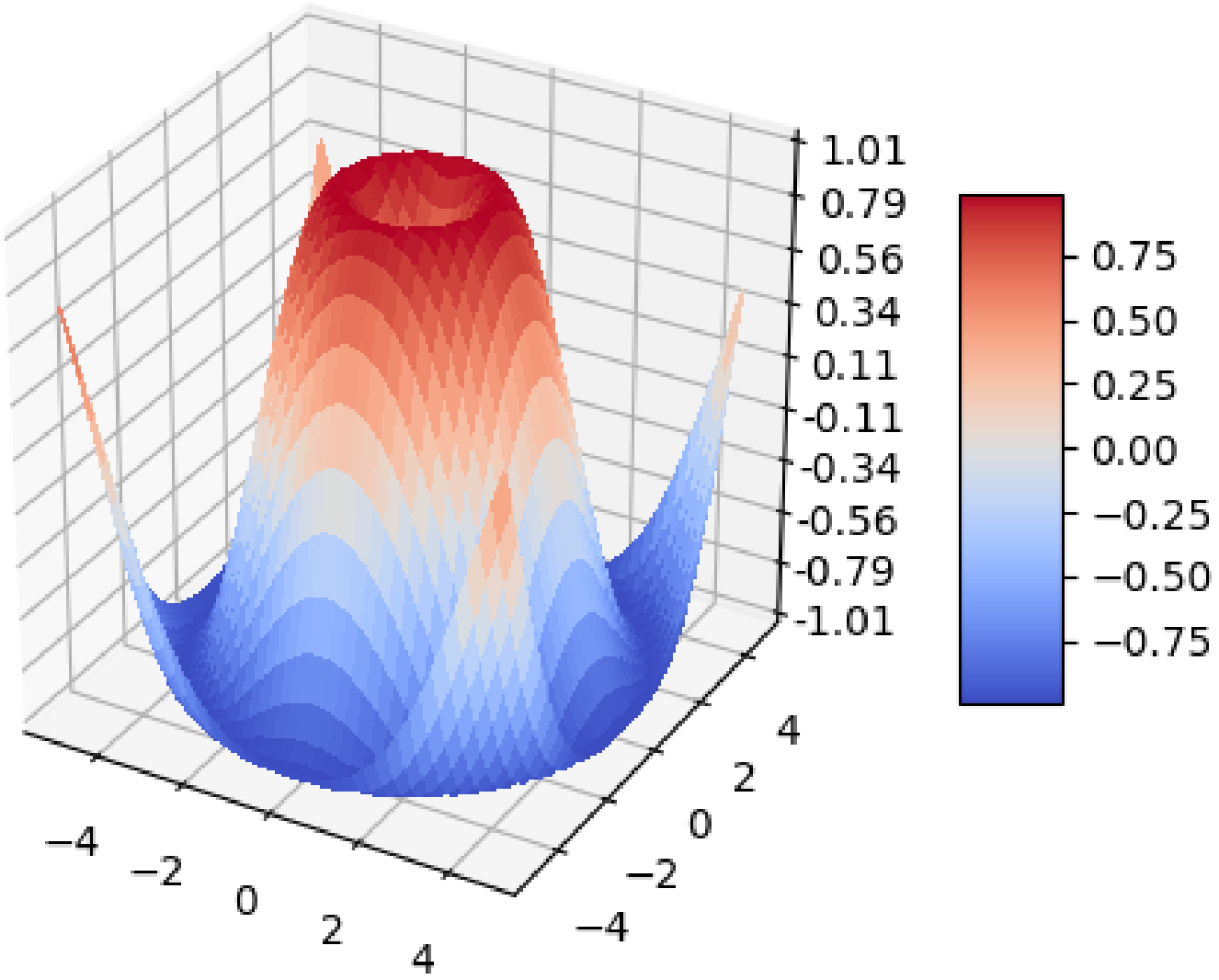
All professionally designed, provide good range of weights and are free to use (open source licensing)

Source Serif Pro **Source Sans 3**

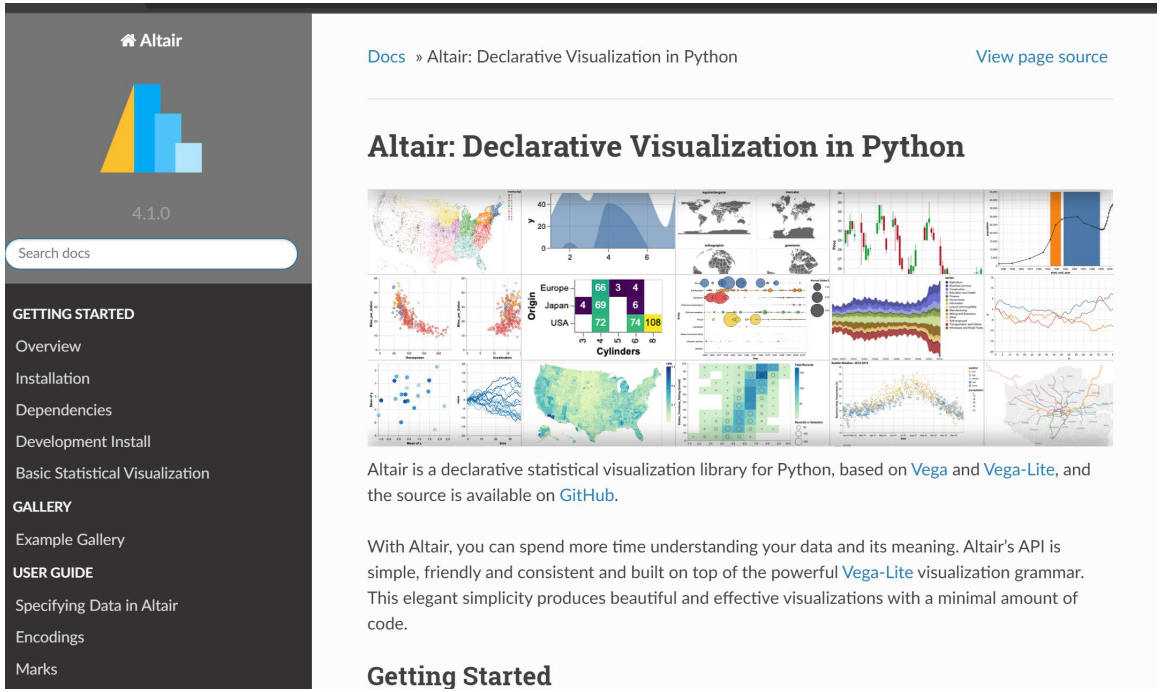
Open Sans

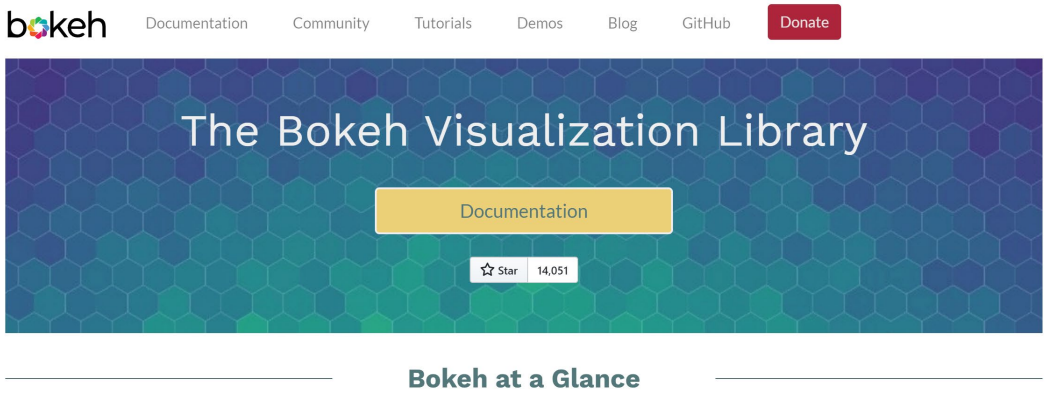
Going Beyond the Basics: 3D

mplot3d gives you built-in 3D capabilities.



Going Beyond the Basics: Interactive Plots





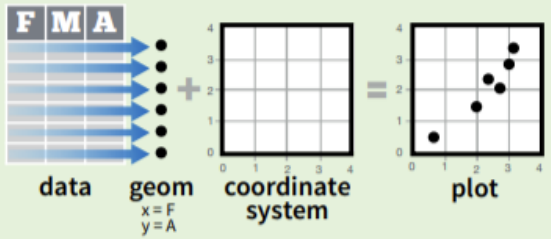
ggplot2

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

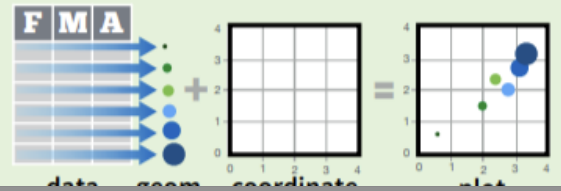
ggplot2 cheat sheet

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.



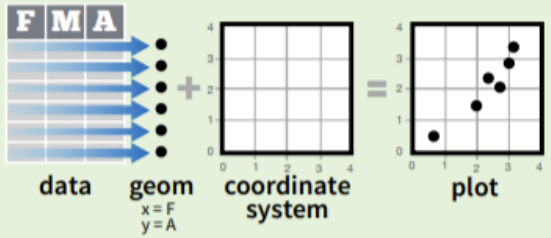
To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



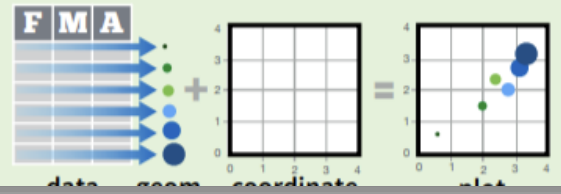
ggplot2 cheat sheet

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



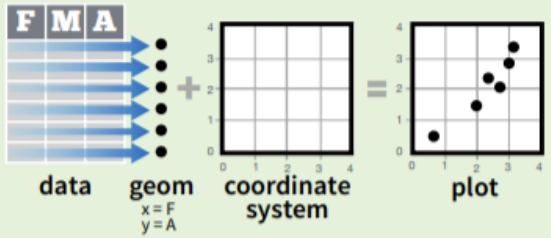
ggplot(data = mpg, **aes(**x = cty, y = hwy**))**

Begins a plot that you finish by adding layers to. Add one geom function per layer.

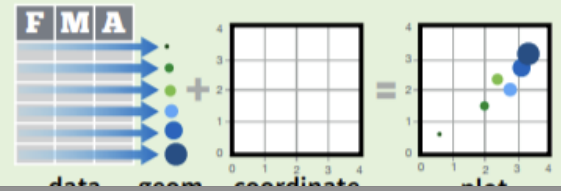
ggplot2 cheat sheet

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



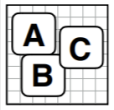
ggplot(data = mpg, **aes(**x = cty, y = hwy**))**

Begins a plot that you finish by adding layers to. Add one geom function per layer.

Two Variables

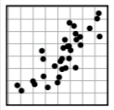
Continuous X, Continuous Y

e <- ggplot(mpg, aes(cty, hwy))



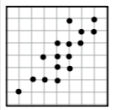
e + **geom_label(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**

x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust



e + **geom_jitter(**height = 2, width = 2**)**

x, y, alpha, color, fill, shape, size

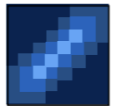


e + **geom_point()**

x, y, alpha, color, fill, shape, size, stroke

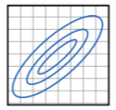
Continuous Bivariate Distribution

h <- ggplot(diamonds, aes(carat, price))



h + **geom_bin2d(**binwidth = c(0.25, 500)**)**

x, y, alpha, color, fill, linetype, size, weight



h + **geom_density2d()**

x, y, alpha, colour, group, linetype, size

h + **geom_hex()**

x, y, alpha, colour, fill, size

Further Resources

- From data to visualization:
  - https://www.data-to-viz.com/
- The R Graph Gallery
  - https://www.r-graph-gallery.com/