Workflow and Data Storage

Projects, Flow Control, and Databases

Housekeeping

- Exams (returned today)
- Practicum 1 due Thursday
  - We will go over in class

Agenda

- Advanced Workflow & Data Storage, aka **Grab Bag Day**
- **Workflow for Portability**
  - Projects, `renv`, etc
- **Flow Control**
  - `if` and `while` loops; apply commands, lists
- **Advanced Storage**
  - Remote data and Databases

Workflow

CONTROL

- and -

Reproducibility

Workflow and folder structure

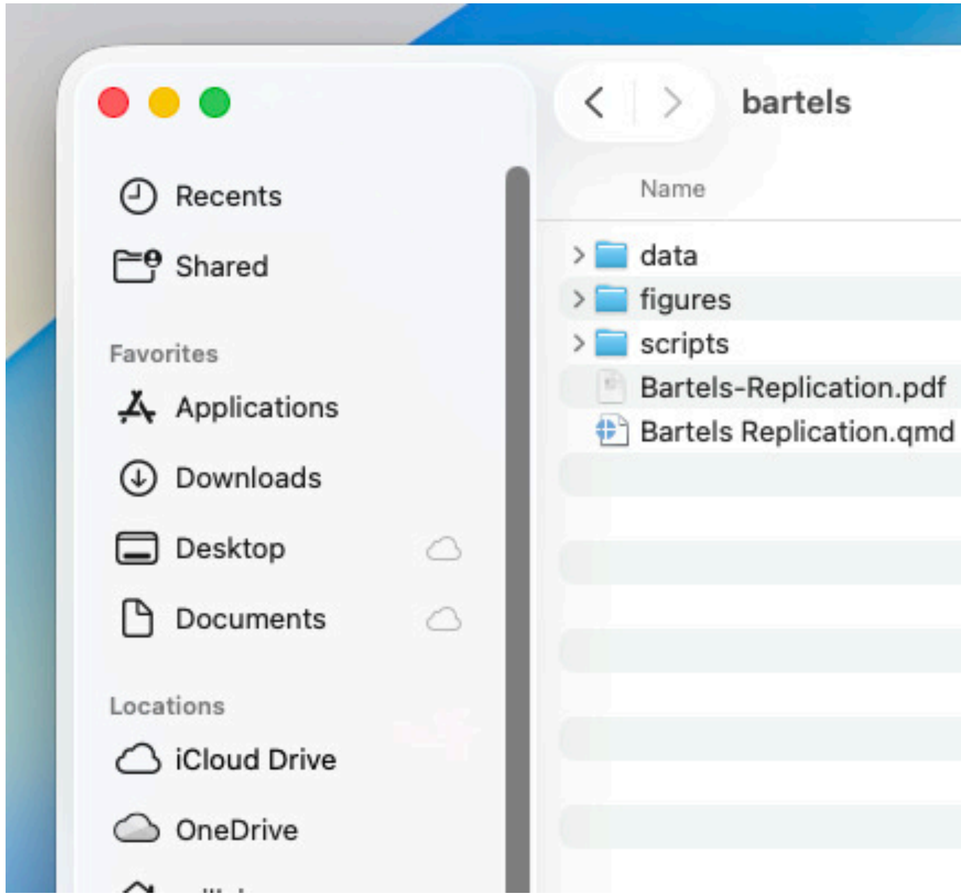
- We often talk about reproducibility as related to **future you**
    - Script files, good folder organization, known hierarchies, etc
    - Quarto and GitHub for reproducible document creation and version control
- We may want different things for truly **portable** reproducibility, however
    - This can also be future you! How likely are you have the exact same folder structure in the future?

Working with others

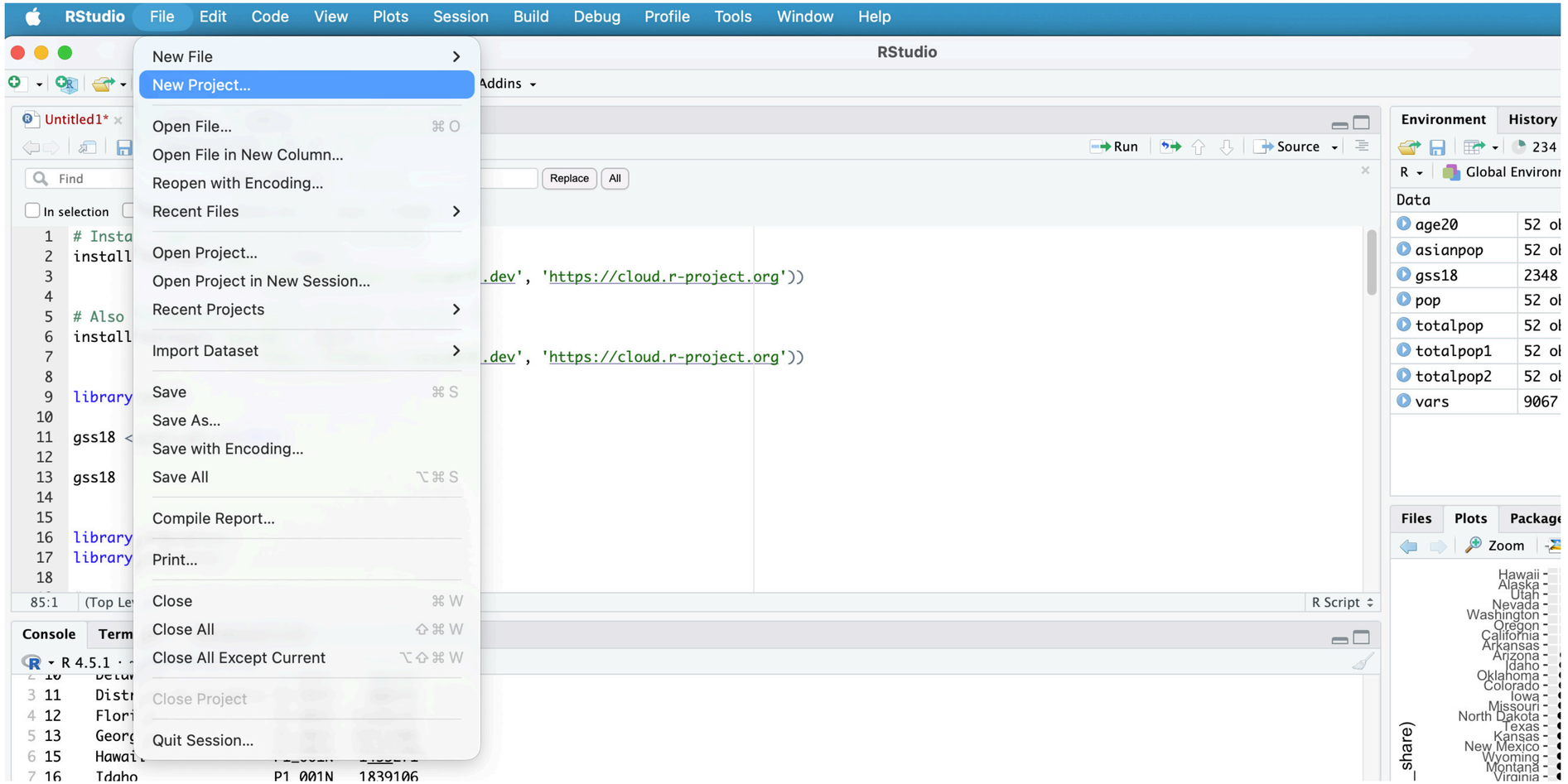
- More commonly, it can be for others (collaborators) using the work on entirely different computers
    - With *god knows what* kind of file organization and R libraries loaded *(shiver)*
- We need some systems to enable easier transport
    - R Projects
    - `renv()`
- Our principle: as much as possible should be **contained in our source files**, leaving as little possible to chance outside that source file

Where are we with data workflow?

- We know to:
  - Use folders for projects
    - With subfolders as useful (“data”, “scripts”, “figures”, etc)
  - Declare libraries at beginnings of scripts
  - Explicitly load data
  - Break code into multiple scripts that call each other
  - Use GitHub for version control

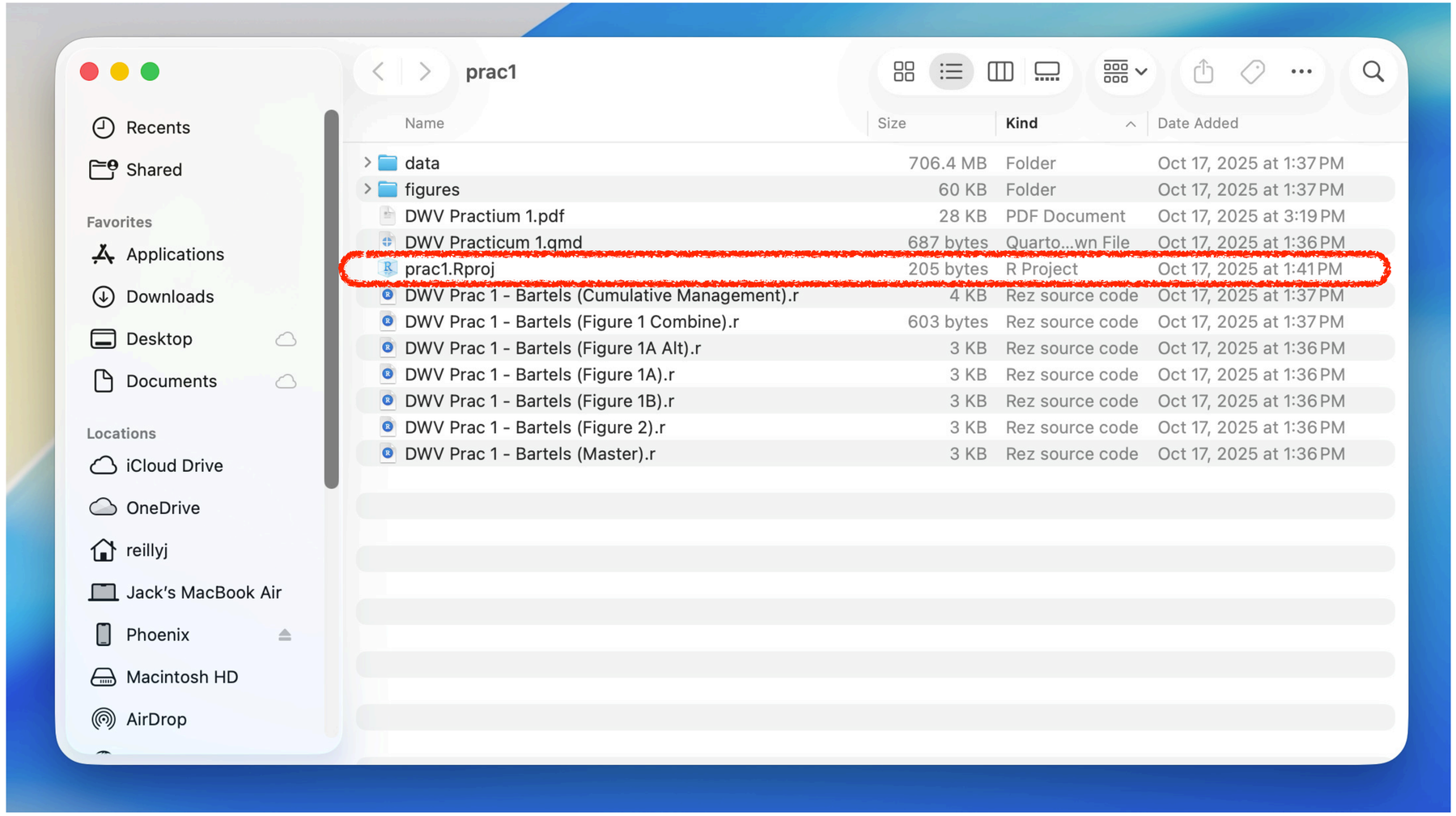


R Projects


RStudio
File
Edit
Code
View
Plots
Session
Build
Debug
Profile
Tools
Window
Help
New File
New Project...
Open File... ⌘O
Open File in New Column...
Reopen with Encoding...
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save ⌘S
Save As...
Save with Encoding...
Save All ⌥⌘S
Compile Report...
Print...
Close ⌘W
Close All ⇧⌘W
Close All Except Current ⌥⇧⌘W
Close Project
Quit Session...
Untitled1*
Addins
Run
Source
Find
Replace
All
In selection
Environment
History
Data
age20 52 ob
asianpop 52 ob
gss18 2348
pop 52 ob
totalpop 52 ob
totalpop1 52 ob
totalpop2 52 ob
vars 9067
Files
Plots
Zoom
Console
R Script
Hawaii
Alaska
Utah
Nevada
Washington
Oregon
California
Arkansas
Arizona
Idaho
Oklahoma
Colorado
Iowa
Missouri
North Dakota
Texas
Kansas
New Mexico
Wyoming
Montana
Virginia

R Projects

- What does a project do for you?
    - Always defines the working directory as the folder the .Rproj file is in
    - Allows *relative file paths*
        - File paths defined in *relation to* the .Rproj working directory, not the absolute file location
    - Reopens all files from last time automatically


prac1
Recents
Shared
Favorites
Applications
Downloads
Desktop
Documents
Locations
iCloud Drive
OneDrive
reillyj
Jack's MacBook Air
Phoenix
Macintosh HD
AirDrop
Name
Size
Kind
Date Added
data 706.4 MB Folder Oct 17, 2025 at 1:37 PM
figures 60 KB Folder Oct 17, 2025 at 1:37 PM
DWV Practium 1.pdf 28 KB PDF Document Oct 17, 2025 at 3:19 PM
DWV Practicum 1.qmd 687 bytes Quarto...wn File Oct 17, 2025 at 1:36 PM
prac1.Rproj 205 bytes R Project Oct 17, 2025 at 1:41 PM
DWV Prac 1 - Bartels (Cumulative Management).r 4 KB Rez source code Oct 17, 2025 at 1:37 PM
DWV Prac 1 - Bartels (Figure 1 Combine).r 603 bytes Rez source code Oct 17, 2025 at 1:37 PM
DWV Prac 1 - Bartels (Figure 1A Alt).r 3 KB Rez source code Oct 17, 2025 at 1:36 PM
DWV Prac 1 - Bartels (Figure 1A).r 3 KB Rez source code Oct 17, 2025 at 1:36 PM
DWV Prac 1 - Bartels (Figure 1B).r 3 KB Rez source code Oct 17, 2025 at 1:36 PM
DWV Prac 1 - Bartels (Figure 2).r 3 KB Rez source code Oct 17, 2025 at 1:36 PM
DWV Prac 1 - Bartels (Master).r 3 KB Rez source code Oct 17, 2025 at 1:36 PM

R Projects

- Now, when I load data:
  - `data<-read_dta(“data/mydata.dta”)`
    - Looks in the “data” subfolder of the project working directory!
- When I write graphics:
  - `ggsave("figures/bartels1_presidential.pdf", bartels1, width = 7, height = 10, dpi = 300)`
    - Saves directly to “figures” subfolder of the projects working directory!

Better, cross platform, file paths

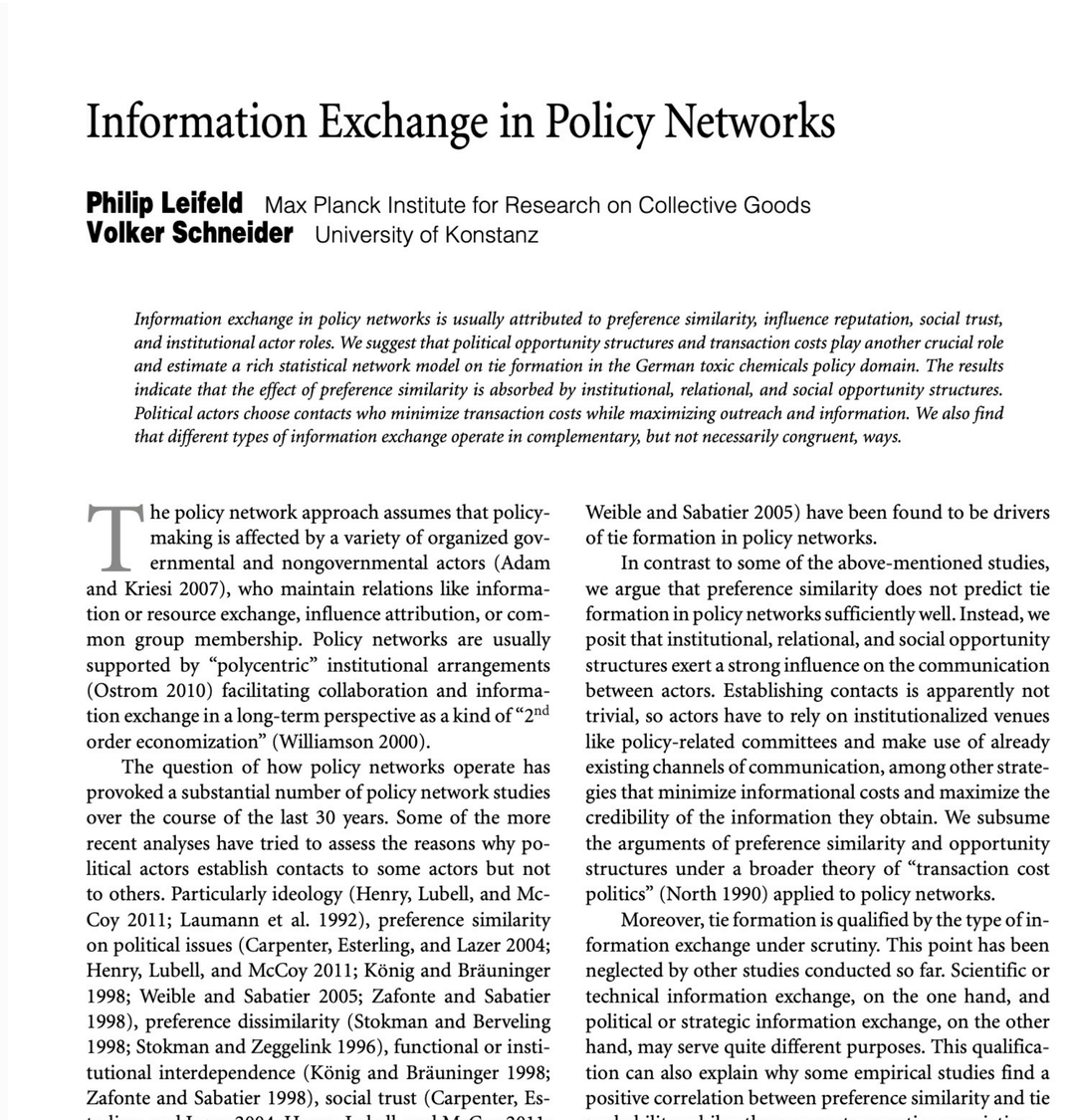
- The `here()` package
  - When used in conjunction with R Projects, lets you identify paths directly in relation to the identified R Project

```
# Load a data file located in the "data"
folder
data <- read_csv(here("data", "survey.csv"))

# Save a plot into the "outputs/plots"
folder
ggsave(here("outputs", "plots",
"my_plot.png"))
```

```
my_project/
├── data/
│   └── survey.csv
├── scripts/
│   └── analysis.R
├── outputs/
│   └── plots/
└── my_project.Rproj
```

That's for files and output

- What else can we self-contain?
  - Libraries
- R packages change over time, and may generate different results
  - Some published work cannot be replicated with newer packages!



Information Exchange in Policy Networks

Philip Leifeld Max Planck Institute for Research on Collective Goods
Volker Schneider University of Konstanz

Information exchange in policy networks is usually attributed to preference similarity, influence reputation, social trust, and institutional actor roles. We suggest that political opportunity structures and transaction costs play another crucial role and estimate a rich statistical network model on tie formation in the German toxic chemicals policy domain. The results indicate that the effect of preference similarity is absorbed by institutional, relational, and social opportunity structures. Political actors choose contacts who minimize transaction costs while maximizing outreach and information. We also find that different types of information exchange operate in complementary, but not necessarily congruent, ways.

The policy network approach assumes that policy-making is affected by a variety of organized governmental and nongovernmental actors (Adam and Kriesi 2007), who maintain relations like information or resource exchange, influence attribution, or common group membership. Policy networks are usually supported by "polycentric" institutional arrangements (Ostrom 2010) facilitating collaboration and information exchange in a long-term perspective as a kind of "2nd order economization" (Williamson 2000).

The question of how policy networks operate has provoked a substantial number of policy network studies over the course of the last 30 years. Some of the more recent analyses have tried to assess the reasons why political actors establish contacts to some actors but not to others. Particularly ideology (Henry, Lubell, and McCoy 2011; Laumann et al. 1992), preference similarity on political issues (Carpenter, Esterling, and Lazer 2004; Henry, Lubell, and McCoy 2011; König and Bräuninger 1998; Weible and Sabatier 2005; Zafonte and Sabatier 1998), preference dissimilarity (Stokman and Berveling 1998; Stokman and Zeggelink 1996), functional or institutional interdependence (König and Bräuninger 1998; Zafonte and Sabatier 1998), social trust (Carpenter, Es-

Weible and Sabatier 2005) have been found to be drivers of tie formation in policy networks.

In contrast to some of the above-mentioned studies, we argue that preference similarity does not predict tie formation in policy networks sufficiently well. Instead, we posit that institutional, relational, and social opportunity structures exert a strong influence on the communication between actors. Establishing contacts is apparently not trivial, so actors have to rely on institutionalized venues like policy-related committees and make use of already existing channels of communication, among other strategies that minimize informational costs and maximize the credibility of the information they obtain. We subsume the arguments of preference similarity and opportunity structures under a broader theory of "transaction cost politics" (North 1990) applied to policy networks.

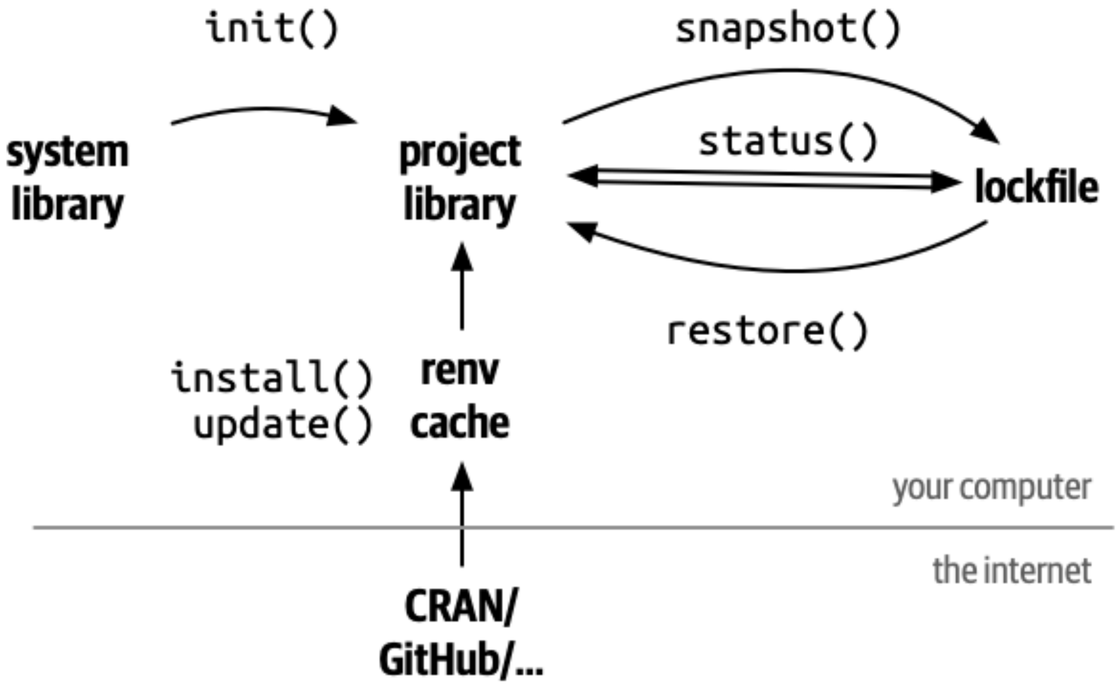
Moreover, tie formation is qualified by the type of information exchange under scrutiny. This point has been neglected by other studies conducted so far. Scientific or technical information exchange, on the one hand, and political or strategic information exchange, on the other hand, may serve quite different purposes. This qualification can also explain why some empirical studies find a positive correlation between preference similarity and tie

The Solution: `renv`

- Think “R Environment” - kind of like a separate virtual environment just for your R project
    - Controls exact versions of libraries to keep consistent across computers
    - Keeps separate archive of libraries from shared repository on computer
- Core insight: a “lockfile” that is stored in your project directory and tracks your package versions

The Solution: `renv`

- We start a fresh project, and:
  - `renv::init()`
  - `renv::snapshot()`
  - `renv::restore()`
- Compatible with git version control, can be run across computers and operating systems



Is it worth it?

- Honestly? I'm not quite sure
  - There are always things that are going to be outside the control of the `renv()`
- For instance:
  - External to R system libraries:
    - Pandoc (heavily relied upon by Quarto & Rmarkdown)
    - Latex (similar)
  - Esoteric R packages might not always load properly

Advice for portability

- Keep your R files **minimal**
  - The more specialization you use - the more libraries - the more likely it is something will break on another's machine (or your future machine)
- Also known as the
  - **K**eep **I**t **S**imple **S**tupid principle

Flow Control

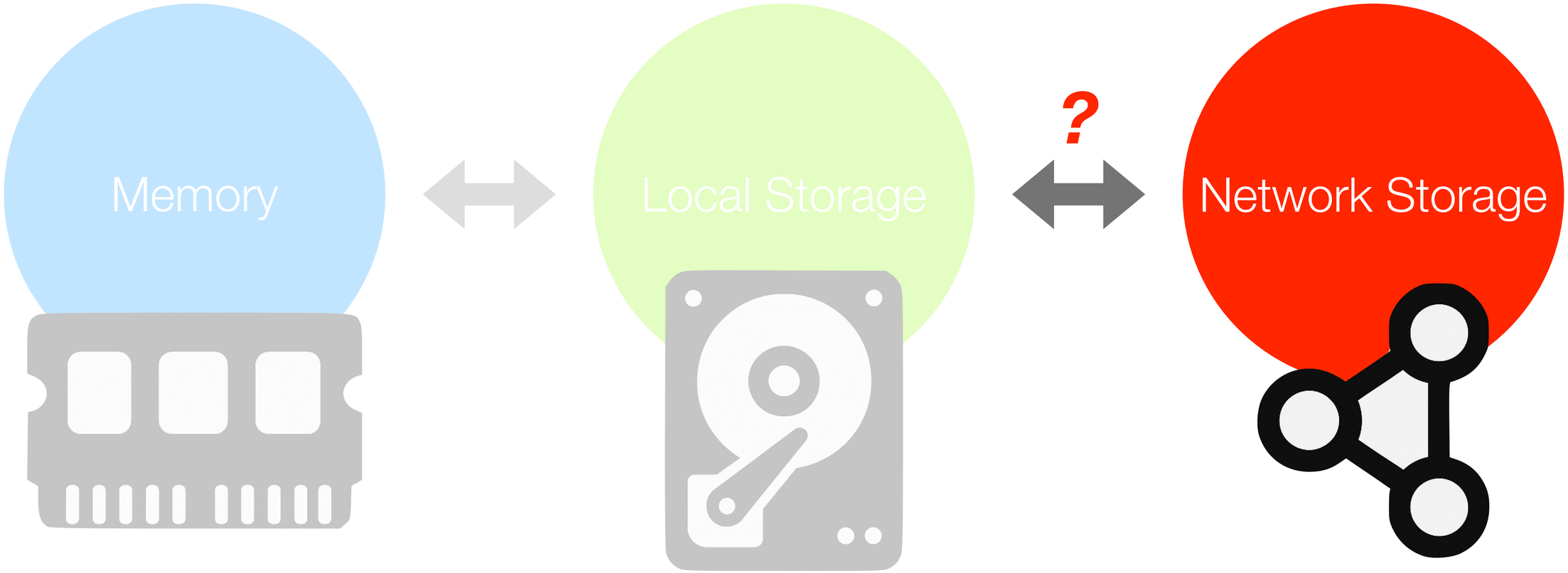
Stay tuned

External Data

External Data

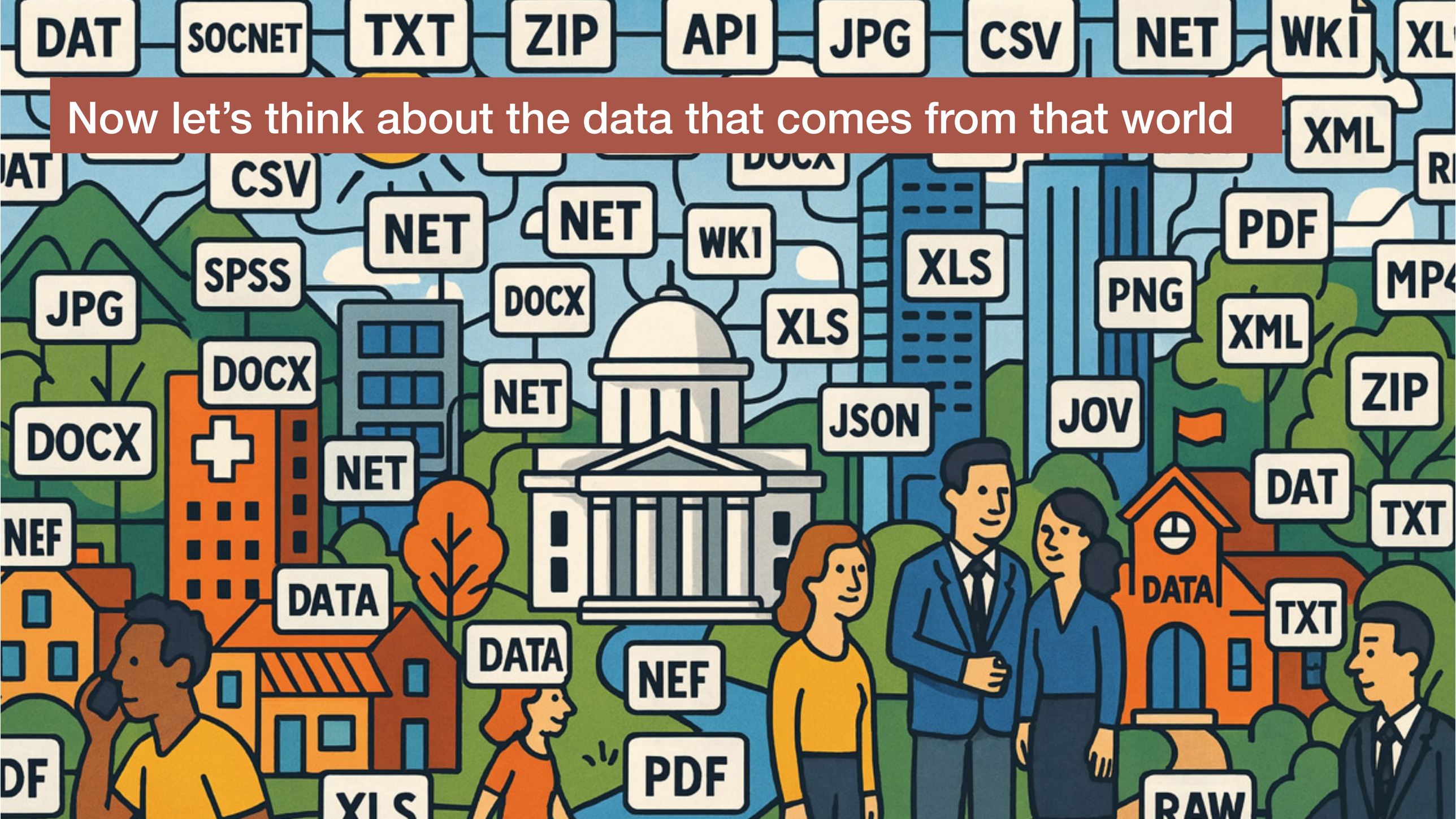
- Up until now, we've primarily covered one kind of data: the kind that is stored **locally** and you read directly in to local memory from your local disk

What about data elsewhere?

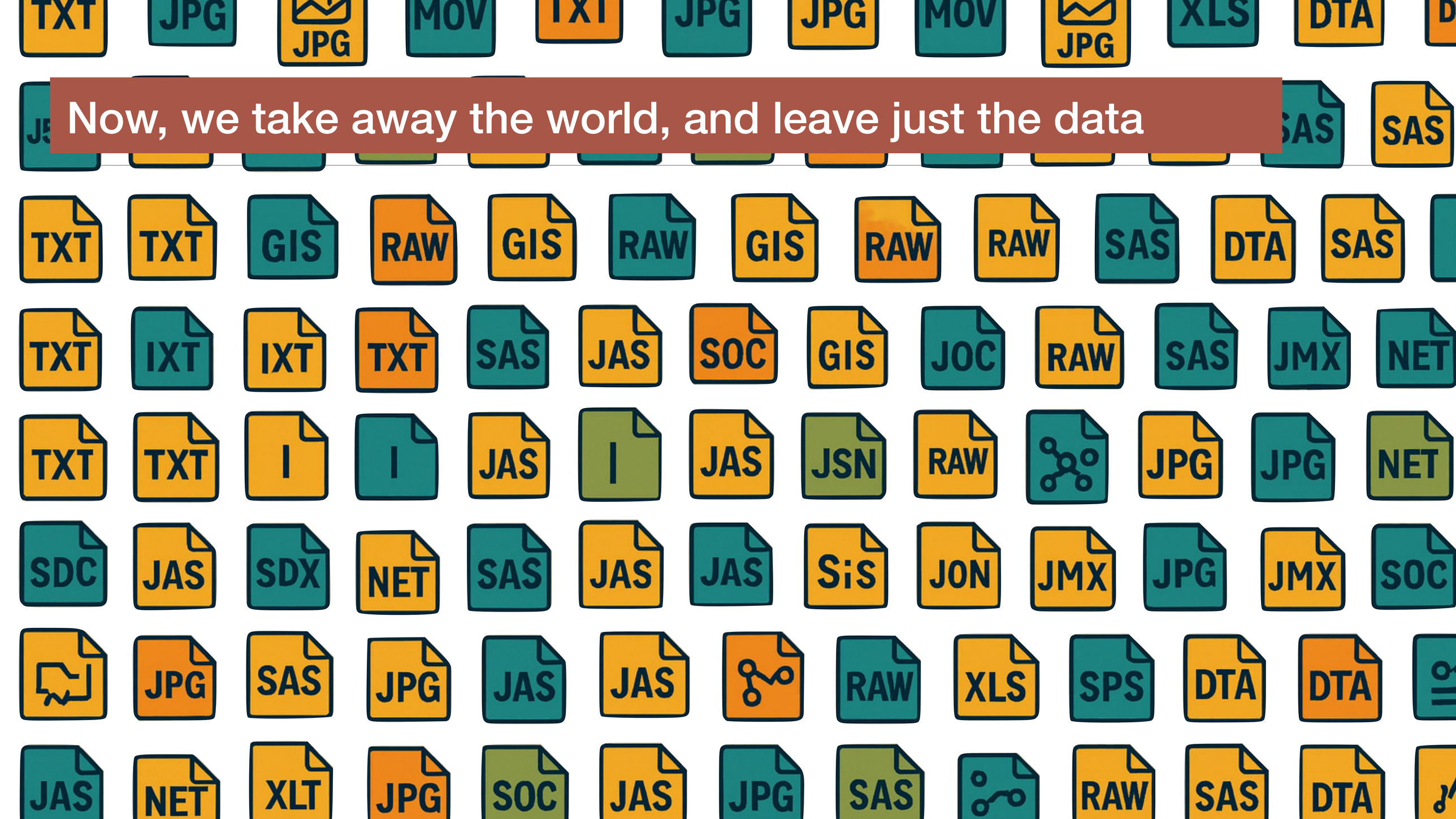


Let’s clarify some commonly-used terminology


Let’s think, big picture about the whole data process starts
SCHOOL


Now let’s think about the data that comes from that world
DAT
SOCNET
TXT
ZIP
API
JPG
CSV
NET
WKI
XML
PDF
PNG
XLS
SPSS
DOCX
JSON
JOV
ZIP
DAT
TXT
NEF
DATA
RAW


Now let’s think about the data that comes from that world
Some is structured, some is not!


Now, we take away the world, and leave just the data


Now, we take away the world, and leave just the residual data
XI
GIS
XLS
DTA
SDS
DTA
SPSS
JPG
XX
TXT
JPG
DTA
JSON
MP4
JSN
MXT
XXL
NET
JPG
TXT
If we collect it all together, this is a Data Lake


Some of this data is structured, so we organize things a little bit
XLS
XLS
XLS
XLS
JSON
JSON
JSON
JSON
TXT
TXT
TXT
GIS
GIS
RAW
RAW
JPG
MOV


And maybe we put them in known boxes and shelves
DTA
TXT
SONE
XLS
TXT
CSV
CSV
NET
NEB
SOCNET
CSV
TXT
this is a Lakehouse


Maybe we organize all of it!
DTA
XLS
CSV
JSON
XML
CSV
XML
SQL
SQL
SQL
JSO
SOL
JSOΛ
XML
CSV
this is a Warehouse

Now, we need to access this

- Lakehouses, Warehouses, etc:
  - Are not single data files
  - Cannot fit on your computer
- Treasure troves of genuinely monstrous amounts of information

For example

- Walmart: data warehouse of 2.5 petabytes (customer transactions)
- Banks: multi-petabyte files (transactions)
- Tech companies: exabytes of customer behavior data
- Social Security database: 2.9 billion numbers and associated information
- New York City Open Data portal: 2,700 datasets, several terabytes overall (at minimum)

How big is big?

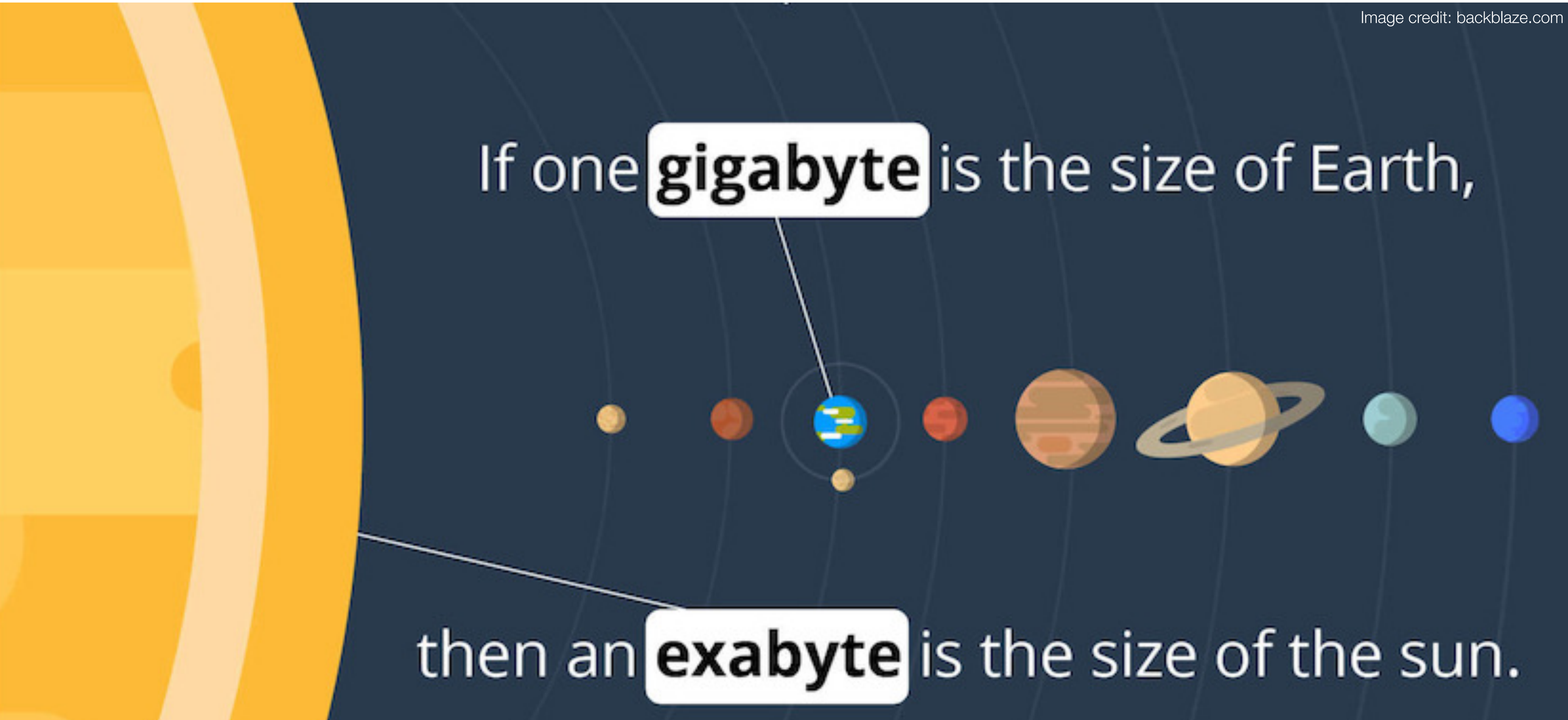


Image credit: backblaze.com

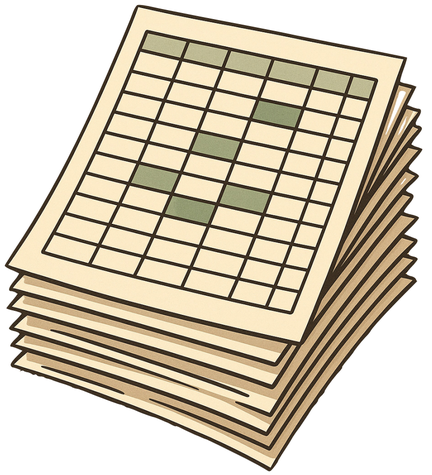
We need new metaphors to manage all of this

- Upjumped rectangular Excel files won't do it anymore
  - There are a variety of tools
  - The most common are Databases (especially **SQL** databases)
- We access databases in one of two ways
  - **APIs** *("Application Programming Interface")*
    - *R packages often exist to simplify matters here*
  - **Databases**

What is a relational database?

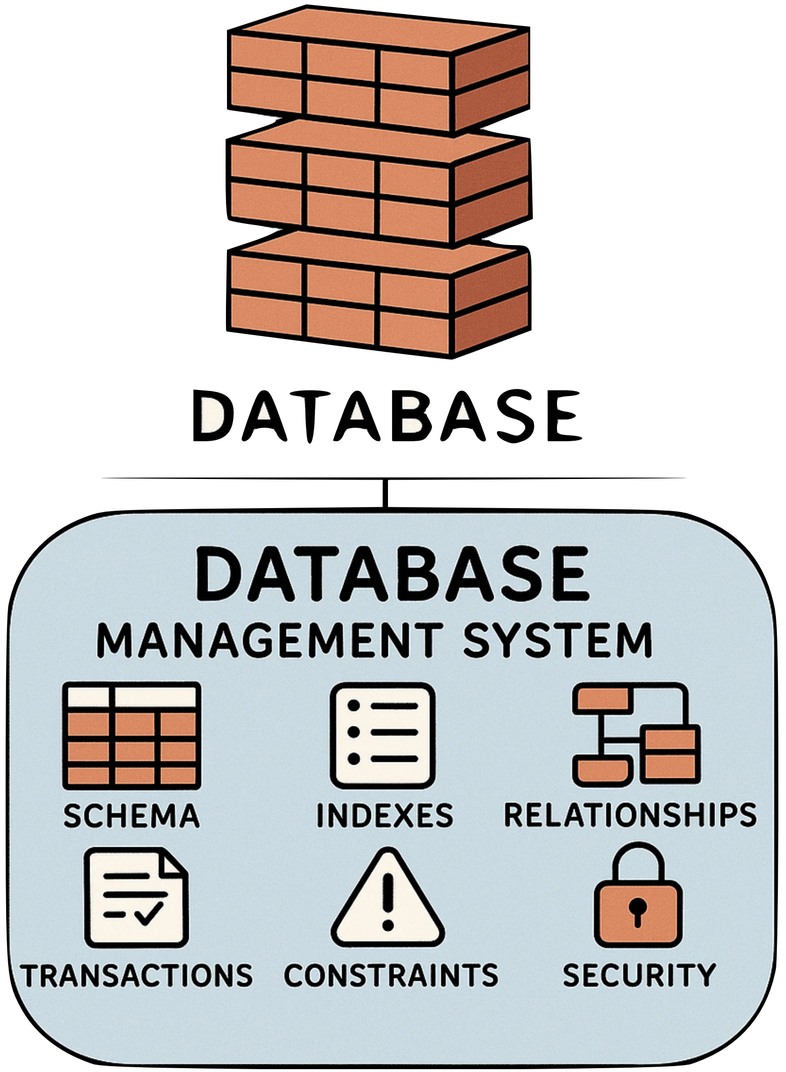
- Somewhere in between the data warehouse and the stack of datasets

DBMS



Databases store a bunch of information for us

- Records themselves, often from different sources or representing different things
- Relations between those records
- They also enable additional nice properties:
  - Indices
  - Security
  - Enforcement schema
  - Controlled multiple access



Database Management System (DBMS)

- An organized storage system for large amounts of data
  - Enforces rules and allows access to records
- You do not access the entire data all at once
  - Rather, you form a *query* that the DBMS then *parses and executes,* ultimately returning you what you asked for
- Kind of like a librarian in a controlled access library
  - You make a request at the desk, the librarian then returns the records you have asked for

Basic database access is not terribly complicated*

- Well, not terribly complicated relative to the complexity of the task and the technological sophistication of the average person trying to access them

- SQL: the most common relation database “language”, and access through R looks familiar:

```
lsq_data = dbGetQuery(db, "SELECT
           election_id, country_name,
           sqrt(0.5 * sum(power(vote_share - seat_share, 2)))
              AS lsq_index
           FROM elections
           WHERE extract(year from election_date) >= 1946
           GROUP BY election_id, country_name")
```

*said in the not-quite-forthright way technology people say these things

What does this mean for you?

- Local files - .csv, .dta., .R, etc
  - are *usually* fine for one researcher
  - And for the kind of files you are likely to use
- But the larger the data you use, the more records and interlinked types are involved, and the more people working with it, the more you're likely to use (or need to access data) via DBMS.

For instance:

- **Professor Reilly** keeps class records in a single spreadsheet
  - What do I have?
    - Grades, attendance, not much else
- **Syracuse University** keeps its student records in databases
  - What does SU have?
    - Financial aid, housing, class registration, grades, athletics, student support systems . . .

Phase I wrap-up

- Base data analysis jobs also often expect some basic familiarity with SQL queries, if you end up looking for those

- What you really need is to know how to access data external to you

Two examples of external data usage

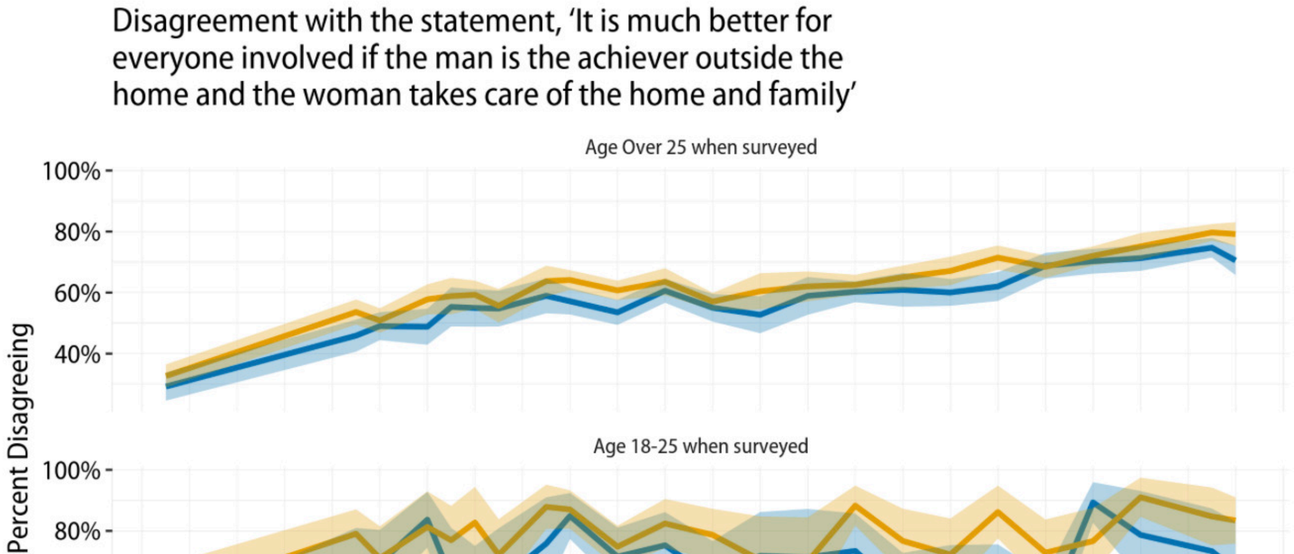
- The General Social Survey (again!)
- The United States Census

The General Social Survey with `gssr` (and `gssrdoc`)

gssr 0.7 Reference Articles Changelog

gssr

The General Social Survey Cumulative Data (1972-2024, release 1) and Panel Data files packaged for easy use in R. The companion package to gssr is gssrdoc, which integrates the GSS codebook into R's help system. I recommend you install both packages.



Why would I use this?

- Ease
- Even if it does fit on your computer, the GSS is large and unwieldy
  - *As you can attest to:* thousands of variables, tens of thousands of observations
- `gssr` and `gssrdoc` are simpler ways to interact with all the information that is the GSS
- Files are more reproducible! No more mucking around in the filesystem to identify your data

Usage


```
# Install 'gssr' from 'ropensci' universe
install.packages('gssr', repos =
  c('https://kjhealy.r-universe.dev',
    'https://cloud.r-project.org'))

# Also recommended: install 'gssrdoc' as well
install.packages('gssrdoc', repos =
  c('https://kjhealy.r-universe.dev',
    ‘https://cloud.r-project.org'))
```

Very simple syntax!


```
> library(gssr)
> gss18 <- gss_get_yr(2018)
> gss18
```

```
# A tibble: 2,348 × 1,136
   year          id wrkstat    hrs1         hrs2         evwork       wrkslf  wrkgovt occ10       prestg10 prestg105plus indus10
   <dbl+lbl> <dbl> <dbl+lbl>  <dbl+lbl>    <dbl+lbl>    <dbl+lbl>    <dbl+l> <dbl+l> <dbl+lbl>   <dbl+lb> <dbl+lbl>      <dbl+lbl>
 1 2018          1 3 [with a… NA(i) [iap]     41        NA(i) [iap] 2 [som… 2 [pri…  630 [hum… 47       59             7580 [emp…
 2 2018          2 5 [retire… NA(i) [iap] NA(i) [iap]      1 [yes] 2 [som… 2 [pri… 9640 [pac… 22       13             4970 [gro…
 3 2018          3 1 [workin…     40      NA(i) [iap] NA(i) [iap] 2 [som… 2 [pri… 1106 [com… 61       92             6680 [wir…
 4 2018          4 1 [workin…     40      NA(i) [iap] NA(i) [iap] 2 [som… 2 [pri… 3320 [dia… 59       79             8190 [hos…
 5 2018          5 5 [retire… NA(i) [iap] NA(i) [iap]      1 [yes] 2 [som… 2 [pri…   10 [chi… 53       73             3390 [ele…
 6 2018          6 5 [retire… NA(i) [iap] NA(i) [iap]      1 [yes] 2 [som… 2 [pri…  120 [fin… 53       65             3380 [nav…
 7 2018          7 1 [workin…     35      NA(i) [iap] NA(i) [iap] 2 [som… 1 [gov… 3600 [nur… 48       50             7580 [emp…
 8 2018          8 1 [workin…     89 [89+… NA(i) [iap] NA(i) [iap] 2 [som… 2 [pri… 5610 [shi… 35       27             4970 [gro…
 9 2018          9 1 [workin…     40      NA(i) [iap] NA(i) [iap] 1 [sel… 2 [pri… 4600 [chi… 35       28             8470 [chi…
10 2018         10 1 [workin…     40      NA(i) [iap] NA(i) [iap] 2 [som… 2 [pri… 6700 [ele… 39       34              770 [con…
# ℹ 2,338 more rows
# ℹ 1,124 more variables: marital <dbl+lbl>, . . .
```

The `gssrdoc` package makes finding things useful, too


```
library(gssr)
data(gss_all) # loads full gss

library(gssrdoc)
gss_all |>
  gss_which_years(c(industry,
      indus80, wrkgovt,
      commute)) |>
  print(n = Inf)
```

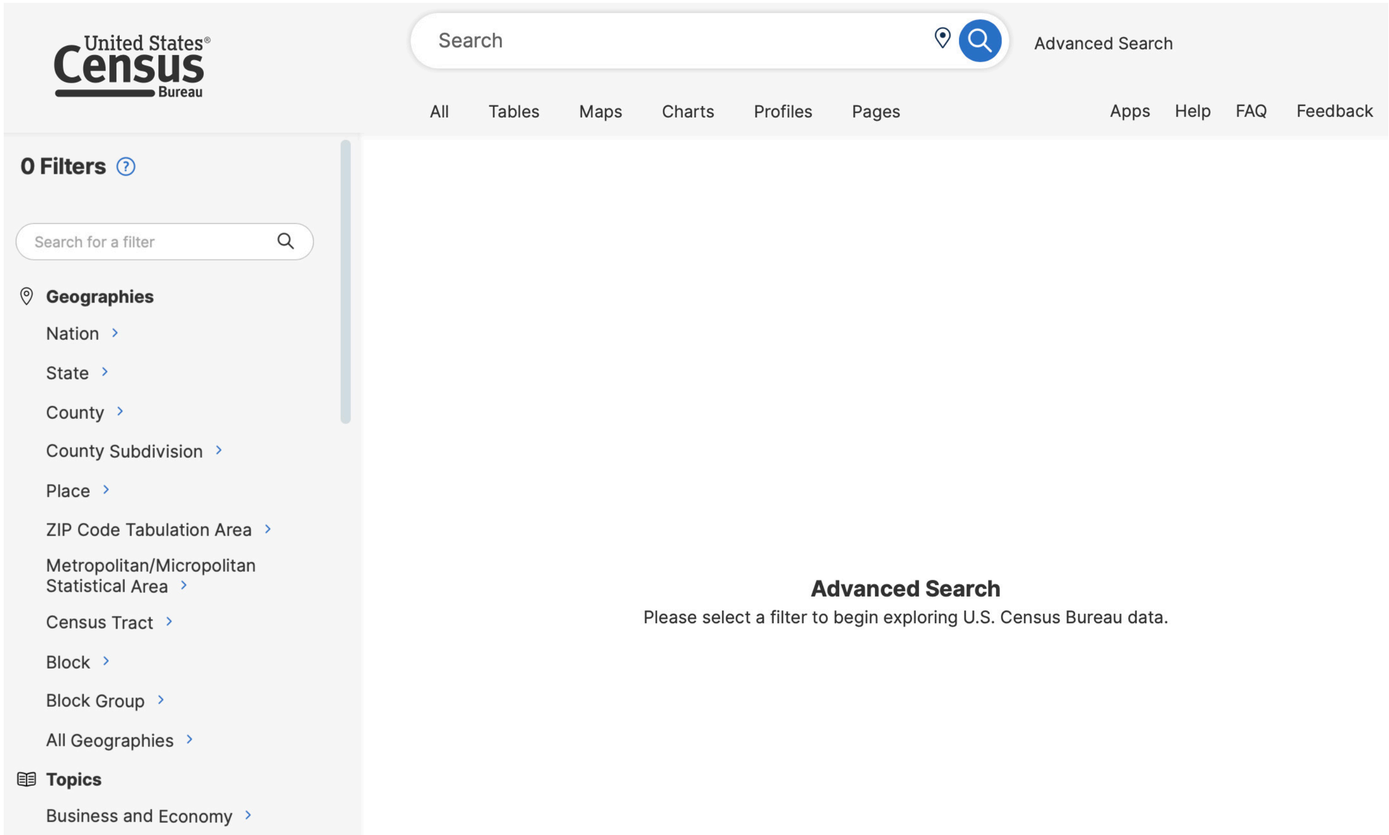
```
# A tibble: 35 × 5
   year      industry indus80 wrkgovt commute
   <dbl+lbl> <lgl>    <lgl>   <lgl>   <lgl>
## 1972      TRUE     FALSE   FALSE   FALSE
## 1973      TRUE     FALSE   FALSE   FALSE
## 1974      TRUE     FALSE   FALSE   FALSE
## 1975      TRUE     FALSE   FALSE   FALSE
## 1976      TRUE     FALSE   FALSE   FALSE
## 1977      TRUE     FALSE   FALSE   FALSE
## 1978      TRUE     FALSE   FALSE   FALSE
## 1980      TRUE     FALSE   FALSE   FALSE
## 1982      TRUE     FALSE   FALSE   FALSE
## 1983      TRUE     FALSE   FALSE   FALSE
## 1984      TRUE     FALSE   FALSE   FALSE
## 1985      TRUE     FALSE   TRUE    FALSE
## 1986      TRUE     FALSE   TRUE    TRUE
## 1987      TRUE     FALSE   FALSE   FALSE
## 1988      TRUE     TRUE    FALSE   FALSE
## 1989      TRUE     TRUE    FALSE   FALSE
## 1990      TRUE     TRUE    FALSE   FALSE
## 1991      FALSE    TRUE    FALSE   FALSE
## 1993      FALSE    TRUE    FALSE   FALSE
```

gssr

- Full documentation on the package website:
  - https://kjhealy.github.io/gssr/
- Very similar, in principle, to what you'll be doing if you work with SQL databases:
  - *Pre*-filtering and *pre*-selecting data from the central repository before it goes into your data object

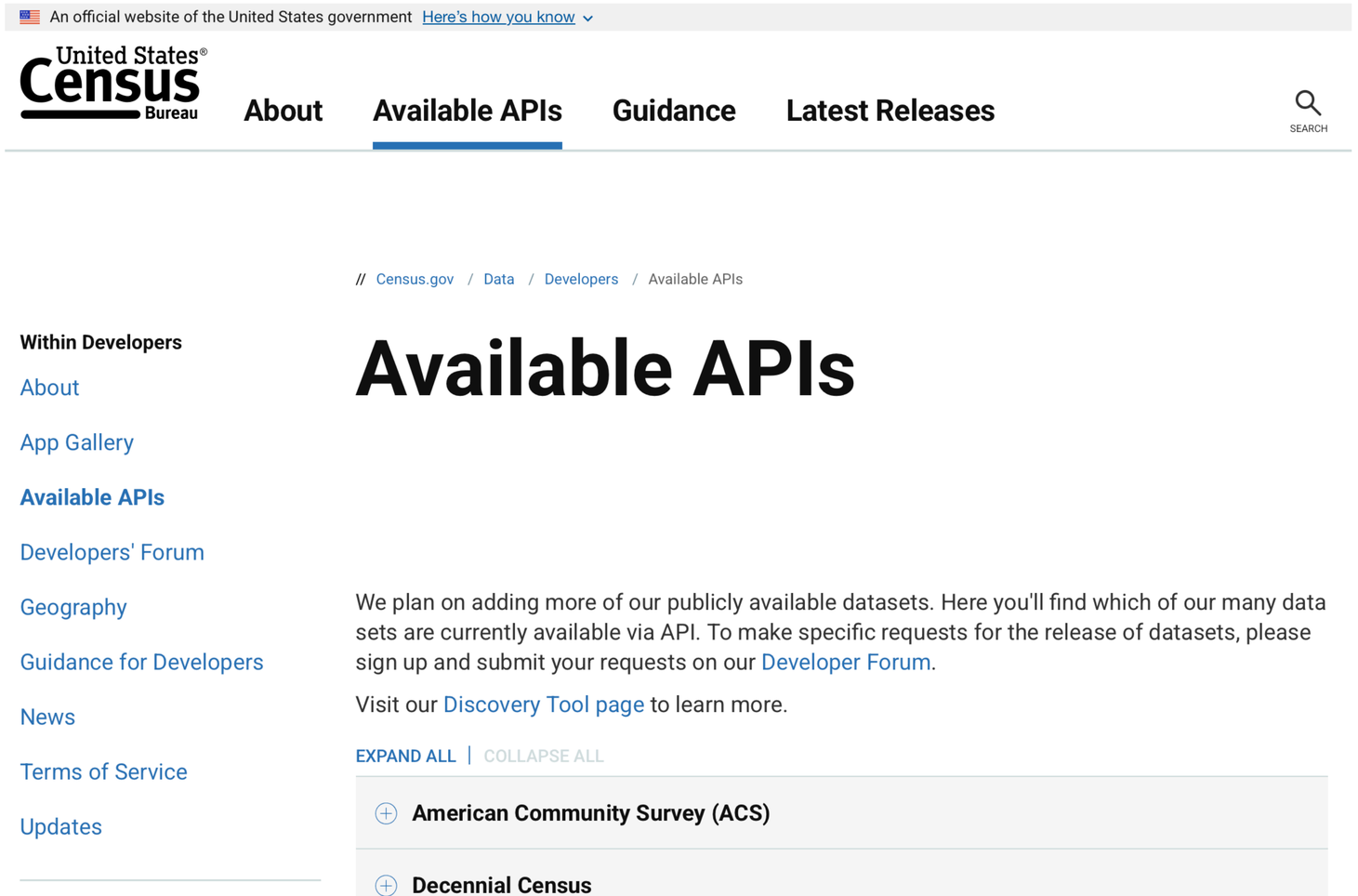
Census data

- You have at least three ways to access census data for manipulation.
  - Website



Census data

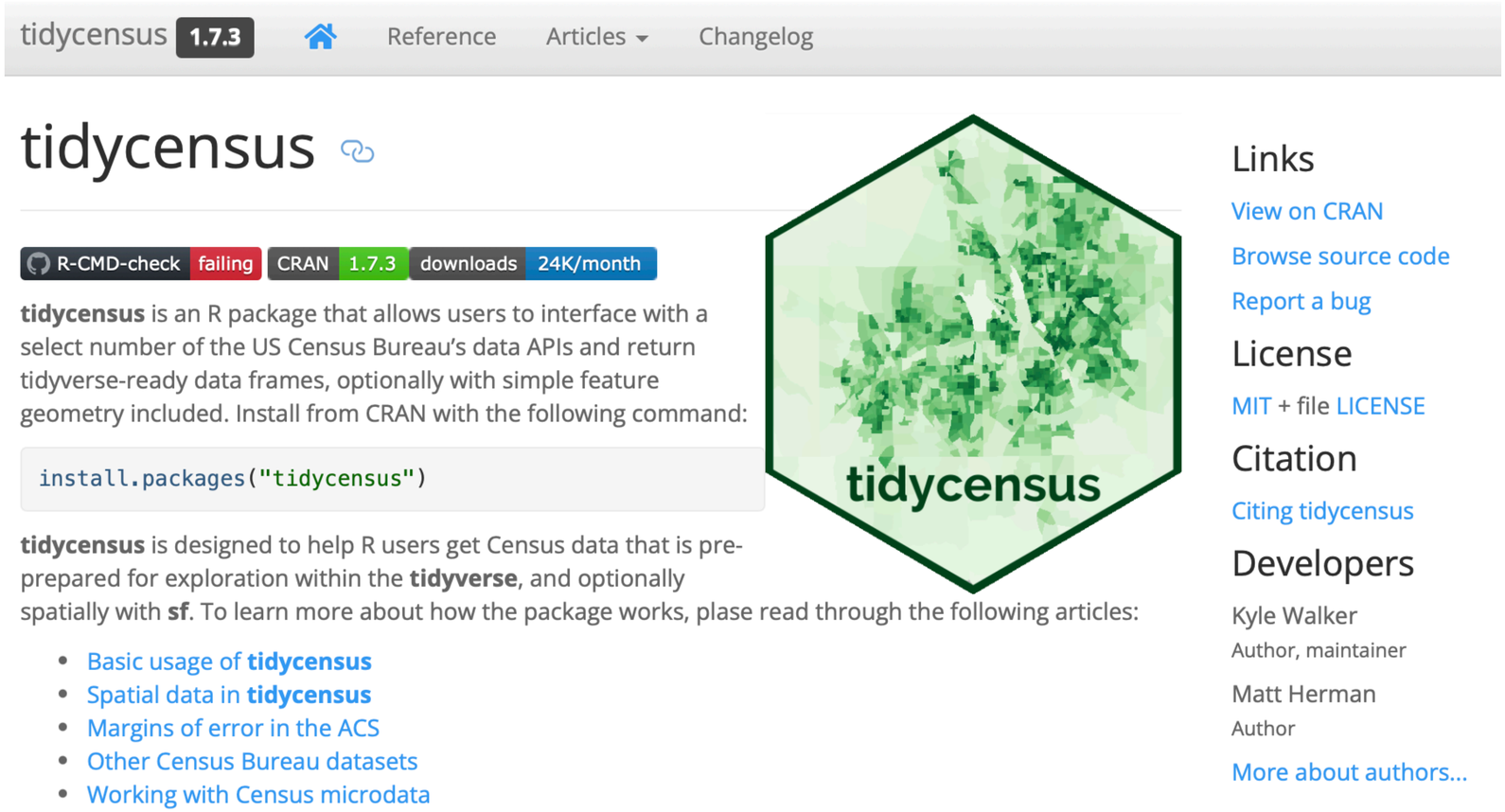
- You have at least three ways to access census data for manipulation.
  - Website
  - Census API



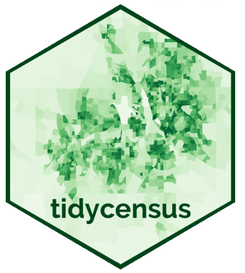
https://www.census.gov/data/developers/data-sets.html

Census data

- You have at least three ways to access census data for manipulation.
  - Website
  - Census API
  - `tidycensus`

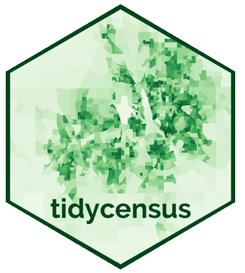


Census data with `tidycensus`



- What does `tidycensus` provide?
  - Similar features to `gssr`
  - Clean calls to core census data archive
  - Storage in R objects

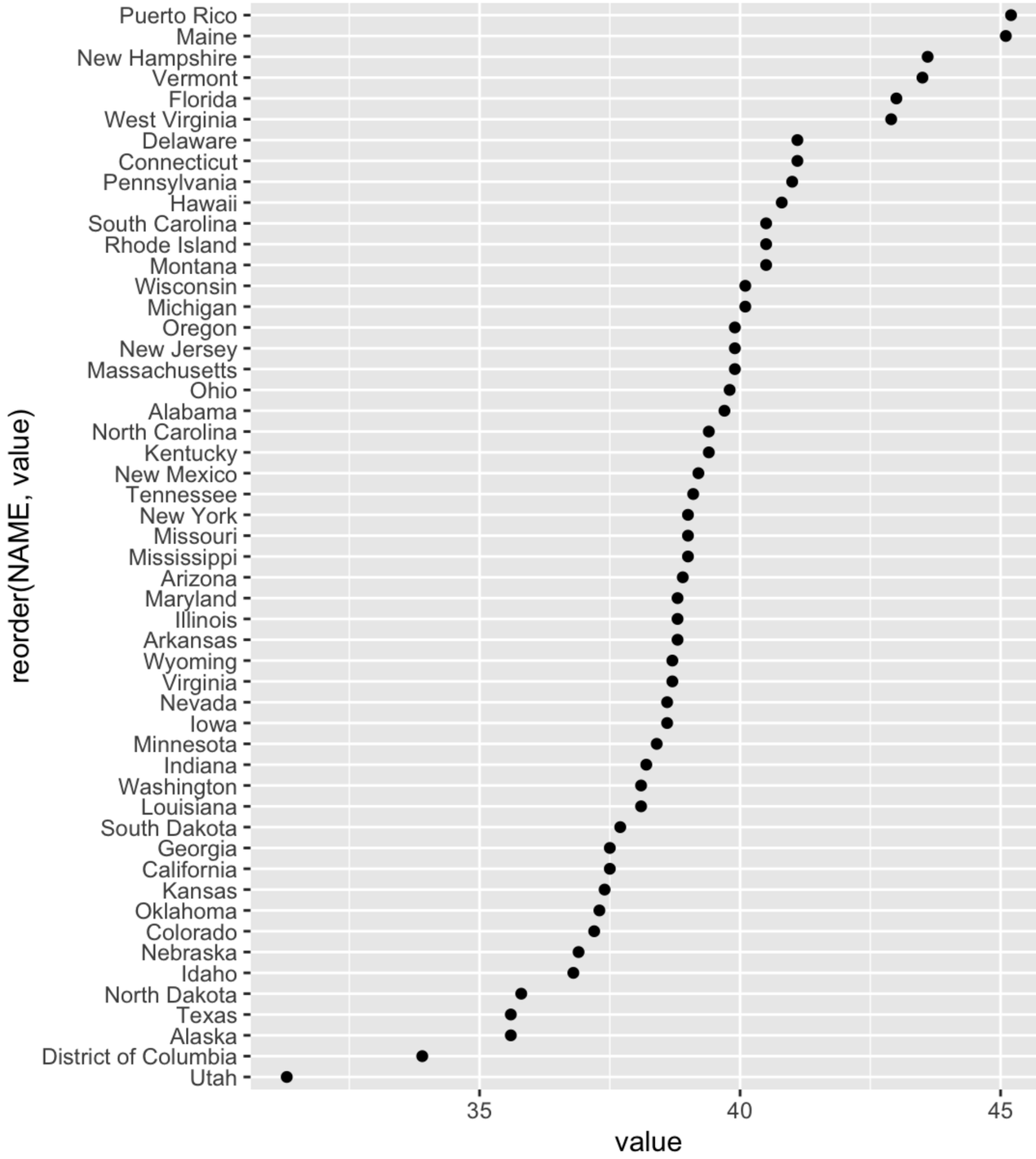
Let’s do an example: pull median age



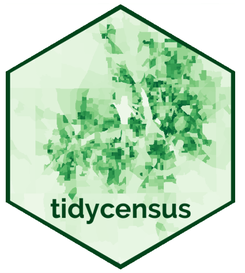

```
age20 <- get_decennial(geography = "state",
                       variables = "P13_001N",
                       year = 2020,
                       sumfile = “dhc") #demographic and housing file

head(age20)
# A tibble: 6 × 4
  GEOID NAME                 variable value
  <chr> <chr>                <chr>    <dbl>
1 09    Connecticut          P13_001N  41.1
2 10    Delaware             P13_001N  41.1
3 11    District of Columbia P13_001N  33.9
4 12    Florida              P13_001N  43
5 13    Georgia              P13_001N  37.5
6 15    Hawaii               P13_001N  40.8
```

Let’s do an example: median age, simple graph

```
age20 %>%
  ggplot(aes(x = value, y =
reorder(NAME, value))) +
  geom_point()
```



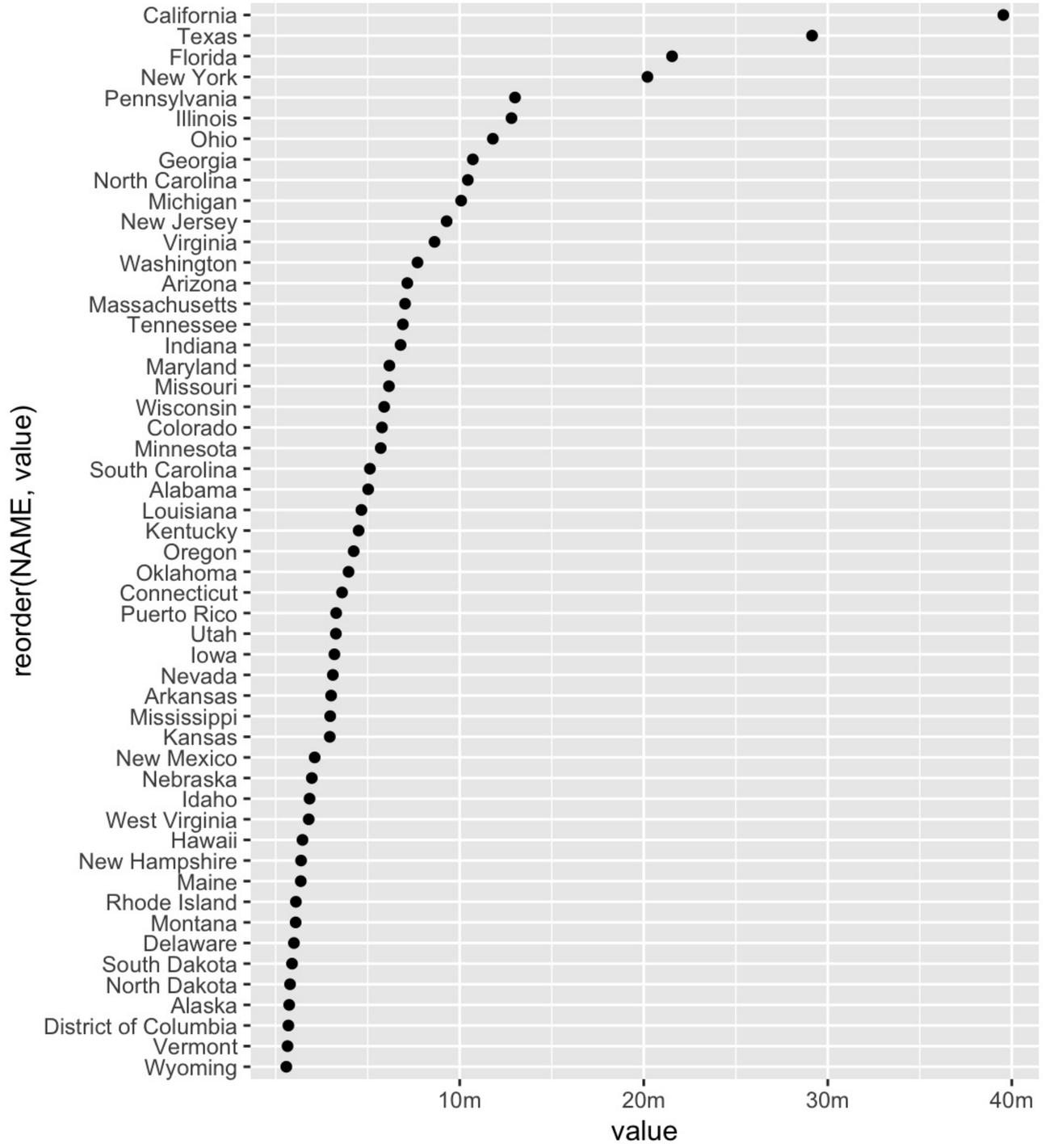
Another example: population




```
#Load all variable names (kind of the entire codebook) into the object vars
vars<- load_variables(2020, "dhc", cache=T)
vars

#Now, we'll search with the stringr package and regular expressions
library(stringr)
vars %>%
  filter(str_detect(concept, regex("population", ignore_case = T)))%>%
  View()
```

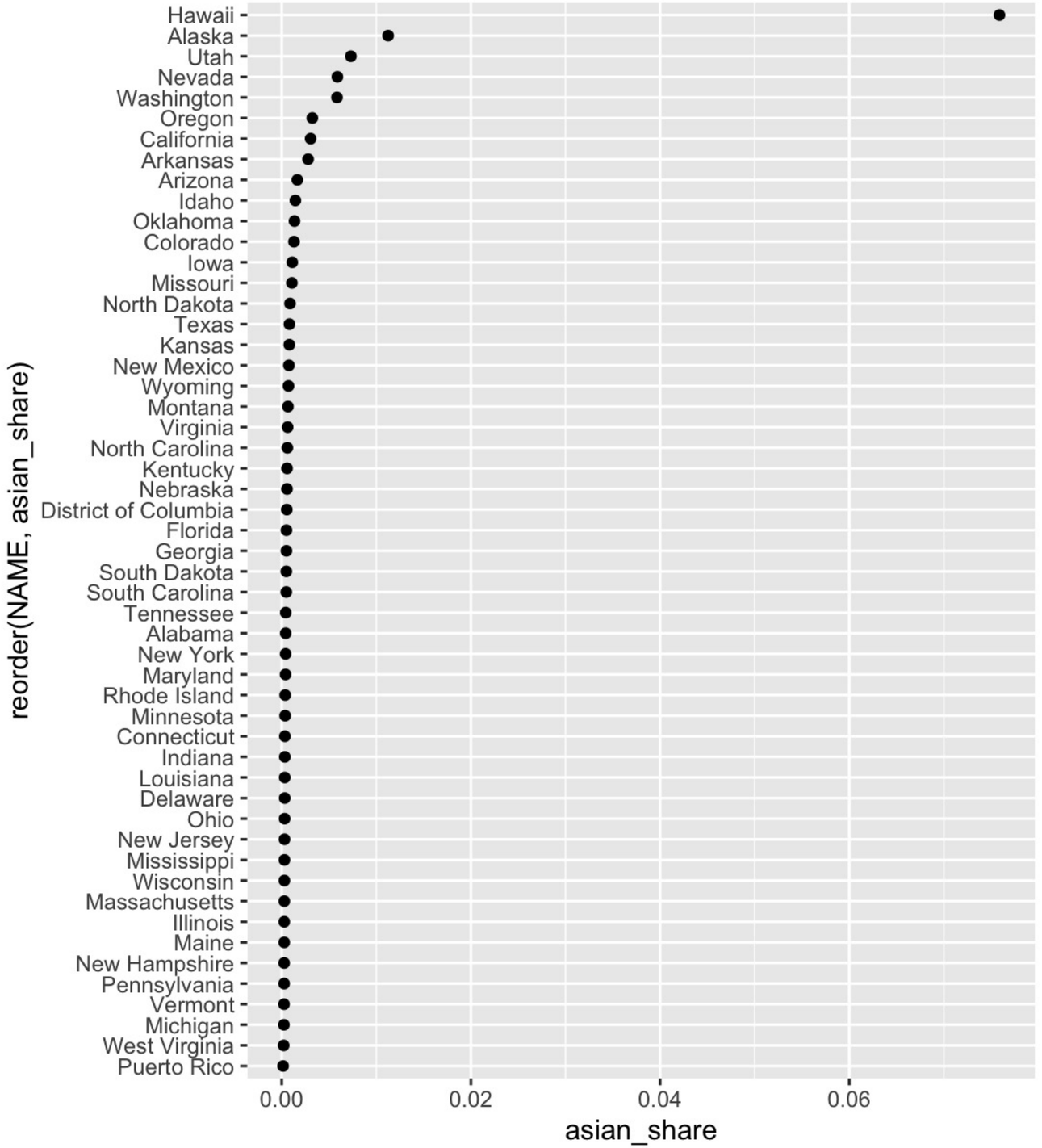
Some code

```
totalpop <- get_decennial(
geography = "state",
  variables = "P1_001N",
  year = 2020,
  sumfile = "dhc")
totalpop

totalpop %>%
  ggplot(aes(x = value,
y = reorder(NAME, value))) +
  geom_point() +
  scale_x_continuous(
    breaks = c(10000000, 20000000,
30000000, 40000000),
    labels = c("10m", "20m", "30m",
"40m")
  )
```



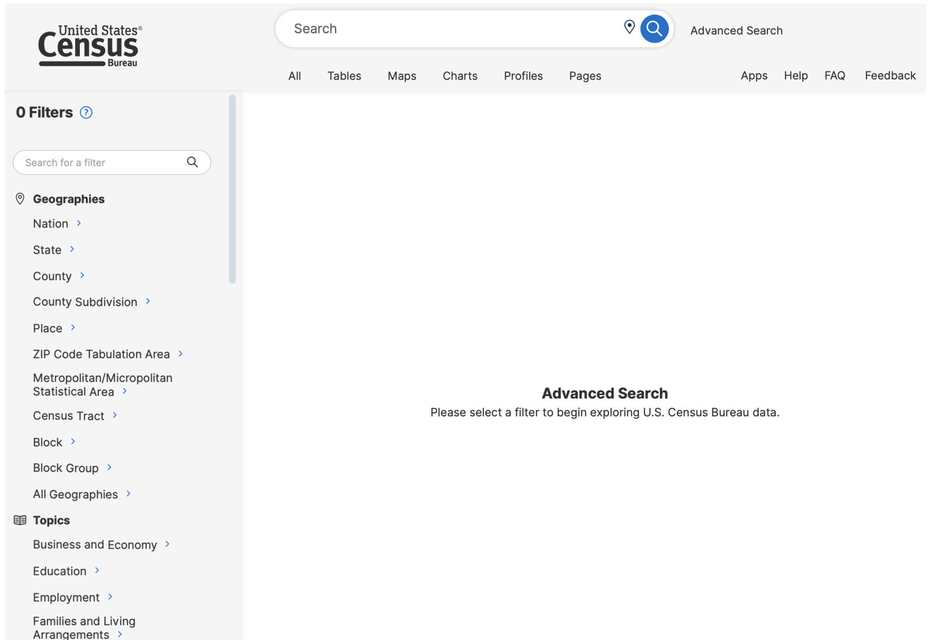
What if we wanted a ratio?

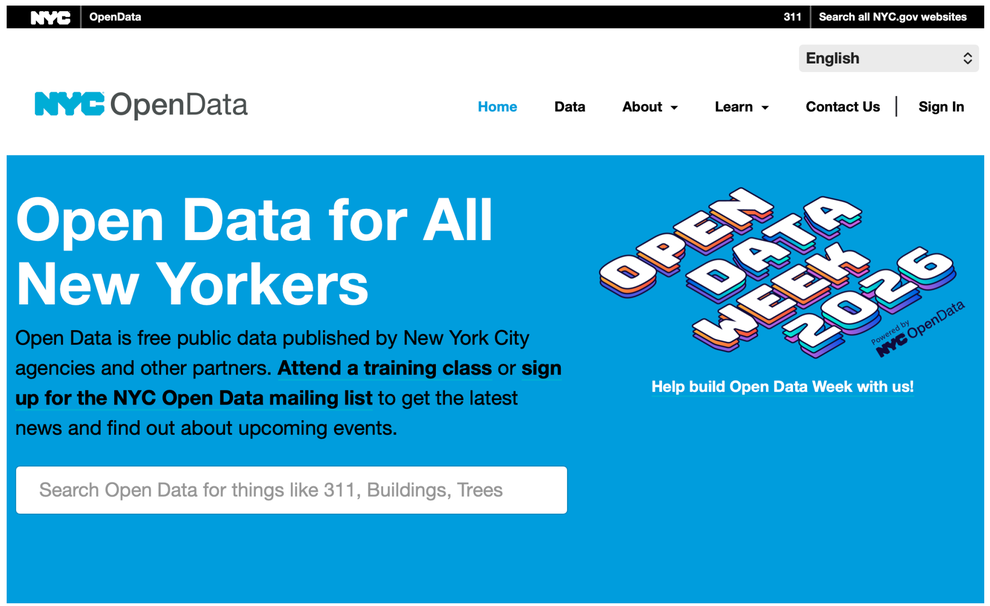
```
pop <- get_decennial(
        geography = "state",
        variables = c(
           total = "P1_001N",
           asian = "P10_007N"),
        year = 2020,
        sumfile = "dhc",
       output = "wide") #without this,
it comes back long

pop$ratio <- pop$asian/pop$total

pop %>%
  ggplot(aes(x = asian_share,
   y = reorder(NAME, asian_share))) +
  geom_point()
```



A practical word

- Many of these datasets have *non-programmatic* ways you can access them.
- Go to the Census, for instance, and specify and download a set
- Or just open the GSS





So, why?

- **Ease, replicability, portability**
  - The more *self-contained* your script files are, the easier they are to work with in the future
  - The GSS is more about convenience
  - The Census, on the other hand, is a different kind of thing

Wrapping Up

- This time:
  - Not as much “things you should memorize”, more “things you should be aware of” to look for if you need
- Next time
  - Workshop: Practicum 1; practicing from today
  - Next week: **Social Networks!**