Tools for a Code-based Data Workflow

McCrea Cobb
mccrea_cobb@fws.gov
907-786-3403
mccrea.cobb

Adam D. Smith
adam_d_smith@fws.gov
706-425-2197
adamdsmith

usfws.github.io/data-mgt-with-r

Outline

Outline

Review the data life cycle and data workflow

Outline

Review the data life cycle and data workflow

Present tools in  for efficiently and effectively working with data along the life cycle

Outline

Review the data life cycle and data workflow

Present some tools in  for efficiently and effectively working with data along the life cycle

Demonstrate a data workflow in  with a case study

How to draw an owl

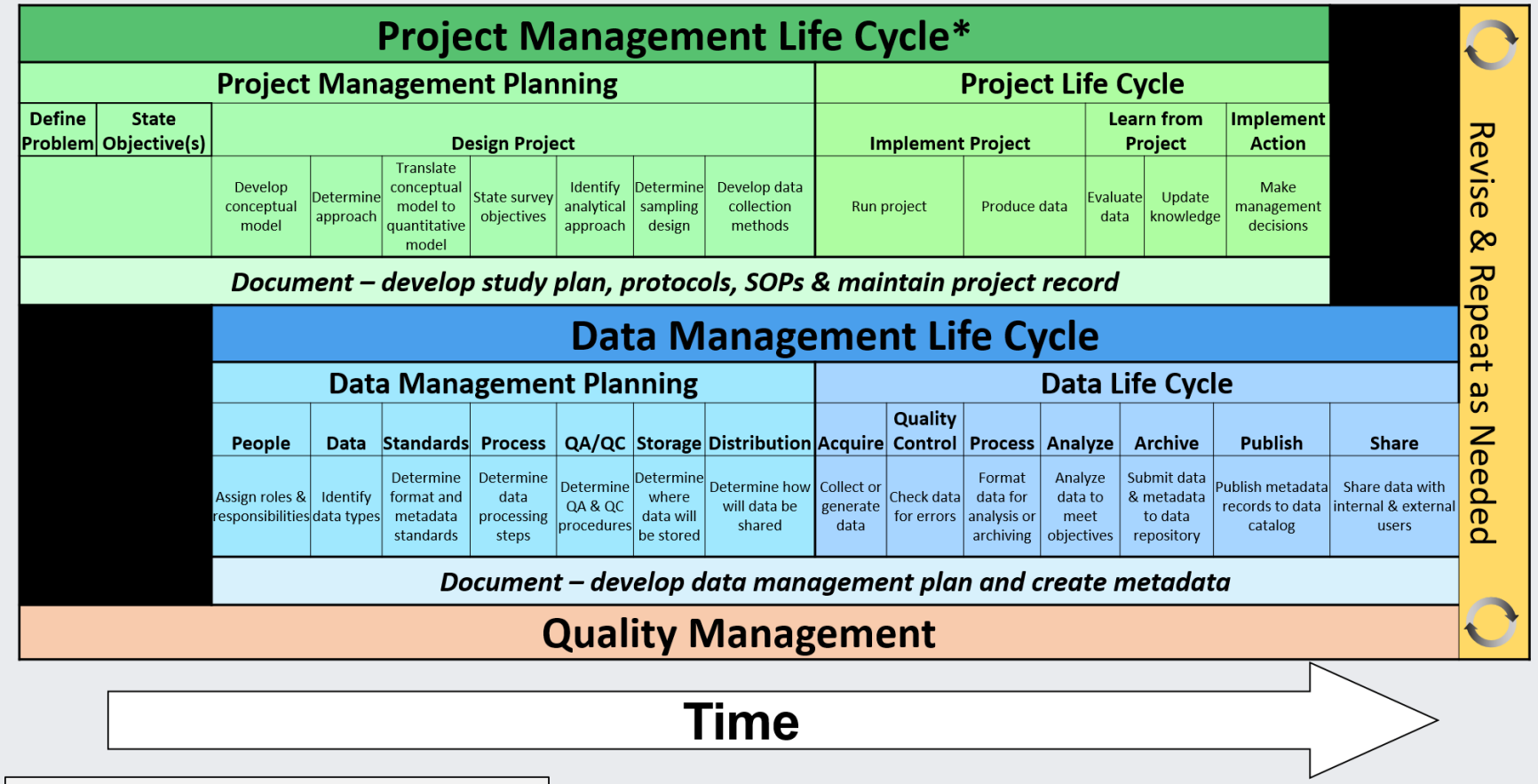
1.

2.

1. Draw some circles

2. Draw the rest of the owl

Project and data life cycles



Sources:
A Road Map for Designing and Implementing a Biological Monitoring Program, Reynolds et al. 2016
Alaska Region Interim Data Management User Guide, Alaska Region Data Stewardship Team

*Not to scale

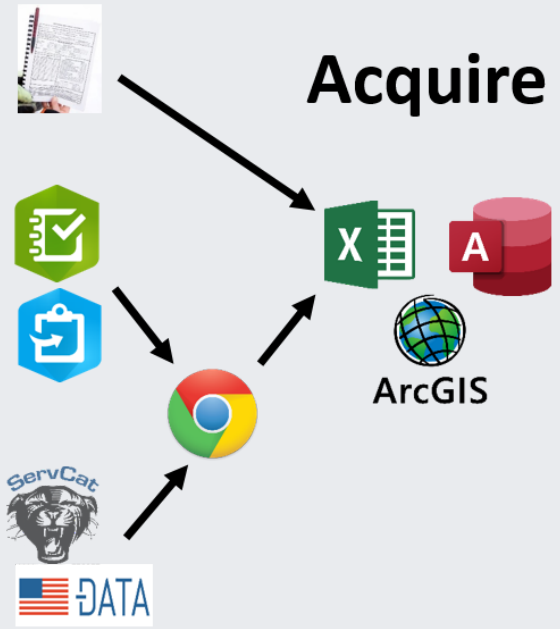
Data Management Life Cycle*

Data Management Planning							Data Life Cycle						
People	**Data**	**Standards**	**Process**	**QA/QC**	**Storage**	**Distribution**	**Acquire**	**Quality Control**	**Process**	**Analyze**	**Archive**	**Publish**	**Share**
Assign data management roles & responsibilities	Identify types of data to be collected or created	Determine metadata standards to be used	Determine how data will be manipulated	Determine data & metadata QA & QC procedures	Determine where data will be stored	Determine how will data be shared	Collect or generate data	Check data for errors	Manipulate data for analysis or final archiving	Produce results & products to meet objectives	Submit data & metadata to data repository	Publish metadata records to data catalog	Share data with internal & external users
Document data and data management strategy – data management plan, metadata													

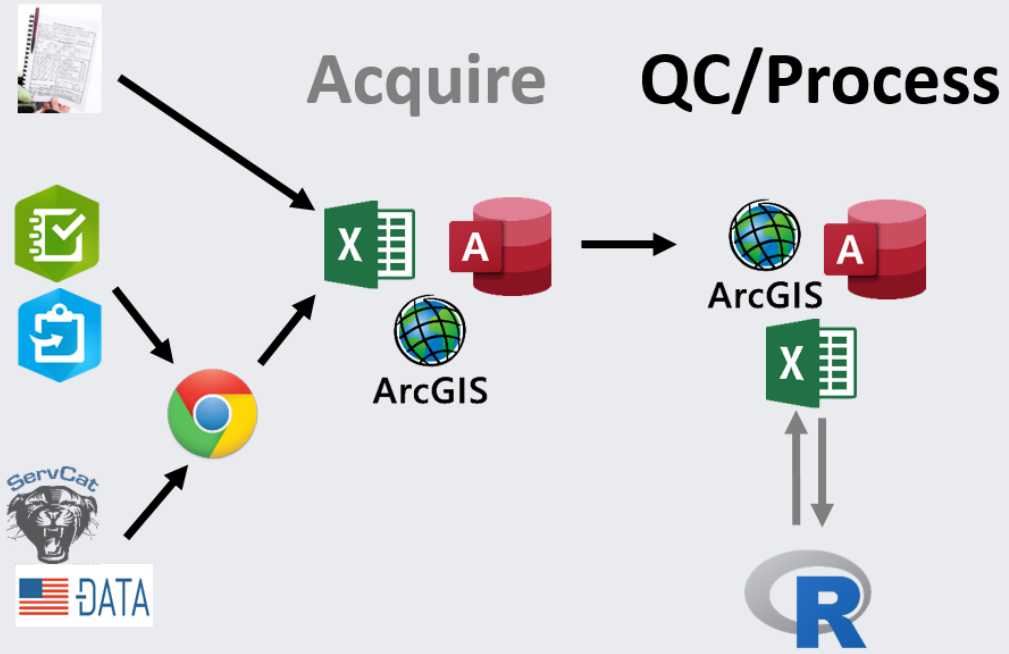
❌ Traditional data workflow

❌ Traditional data workflow

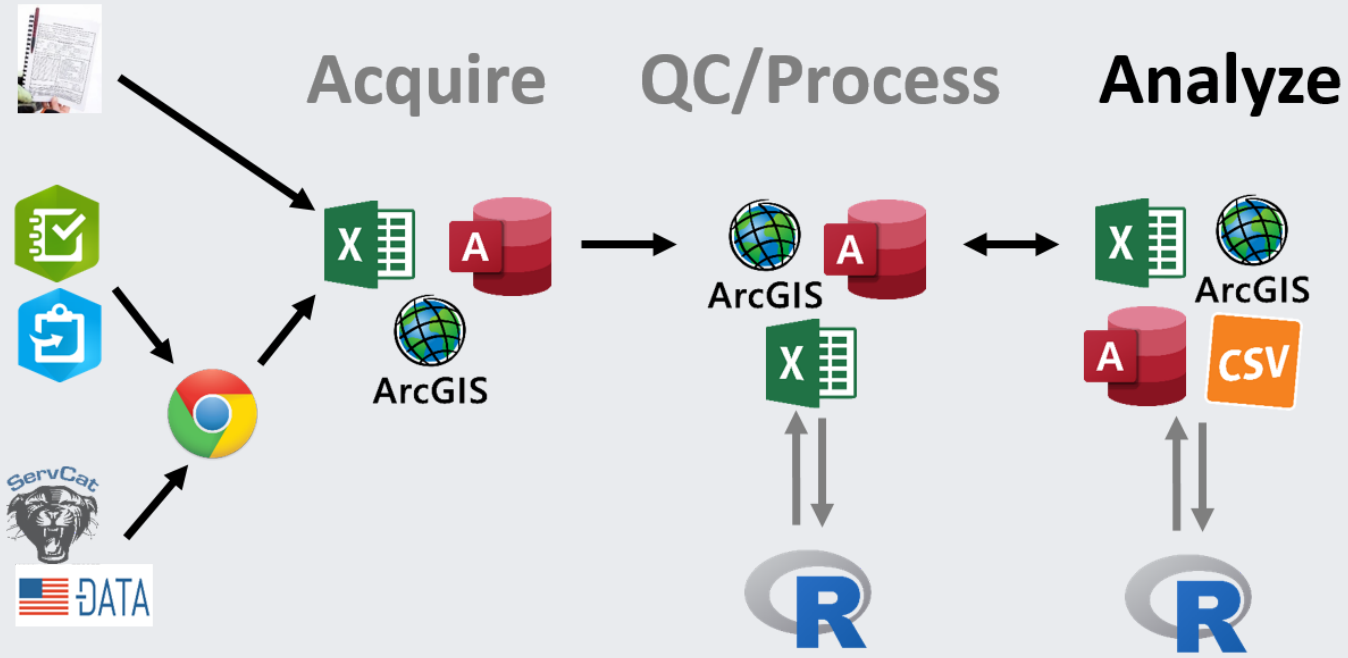
✗ Traditional data workflow



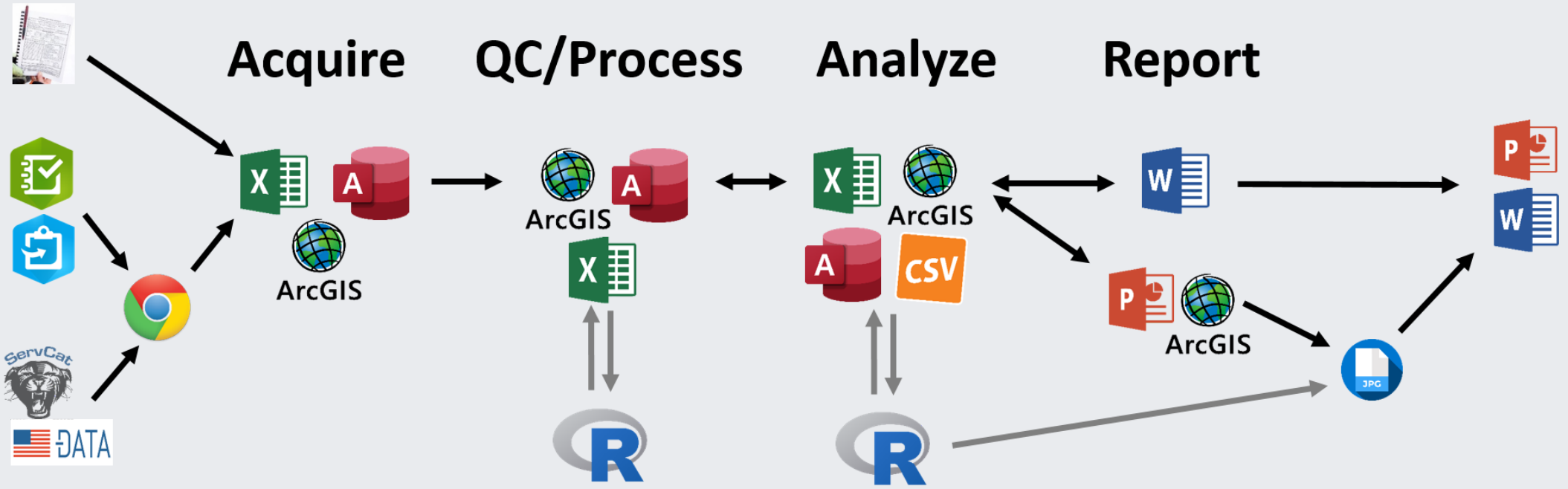
✗ Traditional data workflow



✗ Traditional data workflow



✗ Traditional data workflow ☹



Reproducible

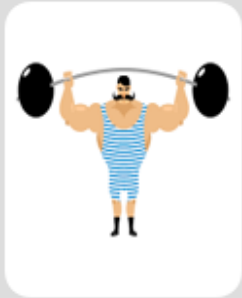
- consistently produces the same results given the same inputs

Efficient

- reduce redundancies, streamline, automated vs. manual tasks, common platforms

Replicable

- all stages of process can be completely and correctly repeated



Robust

- workflow is resilient to technological and human-caused errors

Documented

- external audiences can fully understand and execute procedural stages

Scalable

- ability to accommodate new data

Reproducible

- consistently produces the same results given the same inputs

Efficient

- reduce redundancies, streamline, automated vs. manual tasks, common platforms

Replicable

- all stages of process can be completely and correctly repeated

Robust

- workflow is resilient to technological and human-caused errors

Documented

- external audiences can fully understand and execute procedural stages

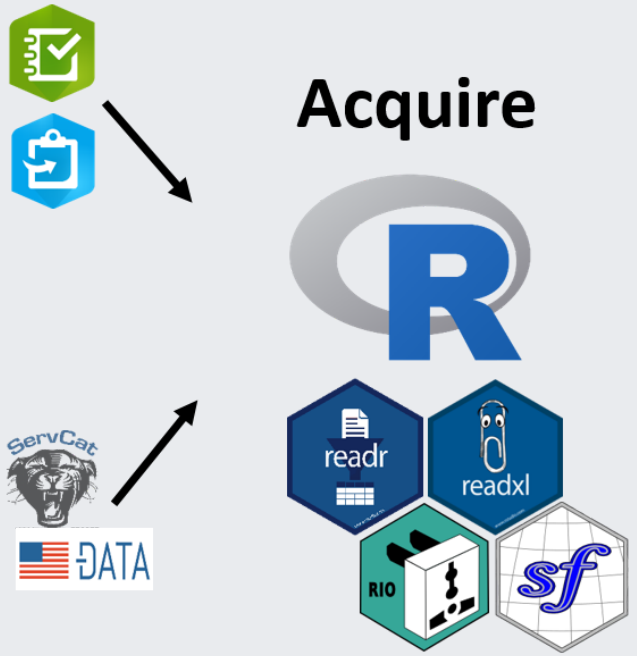
Scalable

- ability to accommodate new data

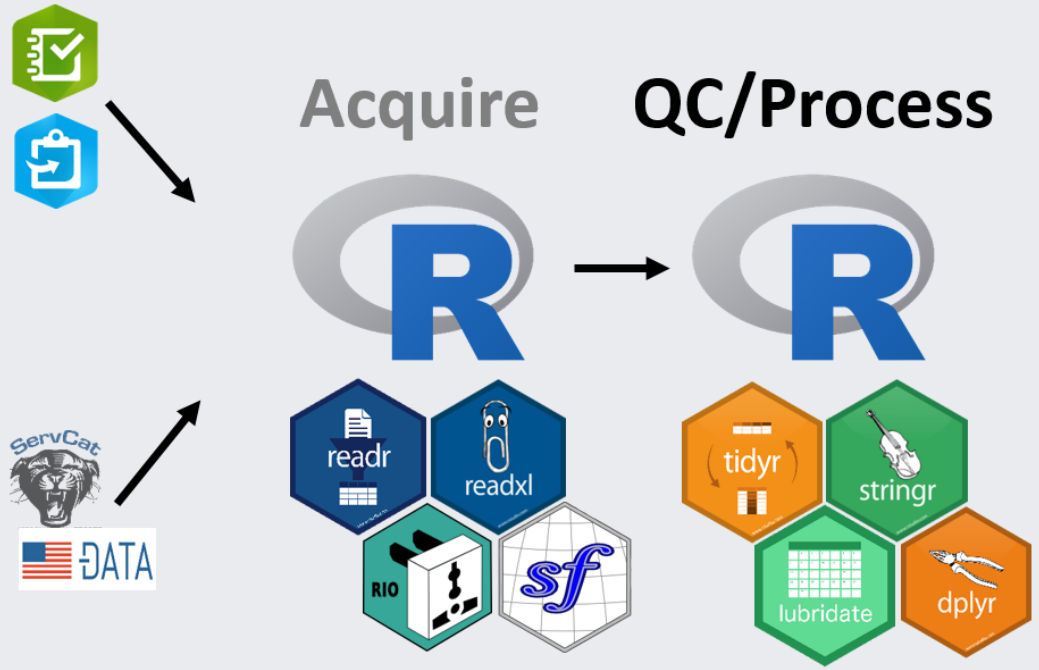

data workflow

data workflow

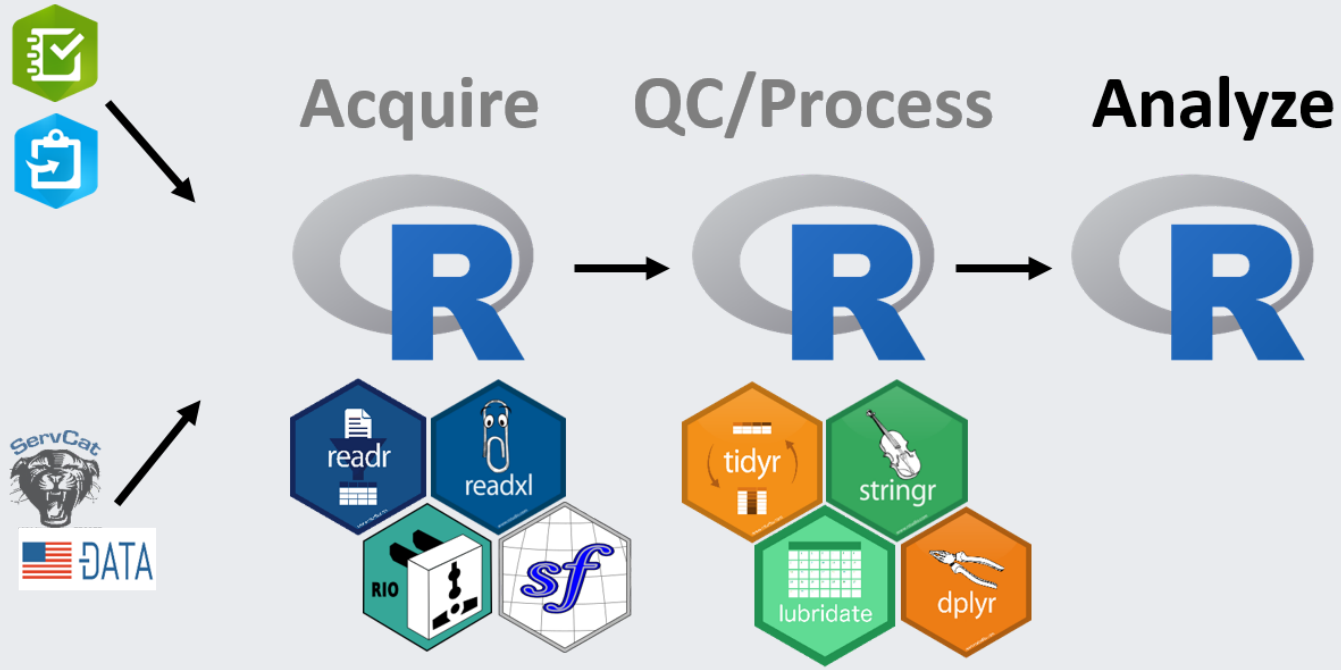
data workflow



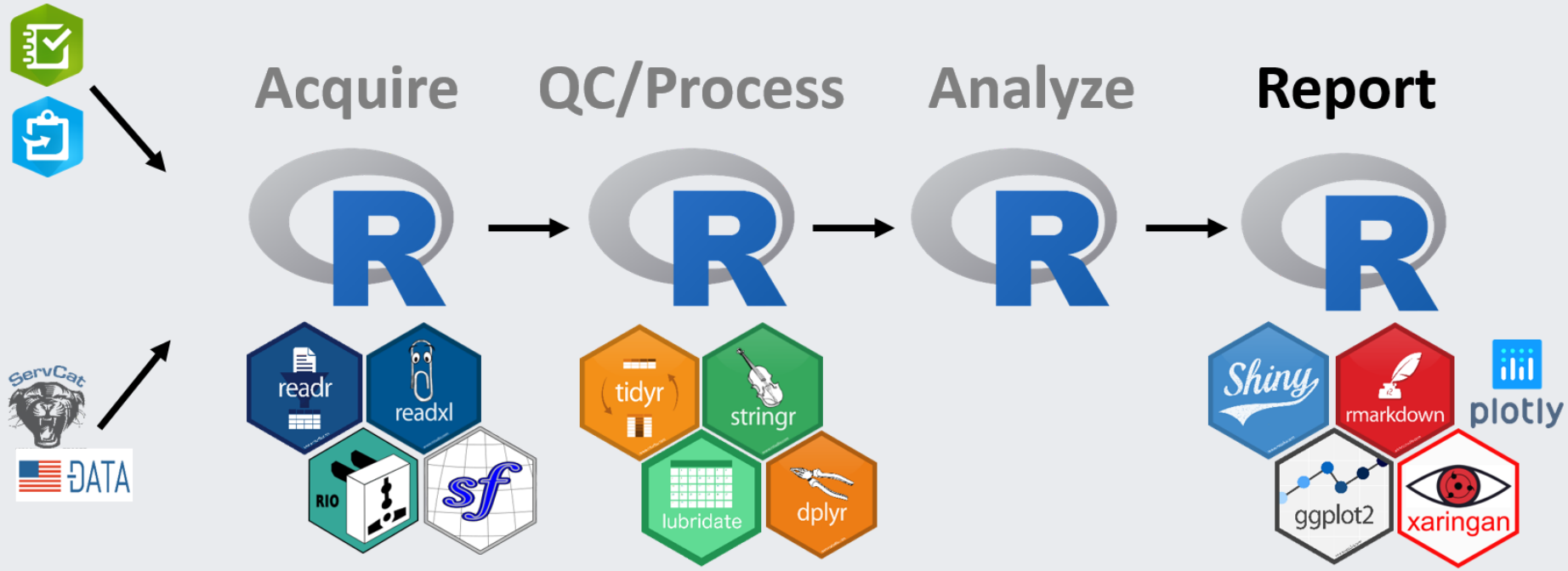
✓ R data workflow



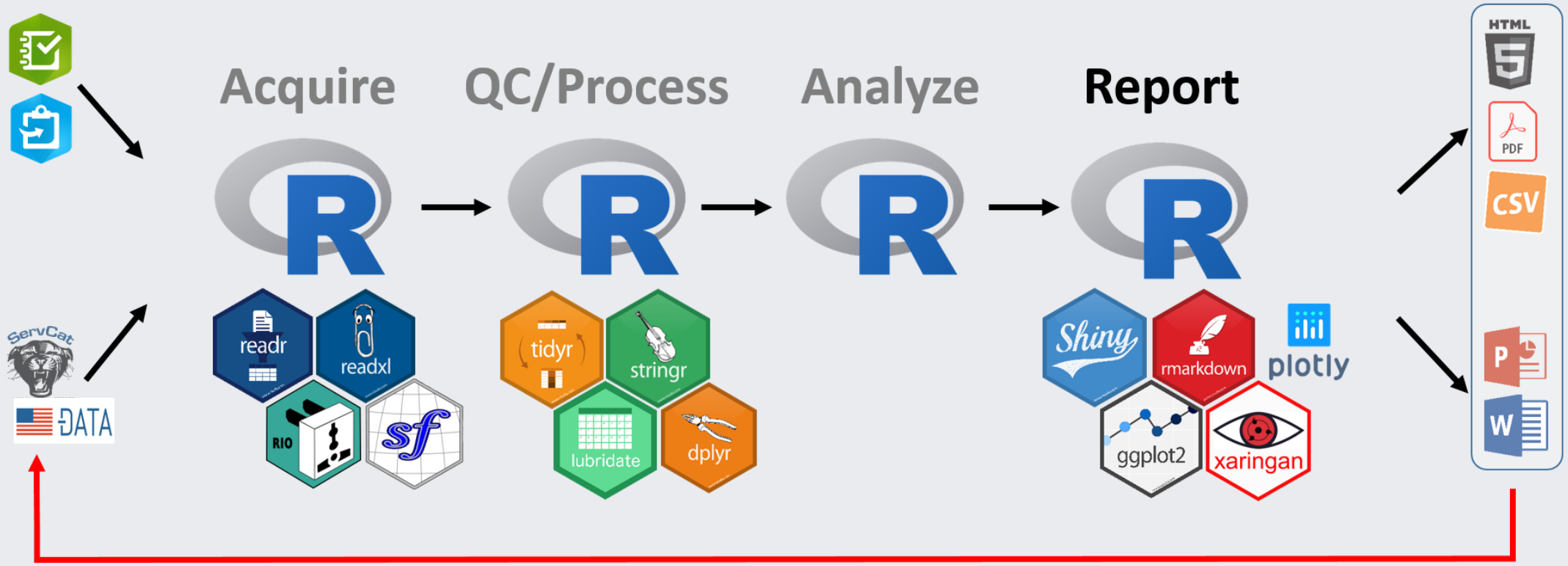
data workflow



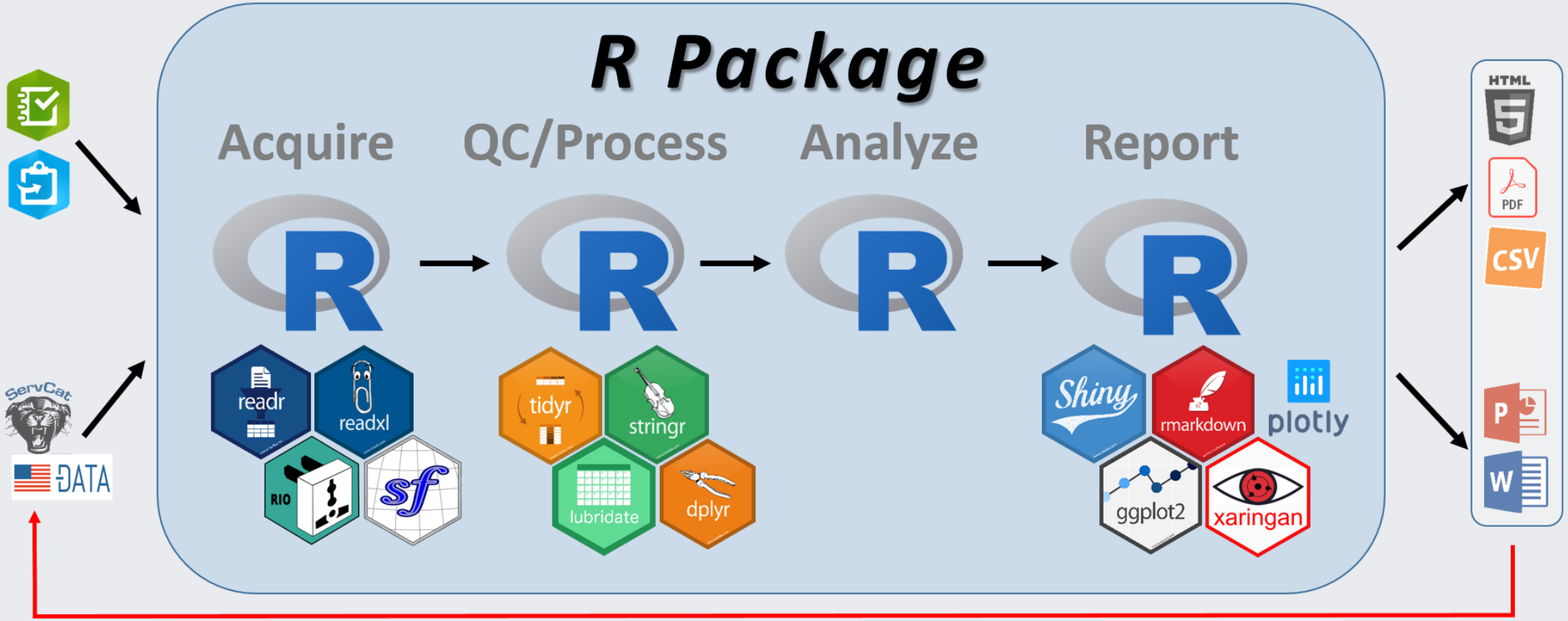
R data workflow



R data workflow



R data workflow



- Free
- Flexible
- Nice integrated development environment (RStudio)
- Large and active community of users
    - Over 15,000 packages

Planning

Data Management Life Cycle*

Data Management Planning

People	Data	Standards	Process	QA/QC	Storage	Distribution
Assign data management roles & responsibilities	Identify types of data to be collected or created	Determine metadata standards to be used	Determine how data will be manipulated	Determine data & metadata QA & QC procedures	Determine where data will be stored	Determine how will data be shared

Data Life Cycle

Acquire	Quality Control	Process	Analyze	Archive	Publish	Share
Collect or generate data	Check data for errors	Manipulate data for analysis or final archiving	Produce results & products to meet objectives	Submit data & metadata to data repository	Publish metadata records to data catalog	Share data with internal & external users

Document data and data management strategy – data management plan, metadata

Organizing an R project

Chose a standardized working directory structure

Provides consistent relative directory paths for your scripts

R packages provide functions to create a standard file directory:

- `MakeProject::MakeProject()`
- `rrtools::use_analysis()`
- `refugetools::create.dir()`
- `prodigenr::setup_project()`

```
project_name/
  admin/
  code/
    functions/
  data/
    derived_data/
    raw_data/
  incoming/
  metadata/
  output/
    figures/
    raw_analysis/
    tables/
  products/
  resources/
    data/
    publications/
    reports/
```

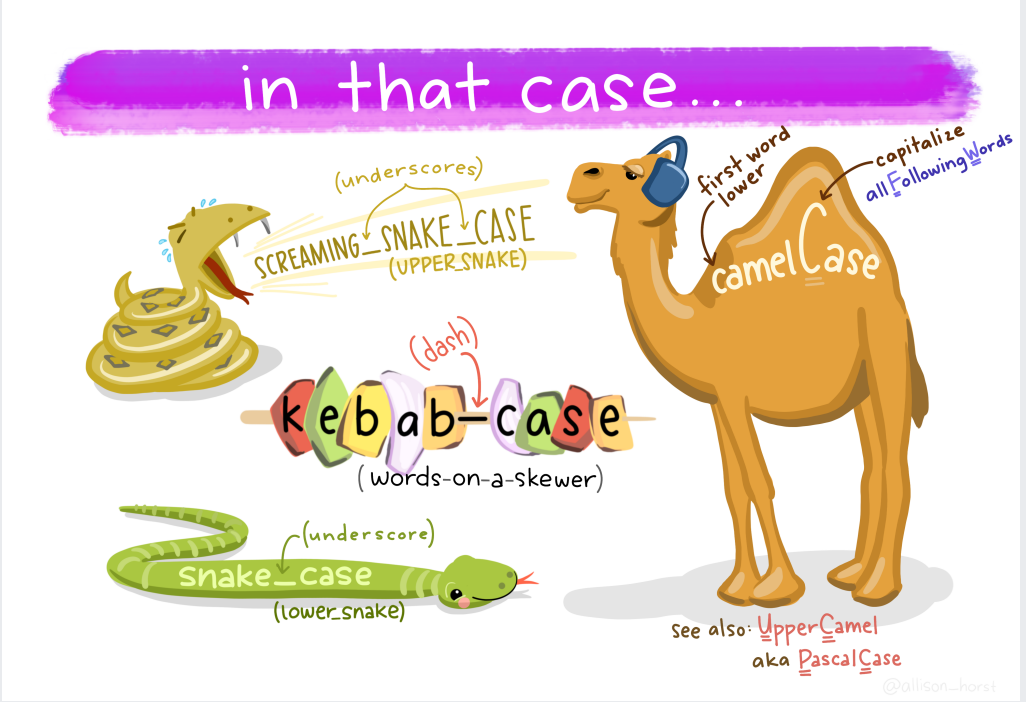
Organizing an R project

Decide on a standardized file naming convention

- Call files what they are
- Keep names short
- Avoid spaces and $pec!@l characters
- ISO 8601 for dates

Programmically filter files:

```
# Exclude files with 2016 or 2017 in their
   names
f ← list.files(path = "./filepath",
               pattern = "[^2016|2017].csv$")

# Load these files
dat ← lapply(f, read.csv)
```



Organizing an R project

Consult a style guide

"Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread"
-Hadley Wickham

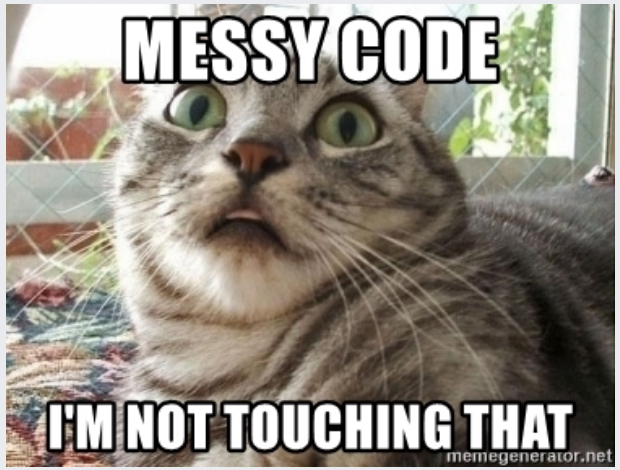
- Strive for consistent and meaningful names
- Review existing style guides:
  - tidyverse style guide
  - Advanced R style guide
  - Google's R style guide
- Helpful R packages: `styler` and `lintr`

```
# R code readability

if(readability()) {
  be_happy()
} else {
  rewrite_code()
}
```

`styler` package

Some messy code 😳

```
p =dat%>%#a comment
select(refuge ,lat, long)
%>%filter(refuge=='kodiak ')%>%
  ggplot(data,aes(x =lat,y=
long,group=refuge))%>%  geom_line()#comment
without space"
```



`styler` package

Some messy code 😐

```
p =dat%>%#a comment
select(refuge ,lat, long)
%>%filter(refuge≠'kodiak ')%>%
  ggplot(data,aes(x =lat,y=
long,group=refuge))%>%  geom_line()#comment
without space"
```

The `style_text()` function cleans it up! 😊

```
style_text("p =dat%>%#a comment
select(refuge ,lat, long)
%>%filter(refuge≠'kodiak ')%>%
  ggplot(data,aes(x =lat,y=
long,group=refuge))%>%  geom_line()#comment
without space")
```

```
p ← dat %>% # a comment
  select(refuge, lat, long) %>%
  filter(refuge ≠ "kodiak ") %>%
  ggplot(data, aes(x = lat, y = long,
   group = refuge)) %>%
  geom_line() # comment without space
```

Project dependencies

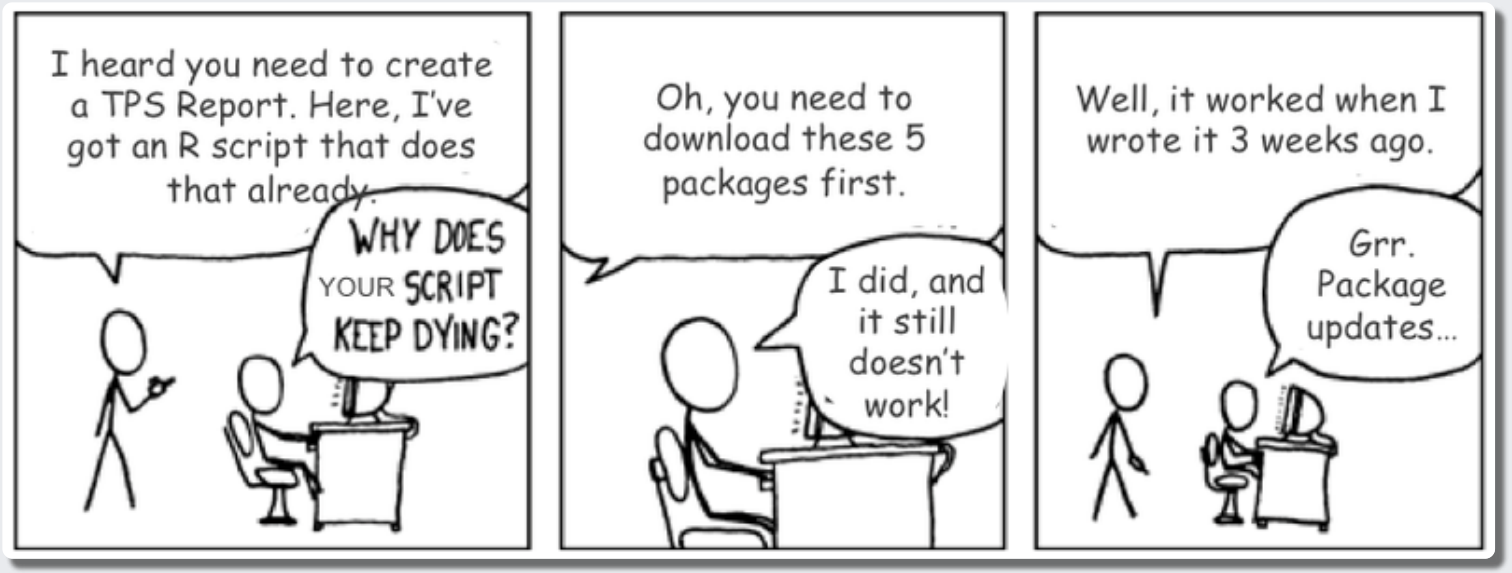
Maintaining dependencies can be frustrating!

An R project should be:

1. **Isolated**
2. **Portable**
3. **Reproducible**

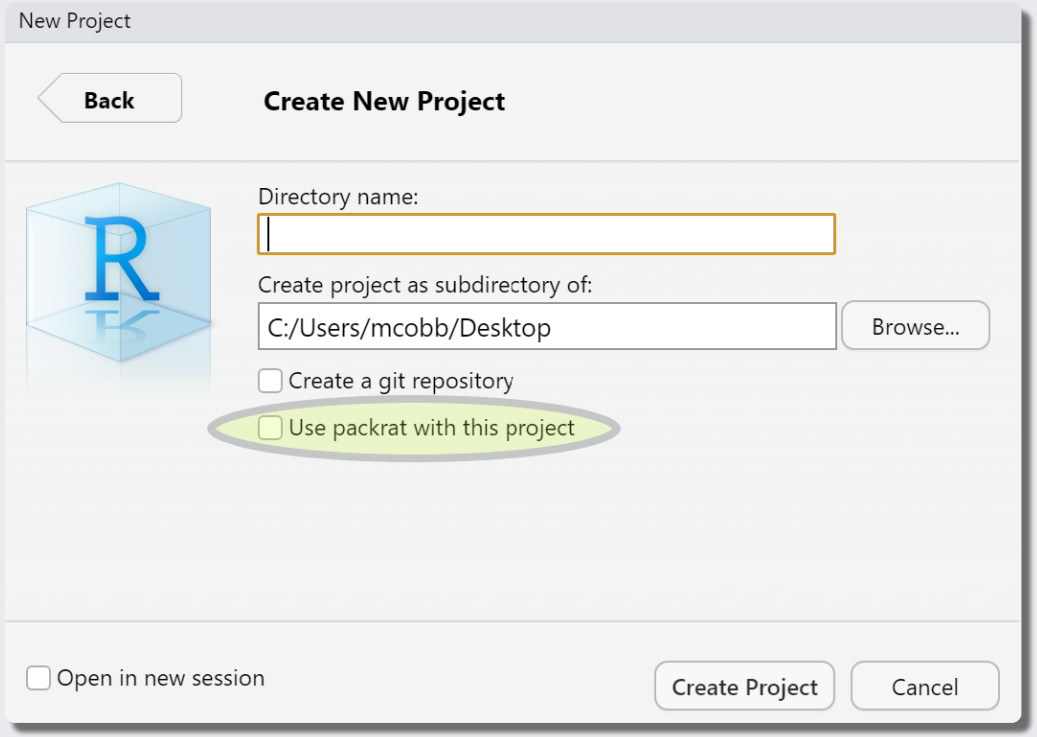
There are R tools to help with this:

- `packrat` package
- `rocker` package



Packrat

Option 1: Add to a new project



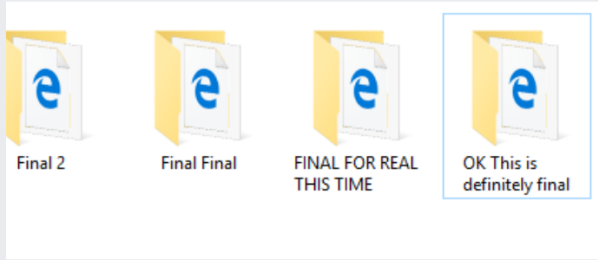
Option 2: Add to an existing project

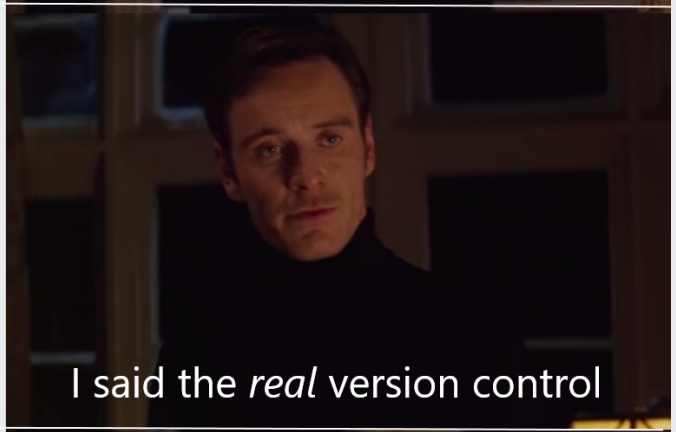
```
# Install packrat
install.packages('packrat')

# Set up your project to use packrat
packrat::init()
```

```
# Install required packages
install.packages('tidyverse')

# Take a snapshot to save the changes to
packrat
packrat::snapshot()
```

Version control

Consider how you will manage versions when ***planning*** your project!





Version control

- Benefits of GitHub
  - Collaboration
  - Storing versions
  - Restoring versions
  - Understanding what happened
  - Backup files
- RStudio makes version control with Git and GitHub easier
- Check out:
  - Resources to learn git (link)
  - Using git from RStudio (link)
  - GitHub and R: An Intro for FWS Biologists (link)

Document

Data Management Life Cycle*													
Data Management Planning							Data Life Cycle						
People	Data	Standards	Process	QA/QC	Storage	Distribution	Acquire	Quality Control	Process	Analyze	Archive	Publish	Share
Assign data management roles & responsibilities	Identify types of data to be collected or created	Determine metadata standards to be used	Determine how data will be manipulated	Determine data & metadata QA & QC procedures	Determine where data will be stored	Determine how will data be shared	Collect or generate data	Check data for errors	Manipulate data for analysis or final archiving	Produce results & products to meet objectives	Submit data & metadata to data repository	Publish metadata records to data catalog	Share data with internal & external users
Document data and data management strategy – data management plan, metadata													

Why document our code?

Reproducibility

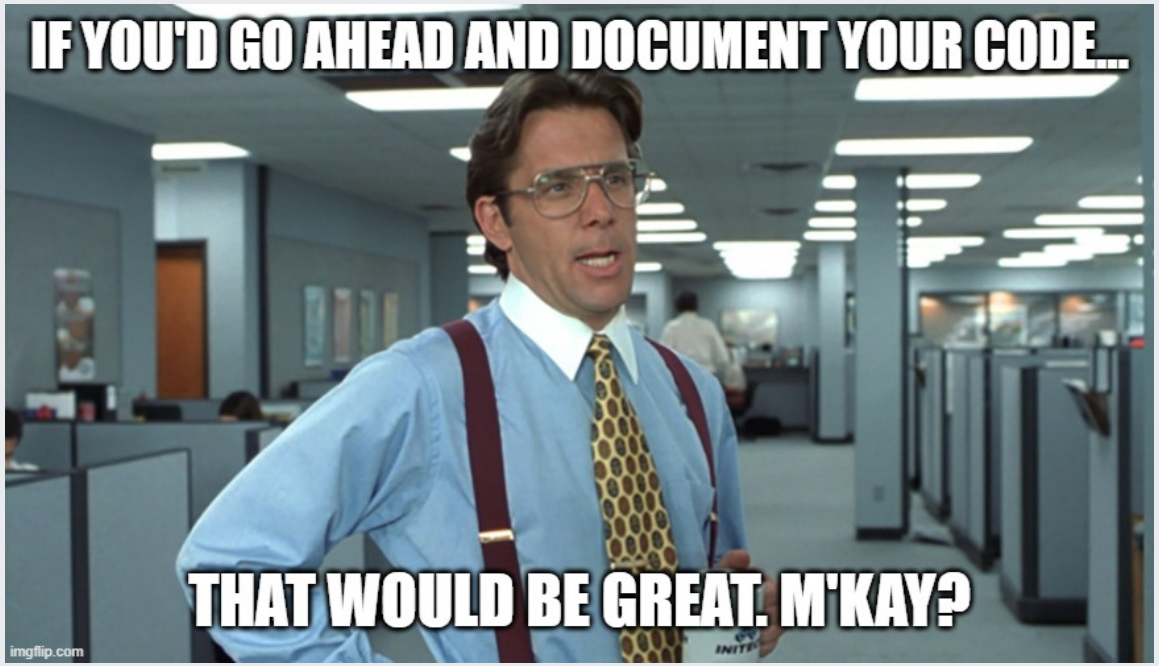
Make it reproducible for:

- colleagues
- future **you**!

Documenting basics

- Wilson et al. 2014 *Best practices for scientific computing*
- Wison et al. 2017 *Good enough practices in scientific computing*
- Lee 2018 *Ten simple rules for documenting scientific software*

Documenting basics



- Comment code as you go
- Name objects informatively
- Write modular code
- Include a project README
- Make dependencies explicit
- Record working environment
- Follow a style guide
- Version control

Comment code as you write it

- "Lab notebook" for code
- Write for people, not computers
- Bare minimum documentation

Commenting R code

- Comment marker in R: # (number sign/hash)

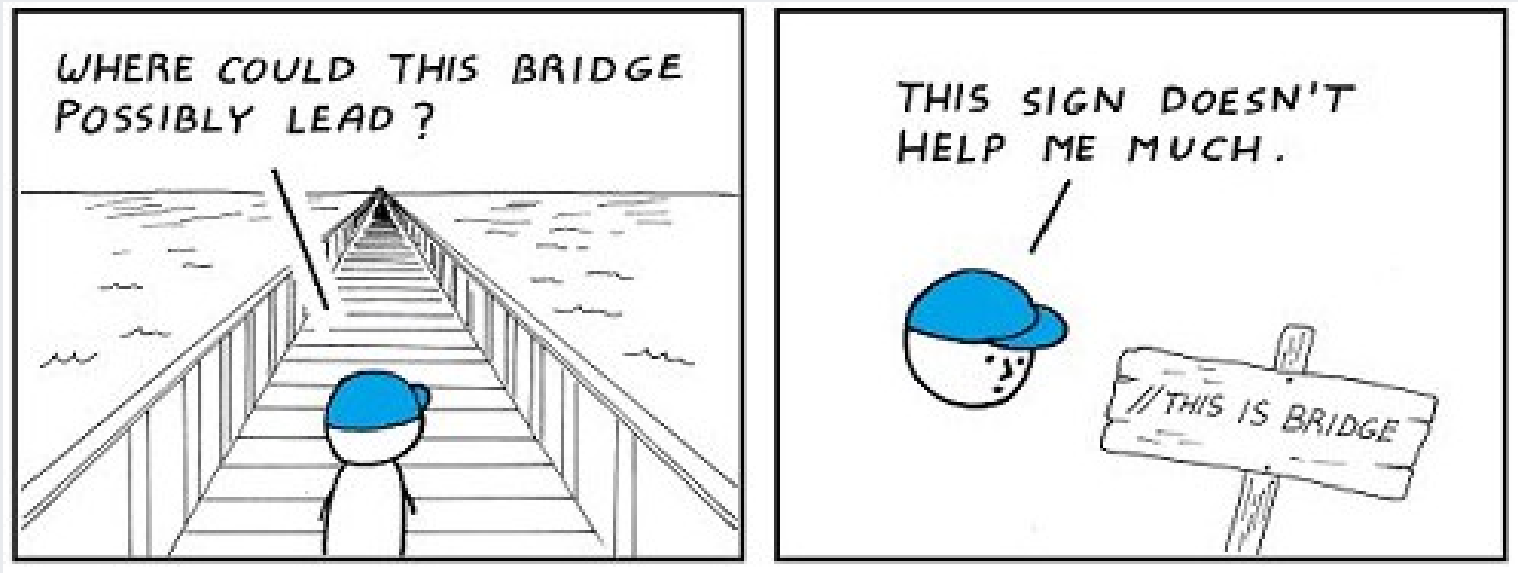
```
# This is a comment

x ← 2 # This is also a comment
```

- Barest minimum: brief explanatory description at top of file

```
# This code retrieves historical METAR weather from NOAA
# buoys; see http://www.ndbc.noaa.gov/. This function
# wraps rnoaa::buoy() to allow multiple year queries
```

Comment code as you write it

- Better: comment **succintly** throughout code
- Focus on your intentions, not mechanics: **"why"** not "what" nor "how"



Comment code as you write it

LESS HELPFUL

```
# Import data
df ← rio::import("survey_data.xlsx")

# Define new variable
df ← mutate(df, above_water = above / (above + below))
```

BETTER

```
# Import manatee aerial survey data
df ← rio::import("survey_data.xlsx")

# Calculate proportion of manatees above water surface at each location
df ← mutate(df, above_water = above / (above + below))
```

Name objects informatively

LESS HELPFUL

```
# Data object naming
df ← rio::import("birds.csv")
df2 ← rio::import("veg.csv")

# Variable naming
r ← 100
a ← pi * r ^ 2

# Function naming
my_fun ← function(r) {
  pi * r ^ 2
}
```

BETTER

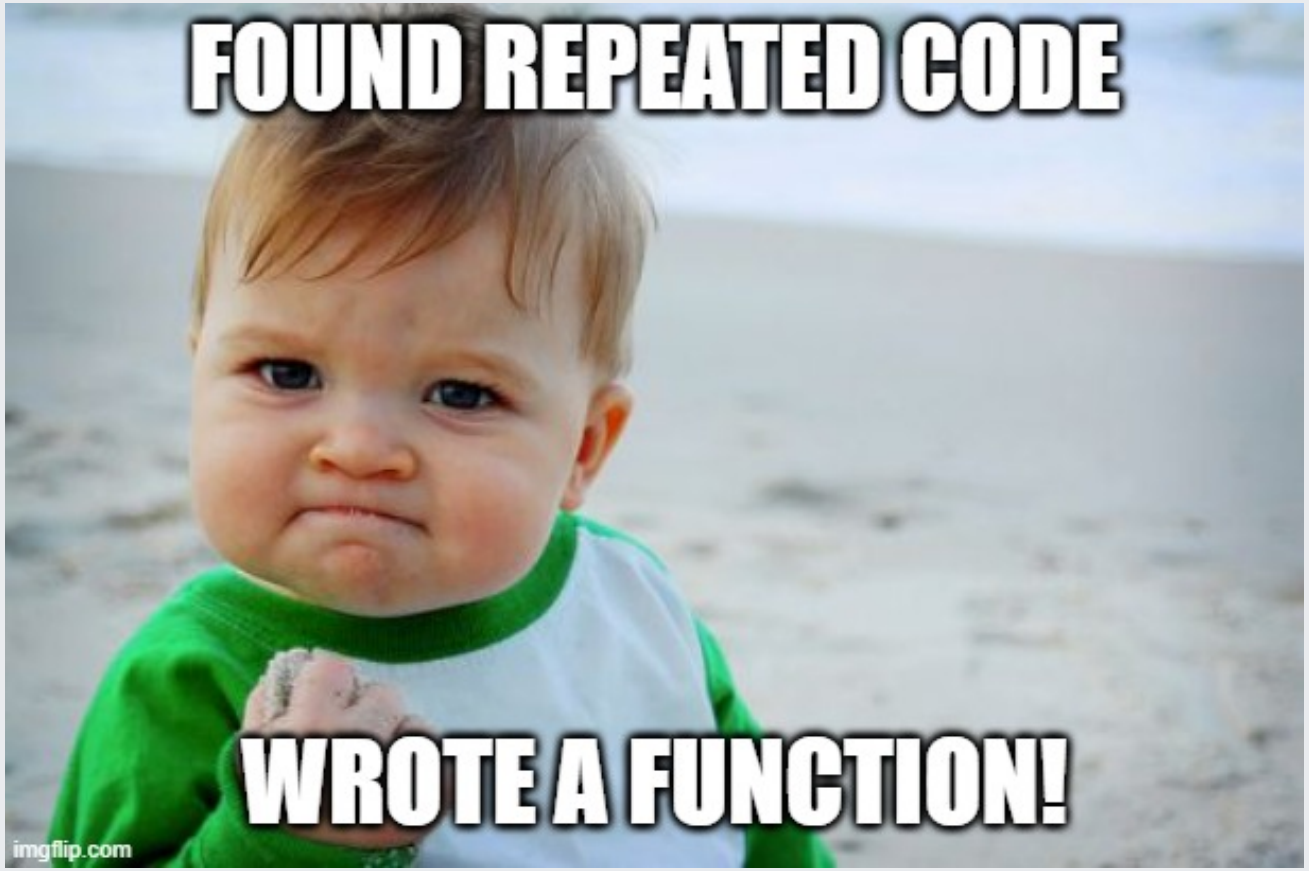
```
# Data object naming
bird_counts ← rio::import("birds.csv")
veg_data ← rio::import("veg.csv")

# Variable naming
survey_radius ← 100
survey_area ← pi * survey_radius ^ 2

# Function naming
calc_circle_area ← function(radius) {
  pi * radius ^ 2
}
```

Modularize your code

- Separate scripts for separate tasks
- Execute or load the functionality with the `source` function

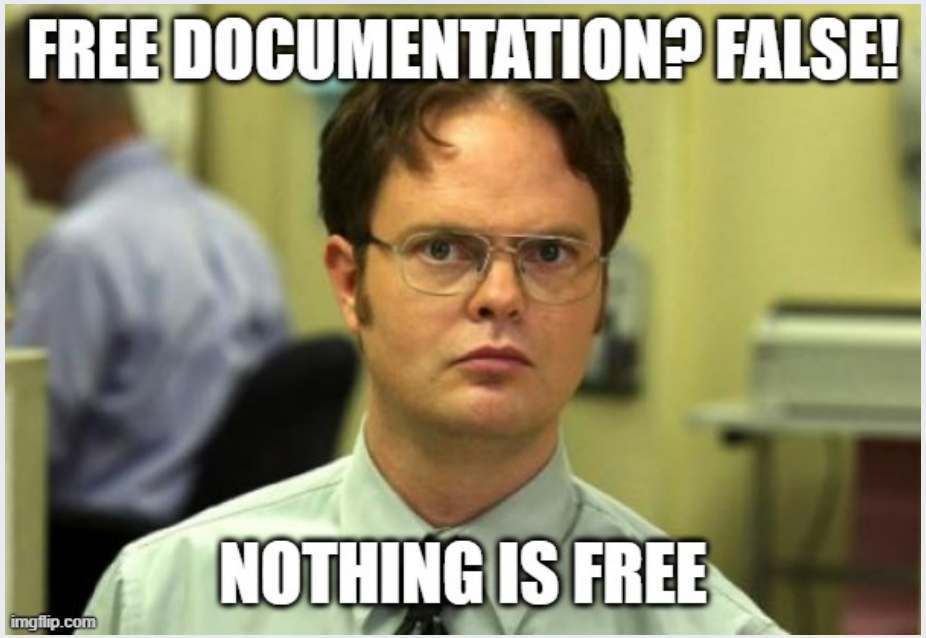
```
source("code/00_load_dependencies.R")
source("code/01_import_data.R")
source("code/02_tidy_data.R")
source("code/03_fit_regression_models.R")
source("code/04_generate_figures.R")
source("code/05_generate_report.R")
```

```r
source("code/functions/createAwesomePlot.R")
figure_2 ← createAwesomePlot(data = bird_counts, outpath = "output/birds_plot.png")
figure_3 ← createAwesomePlot(data = veg_data, output = "output/veg_plot.png")
```



Documentation for (nearly) free!

- `roxygen2`
  - specialized comments (`#'`)
  - translated into documentation



`roxygen2` example

Include a README file

- Briefly introduce your project to users before they access the code

General Structure

1. What your code is/does (with context)
2. Demonstrate how to use your code (install, configure, find help)
3. What your code looks like in action (examples)
4. Other relevant details (license, acknowledgments, working environment)

Include a README file

Handy templates available

- e.g., `usethis::use_readme_rmd`

Bonus points: create a vignette

Make dependencies explicit

Package dependencies of code

- Load required packages at beginning of script

```
# General usage
library(package)

# Example
library(dplyr)
```

Make dependencies explicit

Function dependencies (package source)

- Identify the package source of used functions
- Avoids conflicts when multiple packages share function names

```
# General usage
package::function()

# Examples
rio::import()
dplyr::mutate()
```

Record working environment

- Version info for R, the OS, and packages
- `sessionInfo()` or `devtools::session_info()`

```
> sessionInfo()
R version 3.6.1 (2019-07-05)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 17134)

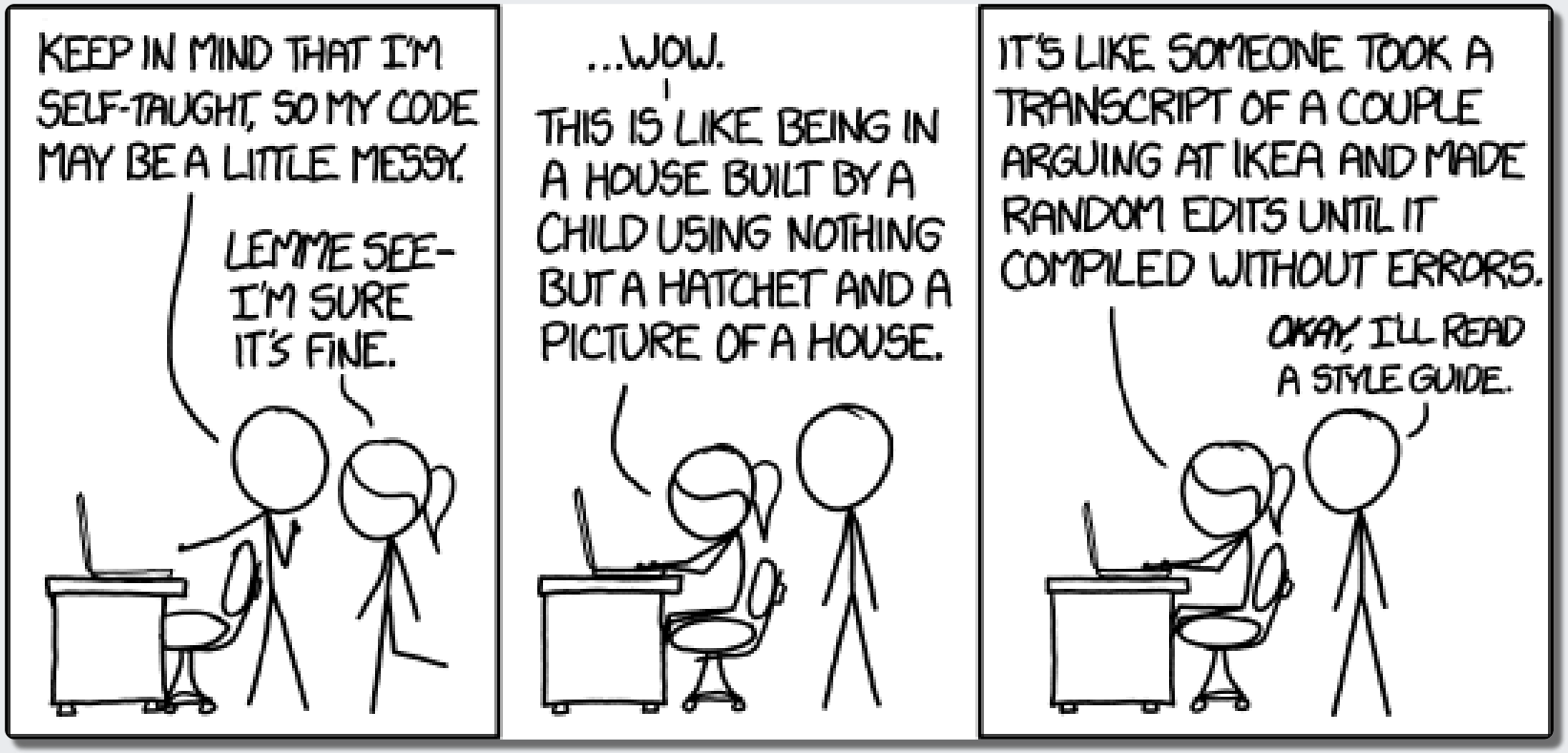
Matrix products: default

locale:
[1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252    LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C                           LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] tidyr_1.1.0          jsonlite_1.6.1       StanHeaders_2.21.0-1 assertthat_0.2.1     stats4_3.6.1         pander_0.6.3
 [7] cellranger_1.1.0     yaml_2.2.1           pillar_1.4.4         backports_1.1.6      glue_1.4.1           digest_0.6.25
[13] pryr_0.1.4           checkmate_2.0.0      colorspace_1.4-1     htmltools_0.4.0      plyr_1.8.6           xaringan_0.16
[19] pkgconfig_2.0.3      rstan_2.19.3         haven_2.2.0          magick_2.3           purrr_0.3.4          scales_1.1.1
[25] processx_3.4.2       openxlsx_4.1.5       rio_0.5.16           tibble_3.0.1         generics_0.0.2       ggplot2_3.3.1
[31] ellipsis_0.3.1       DT_0.13              pacman_0.5.1         repr_1.1.0           skimr_2.1.1          cli_2.0.2
[37] magrittr_1.5         crayon_1.3.4         readxl_1.3.1         evaluate_0.14        ps_1.3.2             fansi_0.4.1
```

Follow (or at least read) a style guide



Put documentation under version control

Acquire

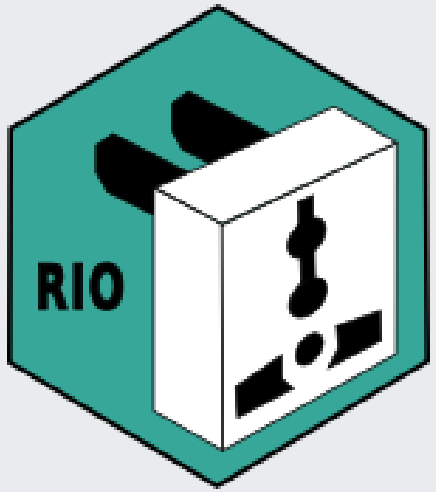
Data Management Life Cycle*													
Data Management Planning							Data Life Cycle						
People	Data	Standards	Process	QA/QC	Storage	Distribution	Acquire	Quality Control	Process	Analyze	Archive	Publish	Share
Assign data management roles & responsibilities	Identify types of data to be collected or created	Determine metadata standards to be used	Determine how data will be manipulated	Determine data & metadata QA & QC procedures	Determine where data will be stored	Determine how will data be shared	Collect or generate data	Check data for errors	Manipulate data for analysis or final archiving	Produce results & products to meet objectives	Submit data & metadata to data repository	Publish metadata records to data catalog	Share data with internal & external users
Document data and data management strategy – data management plan, metadata													

Data sources

- Flat file database
  - e.g., `csv`, `xls`, `tsv`, `txt`, etc.
- Relational database
  - e.g., Access, SQL, etc.
- web API (open or internal)
  - AGOL, ServCat, IRMA, ...

Acquiring data in flat files

- `rio`
- Opinionated defaults simplify data I/O
- `import`, `export`, and `convert` functionality



Supported `rio` file formats

Format	Typical Extension
Comma-separated data	.csv
Pipe-separated data	.psv
Tab-separated data	.tsv
Excel	.xls
Excel	.xlsx
SAS	.sas7bdat
SPSS	.sav
Stata	.dta

Showing 1 to 8 of 24 entries

Previous 1 2 3 Next

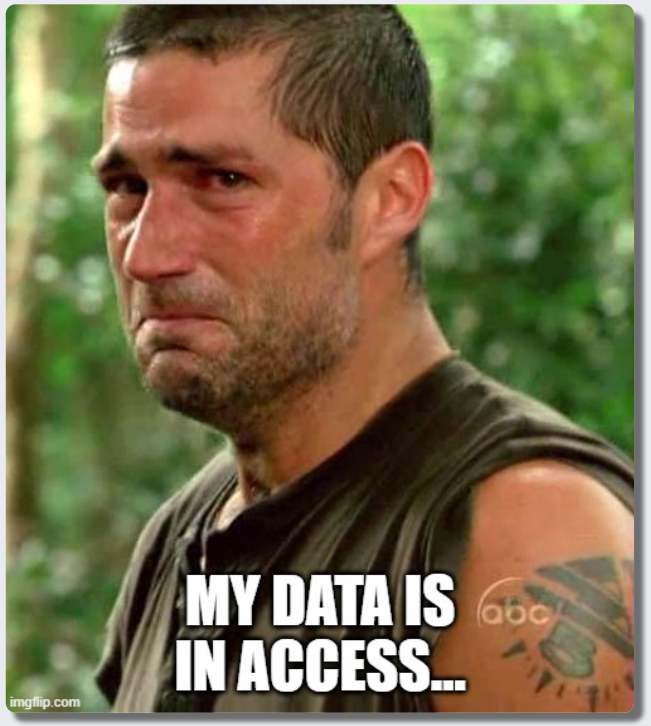
`rio` example - local file or web URL

```
species ← rio::import("data/species.csv")
species ← rio::import("https://motus.org/data/downloads/api-proxy/species")
```

english	scientific
Ruddy Turnstone	Arenaria interpres
Painted Bunting	Passerina ciris
Red Knot	Calidris canutus
Saltmarsh Sparrow	Ammospiza caudacuta

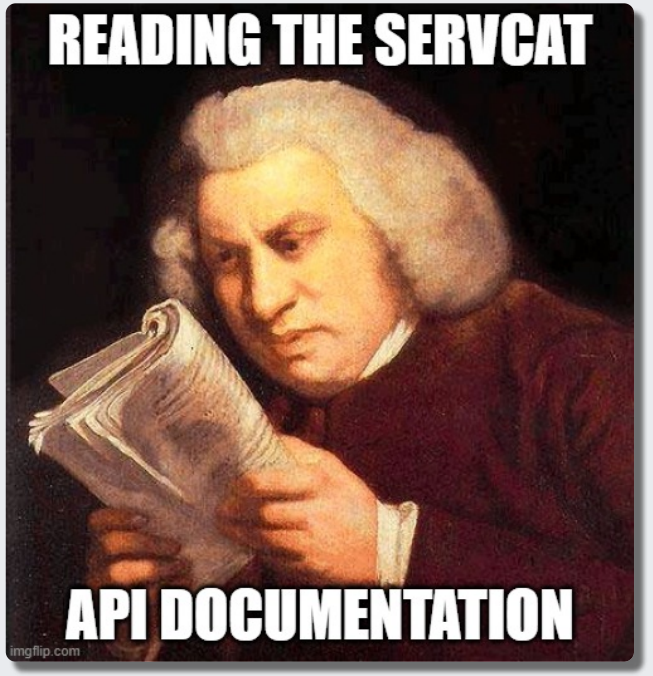
Acquiring data in relational databases

- several database architectures supported
- SQL-related options supported generally
  - can execute queries prior to import (`dplyr`)
- Communication with MS Access is problematic
  - export flat files (e.g., `csv`, `xlsx`) for `rio`



Acquiring data behind web APIs

- Some APIs already have dedicated R packages
  - ScienceBase, eBird, GBIF, BISON, iNaturalist, ITIS
- Do-It-Yourself
  - AGOL, ServCat, IRMA, PRIMR, data.gov
  - `httr` and `crul` packages



RCW Cluster Status_stakeholder

☆ Add to Favorites

A brief summary of the item is not available.

Feature Layer by zachary_cravens@fws.gov_fws

Created: Feb 13, 2020 Updated: Feb 13, 2020 View Count: 3

Description

An in-depth description of the item is not available.

Layers

RCW_Cluster_Status

Open In Export To Service URL

ges in

Descript

This group
National Wi
larger Protc
census or e
survivorshi
as well as d
The framew
to understa

V) on
part of the
opulation
ual adult
birds, video
and adults.
as a means
rs on each

refuge to further species recovery. When refuges are part of larger designated RCW population units (e.g., primary or secondary core), collaborative monitoring with partners using this same protocol framework is encouraged.

```
pull_rcw_cluster_survey ← function() {

  # Do some stuff ...

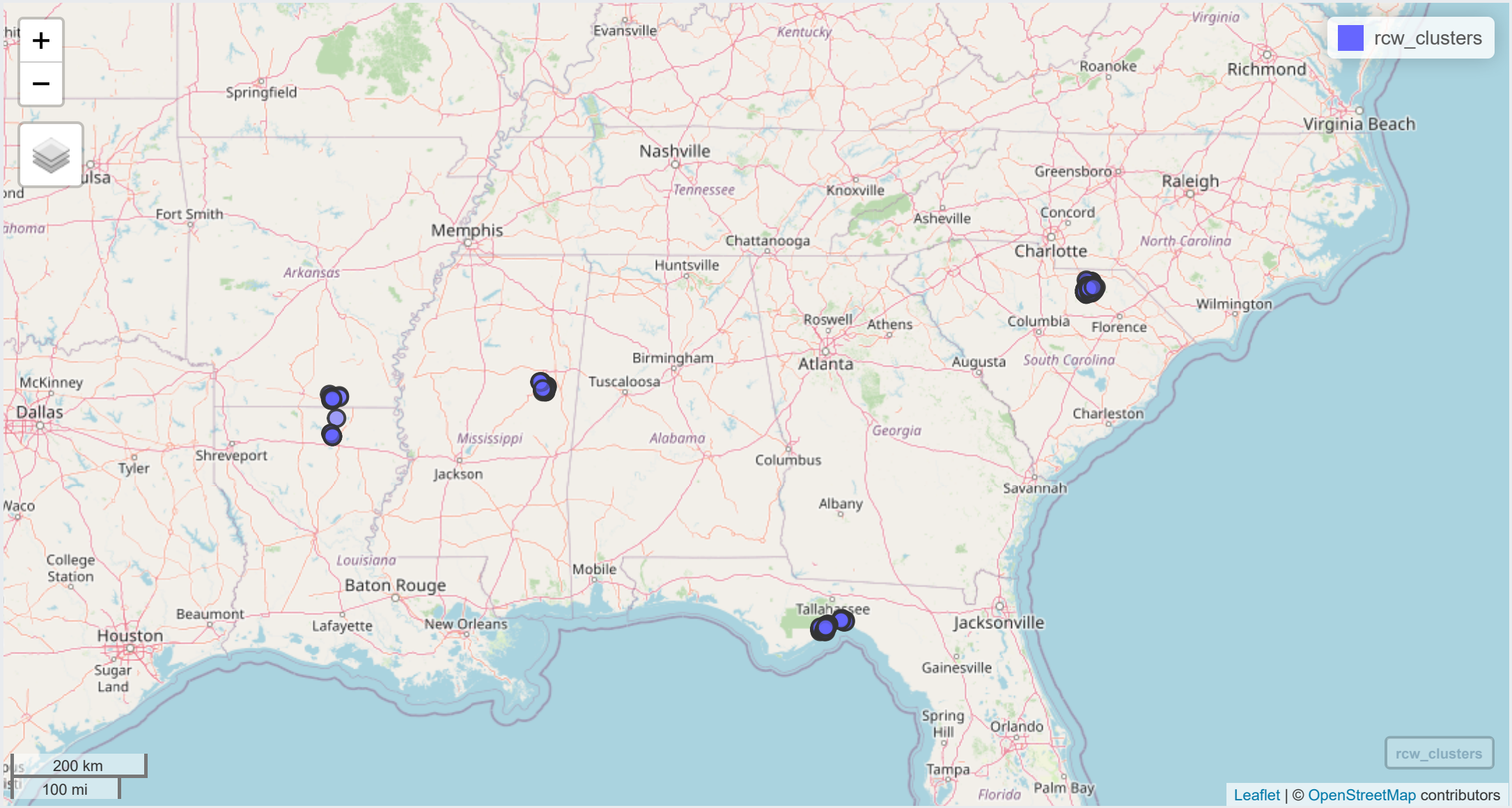
  # Connect and pull content of RCW cluster status feature on USFWS AGOL
  url ← "https://services.arcgis.com/ ... path_RCW_cluster_feature_server"
  token ← "wouldnt_you_like_to_know"
  outFields ← "*"
  where = "1=1"
  layerInfo ← jsonlite::fromJSON(
    httr::content(httr::POST(url,
                             query = list(f = "json", token = token),
                             encode = "form",
                             config = httr::config(ssl_verifypeer = FALSE)),
                  as = "text"))

  # Do some more stuff ...

}
```

```
rcw_clusters ← pull_rcw_cluster_survey()
```

objectid	lit	FK_obs1	FK_obsOther	FK_obs2	cluster_date	clusterID	cluster_status	cluster_n
1	NXB	Steven Lewis	NA	NA	1579046400	118	AC	NA
2	NXB	Steven Lewis	NA	NA	1580191200	103	AC	FALSE
3	NXB	Steven Lewis	NA	NA	1580169600	94	AC	FALSE
4	NXB	Steven Lewis	NA	NA	1579046400	114	AC	NA
5	NXB	Steven Lewis	NA	NA	1578528000	13	AC	NA


rcw_clusters
Evansville
Kentucky
Virginia
Roanoke
Richmond
Springfield
Virginia Beach
Nashville
Greensboro
Raleigh
Tennessee
Knoxville
Fort Smith
Asheville
Concord
Memphis
Chattanooga
Charlotte
North Carolina
Arkansas
Huntsville
Wilmington
Roswell
Athens
Columbia
Florence
Birmingham
Atlanta
Augusta
South Carolina
McKinney
Tuscaloosa
Dallas
Charleston
Mississippi
Alabama
Georgia
Tyler
Shreveport
Jackson
Columbus
Savannah
Albany
College Station
Louisiana
Mobile
Baton Rouge
Tallahassee
Jacksonville
Beaumont
Lafayette
New Orleans
Houston
Gainesville
Sugar Land
Spring Hill
Orlando
Tampa
Palm Bay
Florida
200 km
100 mi
rcw_clusters
Leaflet | © OpenStreetMap contributors

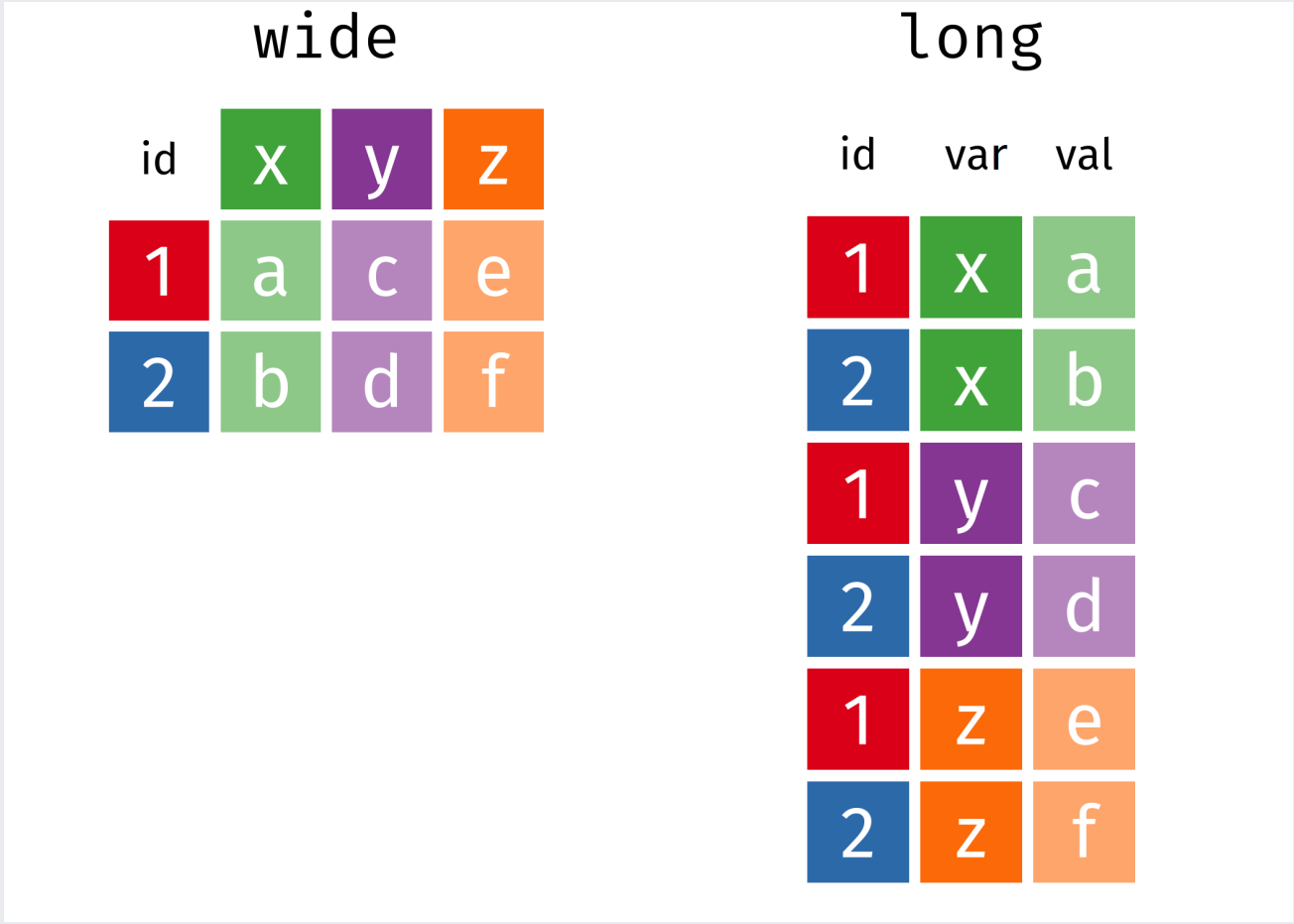
Process

Data Management Life Cycle*													
Data Management Planning							Data Life Cycle						
People	Data	Standards	Process	QA/QC	Storage	Distribution	Acquire	Quality Control	Process	Analyze	Archive	Publish	Share
Assign data management roles & responsibilities	Identify types of data to be collected or created	Determine metadata standards to be used	Determine how data will be manipulated	Determine data & metadata QA & QC procedures	Determine where data will be stored	Determine how will data be shared	Collect or generate data	Check data for errors	Manipulate data for analysis or final archiving	Produce results & products to meet objectives	Submit data & metadata to data repository	Publish metadata records to data catalog	Share data with internal & external users
Document data and data management strategy – data management plan, metadata													

Data processing

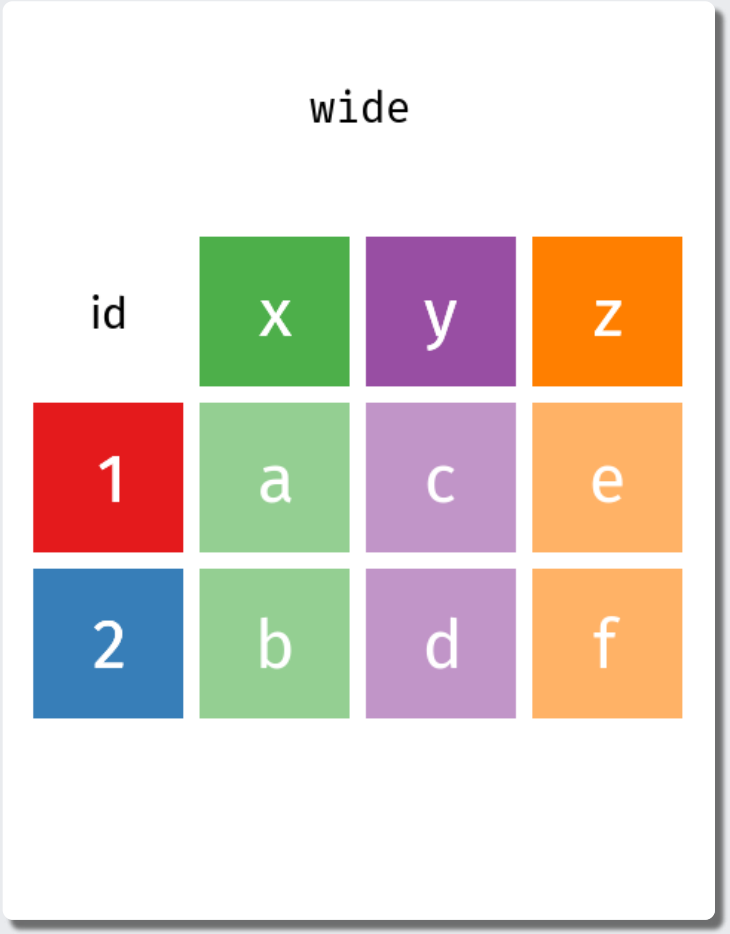
- Reshaping
- Manipulating, transforming, cleaning
- Quality control
- Exploratory data analysis

Data reshaping



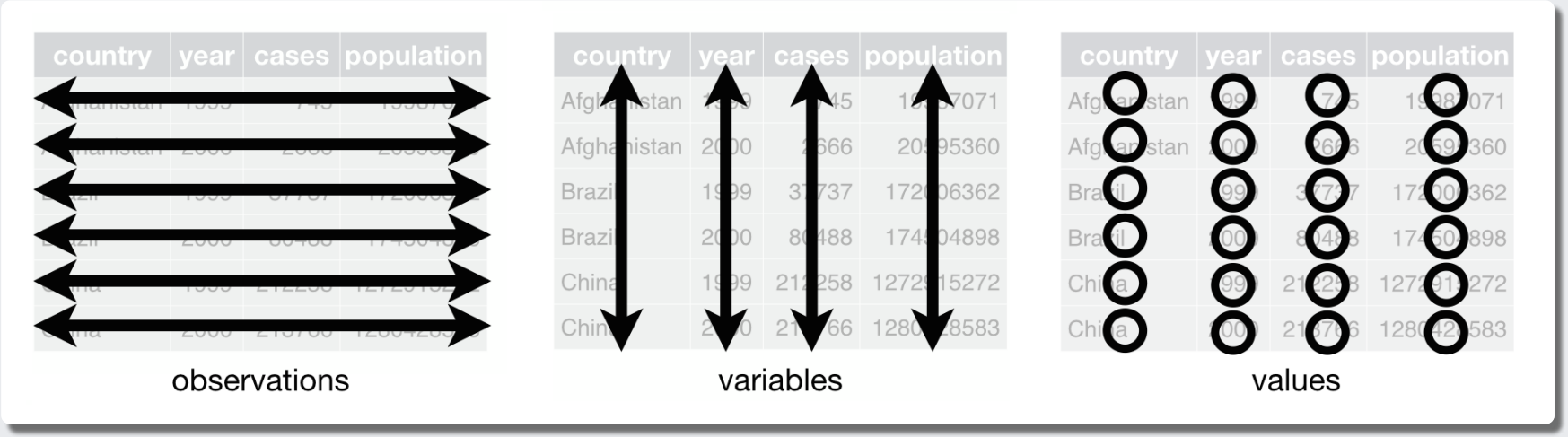
Moving from wide to long

- `tidyr::pivot_*` functions (example)
- `data.table`



"Tidy" data

- Not "clean", but organized such that:
  - each observation gets its own row
  - each variable gets its own column
  - each value gets its own cell



Data manipulation and wrangling

- Transform variables
- Create new variables
- Filter
- Sort
- Group and summarize
- Wrangle (clean)
  - factors
  - strings
  - dates
  - times

Data manipulation/wrangling R packages

Tidyverse

- `tidyr` (pivoting, split/unite)
- `dplyr` (data manipulation)
- `lubridate` (dates, times)
- `forcats` (factors)
- `stringr` (strings)
- facilitates "piped" workflow

Alternative API

- `data.table` (pivoting, manipulation)

Piped workflow

- Pipe operator: %>%

Piped

```
raw_data ← rio::import("survey_data.xlsx")
mod_data ← raw_data %>%
  dplyr::mutate(new_var = var_one + var_two) %>%
  dplyr::filter(new_var < 100) %>%
  dplyr::arrange(sort_var)
```

Quality control

- Passive: **data profiling**
- Active: **assertive programming**

Small mammal trapping data set from Portal Project Teaching Database

```
mammals ← rio::import("https://ndownloader.figshare.com/files/10717177")
```

Data profiling functions

`str` & `summary` (base R)

```
str(mammals)
```

```
## 'data.frame':    2943 obs. of  9 variables:
##  $ record_id      : int  1364 1736 28719 18080 34899 13650 4908 26726 25205 29192 ...
##  $ month          : int  10 4 10 12 10 11 10 7 3 2 ...
##  $ day            : int  8 28 24 15 6 21 24 30 15 20 ...
##  $ year           : int  1978 1979 1998 1990 2002 1987 1981 1997 1997 1999 ...
##  $ plot_id        : int  14 22 22 24 6 6 10 8 1 19 ...
##  $ species_id     : chr  "DS" "DS" "DM" "RM" ...
##  $ sex            : chr  "F" "M" "F" "F" ...
##  $ hindfoot_length: int  50 50 36 17 26 35 NA 36 35 25 ...
##  $ weight         : int  121 76 48 14 31 51 NA 44 54 31 ...
```

```
summary(mammals)
```

```
##    record_id         month              day             year         plot_id
##  Min.   :   15   Min.   : 1.000   Min.   : 1.0   Min.   :1977   Min.   : 1.00
##  1st Qu.: 8630   1st Qu.: 4.000   1st Qu.: 9.0   1st Qu.:1983   1st Qu.: 5.00
##  Median :17929   Median : 7.000   Median :15.0   Median :1990   Median :11.00
##  Mean   :17754   Mean   : 6.498   Mean   :15.9   Mean   :1990   Mean   :11.49
##  3rd Qu.:26794   3rd Qu.: 9.000   3rd Qu.:23.0   3rd Qu.:1997   3rd Qu.:17.00
##  Max.   :35541   Max.   :12.000   Max.   :31.0   Max.   :2002   Max.   :24.00
##
##   species_id           sex            hindfoot_length     weight
##  Length:2943        Length:2943        Min.   :11.0    Min.   :  4.0
##  Class :character   Class :character   1st Qu.:21.0    1st Qu.: 20.0
##  Mode  :character   Mode  :character   Median :31.0    Median : 36.0
##                                        Mean   :29.4    Mean   : 43.4
##                                        3rd Qu.:36.0    3rd Qu.: 48.0
##                                        Max.   :70.0    Max.   :280.0
##                                        NA's   :348     NA's   :274
```

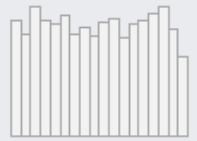
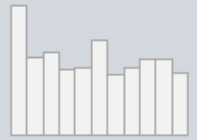
summarytools::dfSummary

```
summarytools::view(summarytools::dfSummary(mammals))
```

Data Frame Summary

mammals

Dimensions: 2943 x 9
Duplicates: 0

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Missing
1	record_id [integer]	Mean (sd) : 17754 (10328.2) min < med < max: 15 < 17929 < 35541 IQR (CV) : 18163.5 (0.6)	2943 distinct values		0 (0%)
2	month [integer]	Mean (sd) : 6.5 (3.4) min < med < max: 1 < 7 < 12 IQR (CV) : 5 (0.5)	12 distinct values		0 (0%)

skimr::skim

```
skimr::skim(mammals)
```

Data summary

Name	mammals
Number of rows	2943
Number of columns	9
_____	
Column type frequency:	
character	2
numeric	7
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
species_id	63	0.98	2	2	0	30	0

DataExplorer::create_report

```
# An .html file is output to your current working directory and opens in browser
DataExplorer::create_report(mammals,
                            output_file = "mammal_data_profile.html")
```

Basic Statistics

Raw Counts

Name	Value
Rows	2,943
Columns	9
Discrete columns	2
Continuous columns	7
All missing columns	0
Missing observations	889
Complete Rows	2,527
Total observations	26,487
Memory allocation	131 Kb

Percentages

Assertive programming

- `assertr` (vignette)

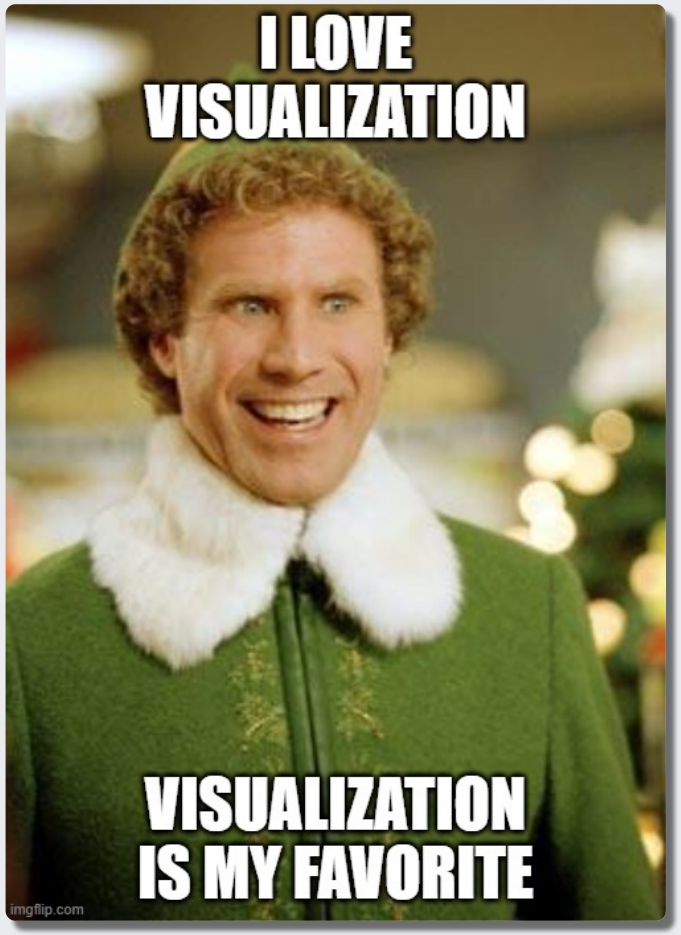
```
# Expect maximum weight
mammals %>%
  assertr::verify(weight < 100)

# Alternative, giving weight range
mammals %>%
  assertr::assert(assertr::within_bounds(15, 100), weight)

# Confirm expected format of sex variable
mammals %>%
  assertr::assert(assertr::in_set(c("M", "F")), sex)
```

Exploratory data analysis

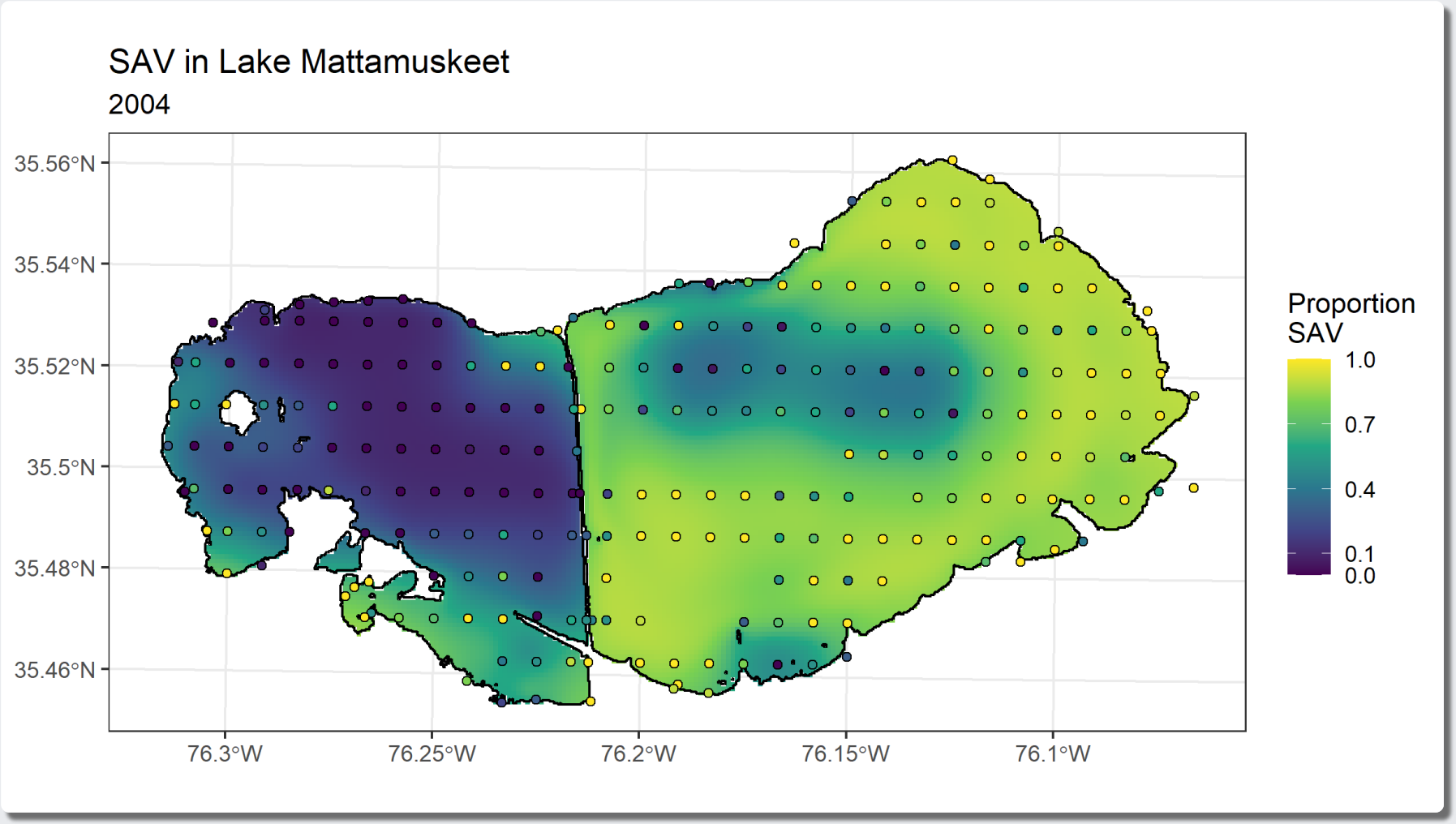
- Aims for understanding
  - variation, missing values, covariation
- Iterative (questions -> answers -> questions)
- Visualization is key

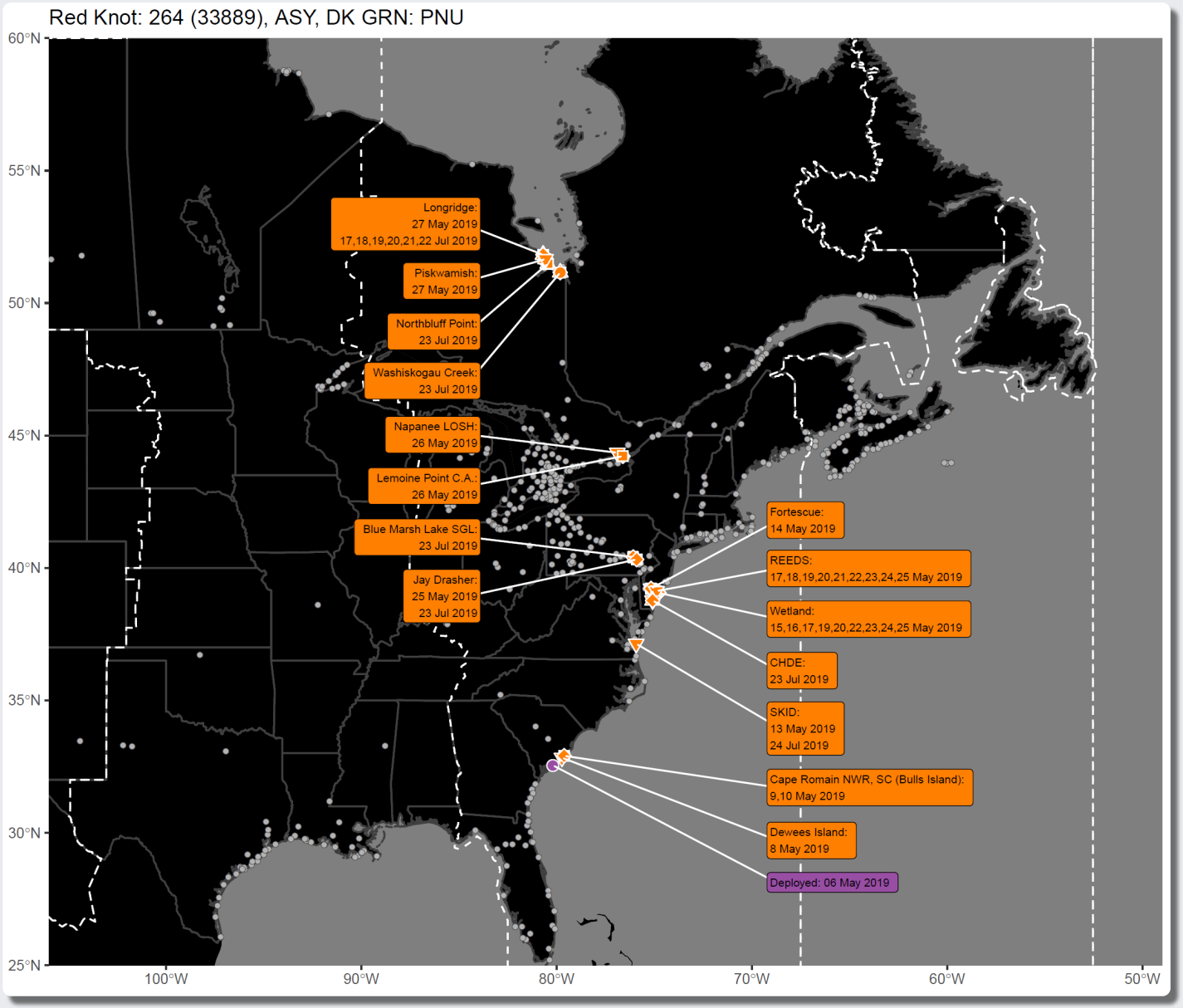


Visualization framework

- `ggplot2`
  - "grammar of graphics"
  - flexible, consistent

Target: *Submerged aquatic vegetation - Mattamuskeet NWR*




Red Knot: 264 (33889), ASY, DK GRN: PNU
60°N
55°N
50°N
45°N
40°N
35°N
30°N
25°N
100°W
90°W
80°W
70°W
60°W
50°W
Longridge:
27 May 2019
17,18,19,20,21,22 Jul 2019
Piskwamish:
27 May 2019
Northbluff Point:
23 Jul 2019
Washiskogau Creek:
23 Jul 2019
Napanee LOSH:
26 May 2019
Lemoine Point C.A.:
26 May 2019
Blue Marsh Lake SGL:
23 Jul 2019
Jay Drasher:
25 May 2019
23 Jul 2019
Fortescue:
14 May 2019
REEDS:
17,18,19,20,21,22,23,24,25 May 2019
Wetland:
15,16,17,19,20,22,23,24,25 May 2019
CHDE:
23 Jul 2019
SKID:
13 May 2019
24 Jul 2019
Cape Romain NWR, SC (Bulls Island):
9,10 May 2019
Dewees Island:
8 May 2019
Deployed: 06 May 2019

Analyze

Data Management Life Cycle*													
Data Management Planning							Data Life Cycle						
People	Data	Standards	Process	QA/QC	Storage	Distribution	Acquire	Quality Control	Process	Analyze	Archive	Publish	Share
Assign data management roles & responsibilities	Identify types of data to be collected or created	Determine metadata standards to be used	Determine how data will be manipulated	Determine data & metadata QA & QC procedures	Determine where data will be stored	Determine how will data be shared	Collect or generate data	Check data for errors	Manipulate data for analysis or final archiving	Produce results & products to meet objectives	Submit data & metadata to data repository	Publish metadata records to data catalog	Share data with internal & external users
Document data and data management strategy – data management plan, metadata													

CRAN task views

- relevant R packages for ~ 40 topics
- emphasize packages, not methodologies
- not endorsements

Diverse topics

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Databases	Databases with R
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data

When all else fails...

Sharing and Archiving

Data Management Life Cycle*													
Data Management Planning							Data Life Cycle						
People	Data	Standards	Process	QA/QC	Storage	Distribution	Acquire	Quality Control	Process	Analyze	Archive	Publish	Share
Assign data management roles & responsibilities	Identify types of data to be collected or created	Determine metadata standards to be used	Determine how data will be manipulated	Determine data & metadata QA & QC procedures	Determine where data will be stored	Determine how will data be shared	Collect or generate data	Check data for errors	Manipulate data for analysis or final archiving	Produce results & products to meet objectives	Submit data & metadata to data repository	Publish metadata records to data catalog	Share data with internal & external users
Document data and data management strategy – data management plan, metadata													


TELL ME MORE ABOUT WRITING REPORTS
REPORT ABOUT REPORTING
memegenerator.net

Literate programming

You can generate reports *directly from R!*

Allows for:

- Code to be directly inserted into document
- Multiple outputs
- Easy to use
- Increased reproducibility
- Fewer sources of errors


Rmarkdown
TEXT. CODE. OUTPUT.
(GET IT TOGETHER, PEOPLE.)
#HEADERS
CODE
TEXT
PANDOC
KNITR
FAIRY DUST
RECIPE
1. ADD TEXT
2. ADD CODE
3. KNIT
4. (magic)
5. CELEBRATE PERCEIVED WIZARDRY
rmarkdown
knitr
pandoc
.Rmd
.md

The YAML header

```
---
title: "My report"
author: "McCrea Cobb"
date: "6/24/2020"
output: pdf_document
---
```

A more complex header

```
---
title: |
    | {width=5cm}
    |
    | \LARGE Region 4 Inventory and Monitoring Branch
subtitle: |
    | \Large Mobile Acoustical Bat Monitoring
    | \Large Annual Summary Report
author: 'r <span style='background-color:#ffff7f'>params</span>$year'
date: 'r <span style='background-color:#ffff7f'>params</span>$station'
output:
  pdf_document:
    includes:
      in_header: MABM_report_preamble.tex
urlcolor: blue
params:
  year: 0 # placeholder
  station: placeholder
  stn_start_yr: 0 # placeholder
  route_path: placeholder
  survey_path: placeholder
  bat_path: placeholder
  spp_path: placeholder
  out_dir: placeholder
  goog_API_key: placeholder
---
```

Some Rmarkdown code

This is an **R Markdown** document. You can write inline code like this: 1 + 1 = `r 1 + 1` .

You can embed an R code chunk like this:

````
```{r eval=TRUE, echo=FALSE}
DT::datatable(iris[1:3],
              rownames = FALSE,
              options = list(pageLength = 3))
```
````

Including Figures

You can also embed figures, for example:

````
```{r, eval=TRUE, echo=FALSE}
library(leaflet)

leaflet() %>%
  addTiles() %>%
  addMarkers(lng = 174.768,
             lat = -36.852,
             popup = "The birthplace of R")
```
````

Some Rmarkdown code

This is an **R Markdown** document. You can write inline code like this: 1 + 1 = `r 1 + 1` .

You can embed an R code chunk like this:

```
```{r eval=TRUE, echo=FALSE}
DT::datatable(iris[1:3],
              rownames = FALSE,
              options = list(pageLength = 3))
```
```

Including Figures

You can also embed figures, for example:

```
```{r, eval=TRUE, echo=FALSE}
library(leaflet)

leaflet() %>%
  addTiles() %>%
  addMarkers(lng = 174.768,
             lat = -36.852,
             popup = "The birthplace of R")
```
```

Some Rmarkdown code

This is an **R Markdown** document. You can write inline code like this: 1 + 1 = 2.

You can embed an R code chunk like this:

Show 3 entries Search:

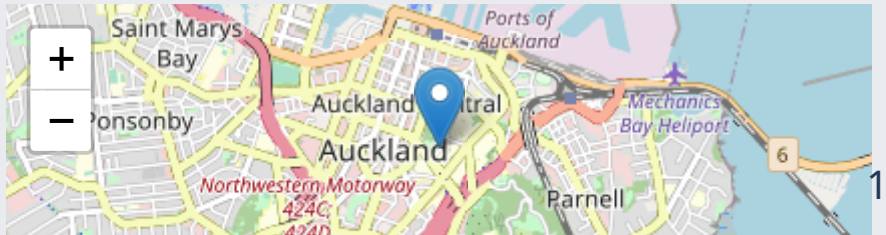
Sepal.Length	Sepal.Width	Petal.Length
5.1	3.5	1.4
4.9	3	1.4
4.7	3.2	1.3

Showing 1 to 3 of 150 entries

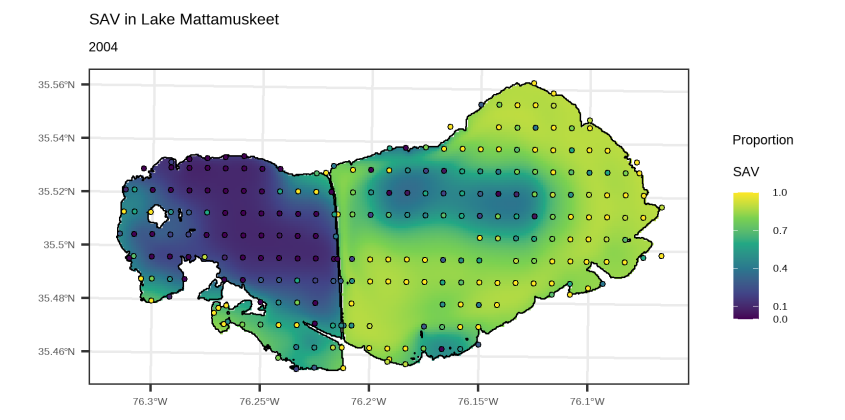
Previous 1 2 3 4 5 ... 50 Next

Including Figures

You can also embed figures, for example:



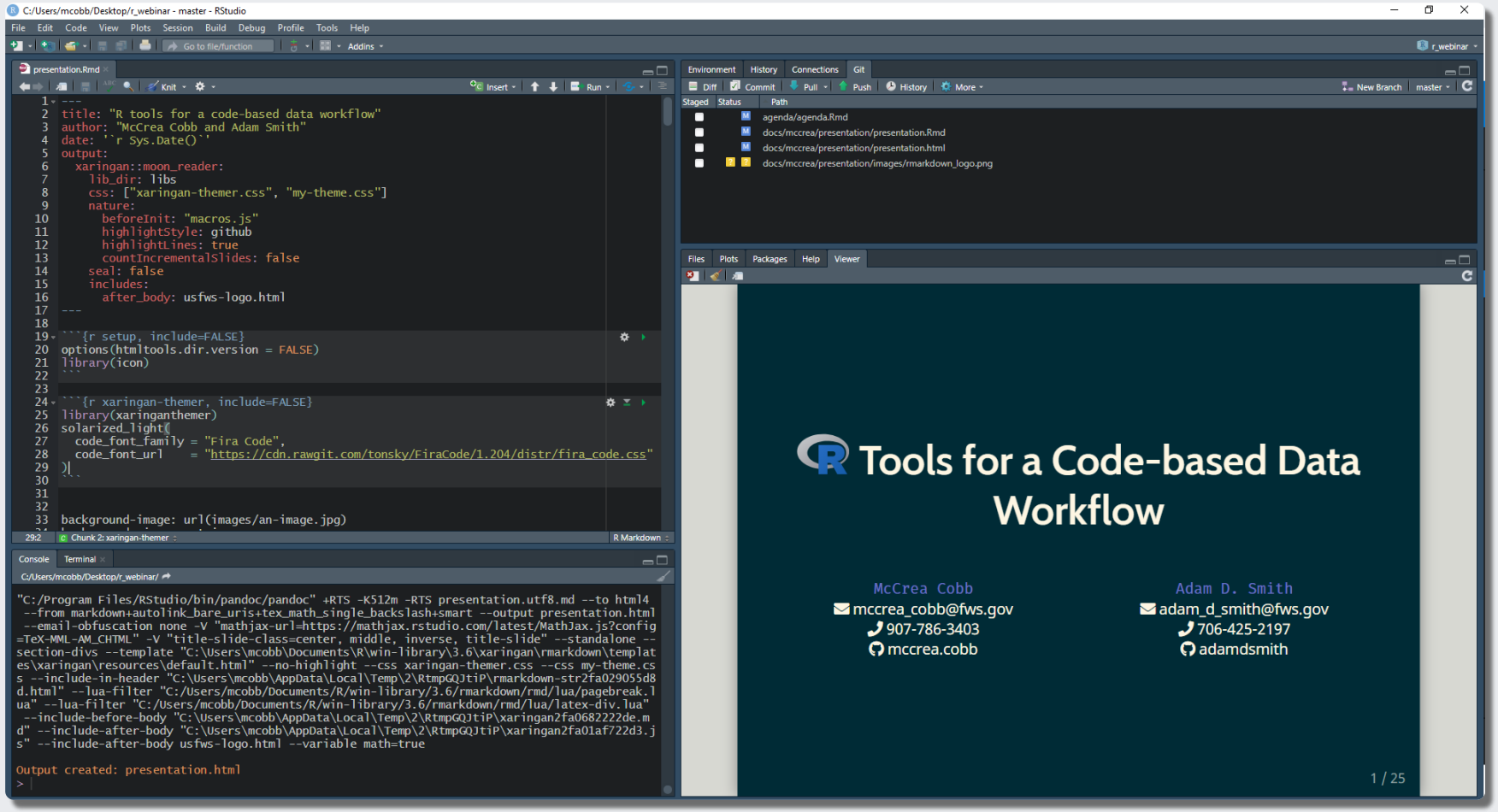
```
ggplot(data = sav_model_fit) +
  geom_raster(aes(x = x, y = y, fill = sav)) +
  geom_sf(data = LM, color = "black", fill = NA) +
  geom_point(data = sav_pts, shape = 21,
             aes(x, y, fill = sav)) +
  scale_fill_viridis_c("Proportion\nSAV",
                       breaks = c(0,0.1,0.4,0.7,1),
                       limits = c(0,1)) +
  coord_sf() +
  labs(title = "SAV in Lake Mattamuskeet",
       subtitle = "2004",
       x = NULL, y = NULL) +
  theme_bw()
```



ggplot2 workshop part 1

ggplot2 workshop part 2

Presentations



Web applications

- Build interactive web apps from R
- Host standalone apps online
- Embed them in RMarkdown docs

Animal tracking example

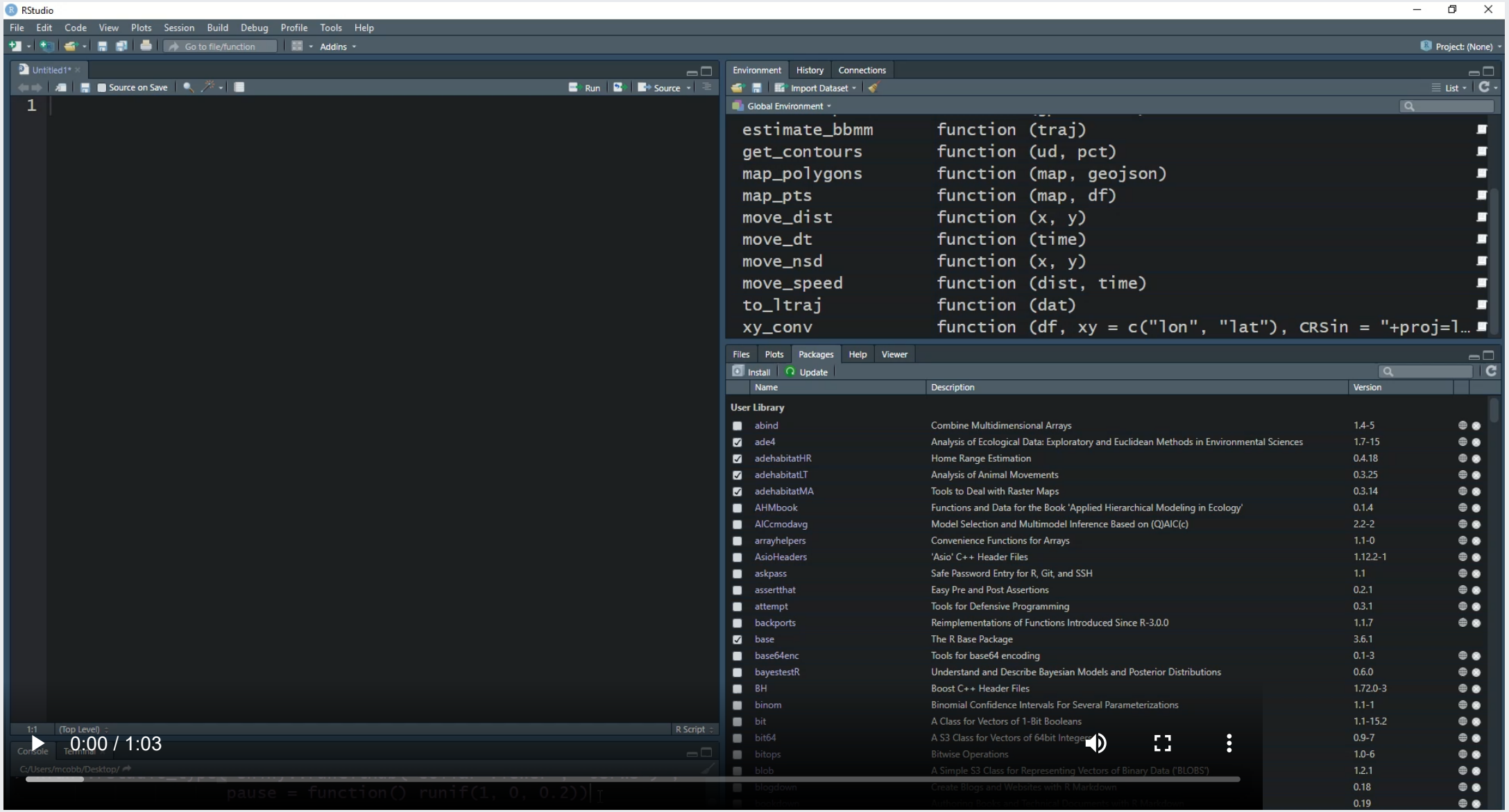
CRAN Task View: Processing and Analysis of Tracking Data

Maintainer: Rocío Joo, Matthew E. Boone, Michael Sumner and Mathieu Basille
Contact: rocio.joo at ufl.edu
Version: 2020-06-29
URL: https://CRAN.R-project.org/view=Tracking

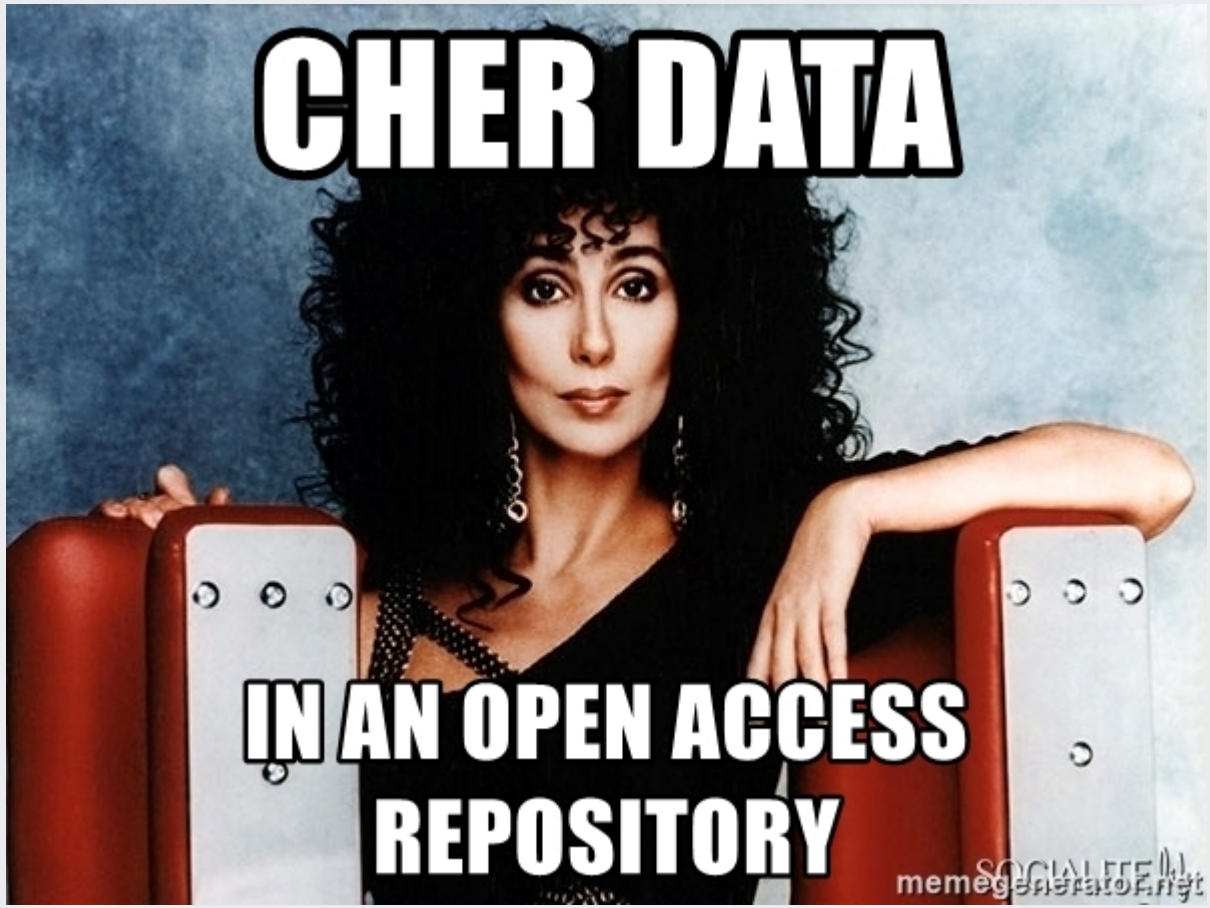
This CRAN Task View (CTV) contains a list of packages useful for the processing and analysis of tracking data. Besides the maintainers, the following people contributed to the creation of this task view: Achim Zeileis, Edzer Pebesma.

Movement of an object (both living organisms and inanimate objects) is defined as a change in its geographic location in time, so movement data can be defined by a space and a time component. Tracking data are composed by at least 2-dimensional spatial coordinates (x,y) and a time index (t), and can be seen as the geometric representation (the trajectory) of an object's path. The packages listed here, henceforth called **tracking packages** , are those explicitly developed to either create, transform or analyze tracking data (i.e. (x,y,t)), allowing a full workflow from raw data from tracking devices to final analytical outcome. For instance, a package that would use accelerometer, gyroscope and magnetometer data to reconstruct an objects's trajectory—most likely an animal's trajectory—via dead-reckoning, thus transforming those data into an (x,y,t) format, would fit into the definition. However, a package analyzing accelerometry series to detect changes in behavior would not fit. See more on this in Joo *et al.* (2019) . Regarding (x,y), some packages may assume 2-D Euclidean (Cartesian) coordinates, and others may assume geographic (longitude/latitude) coordinates. We encourage the users to verify how coordinates are processed in the packages, as the consequences can be important in terms of spatial attributes (e.g. distance, speed and angles).

The packages included here are mainly tracking packages though we include a subsection of other movement-related packages. The packages are mainly from CRAN


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1
Environment History Connections
Import Dataset
List
Global Environment
estimate_bbmm function (traj)
get_contours function (ud, pct)
map_polygons function (map, geojson)
map_pts function (map, df)
move_dist function (x, y)
move_dt function (time)
move_nsd function (x, y)
move_speed function (dist, time)
to_ltraj function (dat)
xy_conv function (df, xy = c("lon", "lat"), CRSin = "+proj=l…
Files Plots Packages Help Viewer
Install
Update
Name
Description
Version
User Library
abind Combine Multidimensional Arrays 1.4-5
ade4 Analysis of Ecological Data: Exploratory and Euclidean Methods in Environmental Sciences 1.7-15
adehabitatHR Home Range Estimation 0.4.18
adehabitatLT Analysis of Animal Movements 0.3.25
adehabitatMA Tools to Deal with Raster Maps 0.3.14
AHMbook Functions and Data for the Book 'Applied Hierarchical Modeling in Ecology' 0.1.4
AICcmodavg Model Selection and Multimodel Inference Based on (Q)AIC(c) 2.2-2
arrayhelpers Convenience Functions for Arrays 1.1-0
AsioHeaders 'Asio' C++ Header Files 1.12.2-1
askpass Safe Password Entry for R, Git, and SSH 1.1
assertthat Easy Pre and Post Assertions 0.2.1
attempt Tools for Defensive Programming 0.3.1
backports Reimplementations of Functions Introduced Since R-3.0.0 1.1.7
base The R Base Package 3.6.1
base64enc Tools for base64 encoding 0.1-3
bayestestR Understand and Describe Bayesian Models and Posterior Distributions 0.6.0
BH Boost C++ Header Files 1.72.0-3
binom Binomial Confidence Intervals For Several Parameterizations 1.1-1
bit A Class for Vectors of 1-Bit Booleans 1.1-15.2
bit64 A S3 Class for Vectors of 64bit Integers 0.9-7
bitops Bitwise Operations 1.0-6
blob A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS') 1.2.1
blogdown Create Blogs and Websites with R Markdown 0.18
0.19
1:1 (Top Level) R Script
Console Terminal
C:/Users/mcobb/Desktop/
0:00 / 1:03

Archiving



Archiving

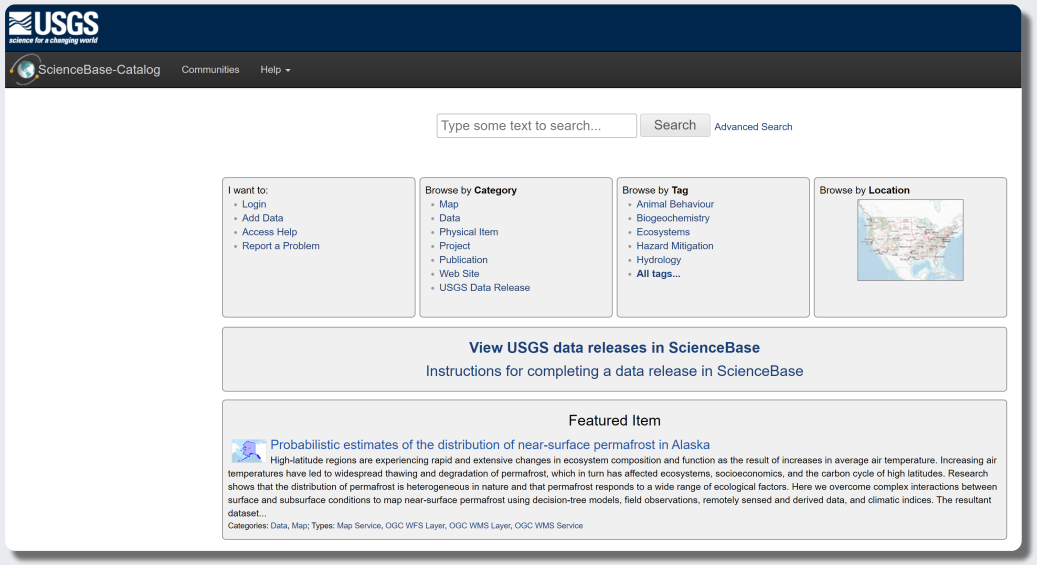

We Can Do It!
(in R)
POST FEB. 15 TO FEB. 28
WAR PRODUCTION CO-ORDINATING COMMITTEE


zenodo


figshare

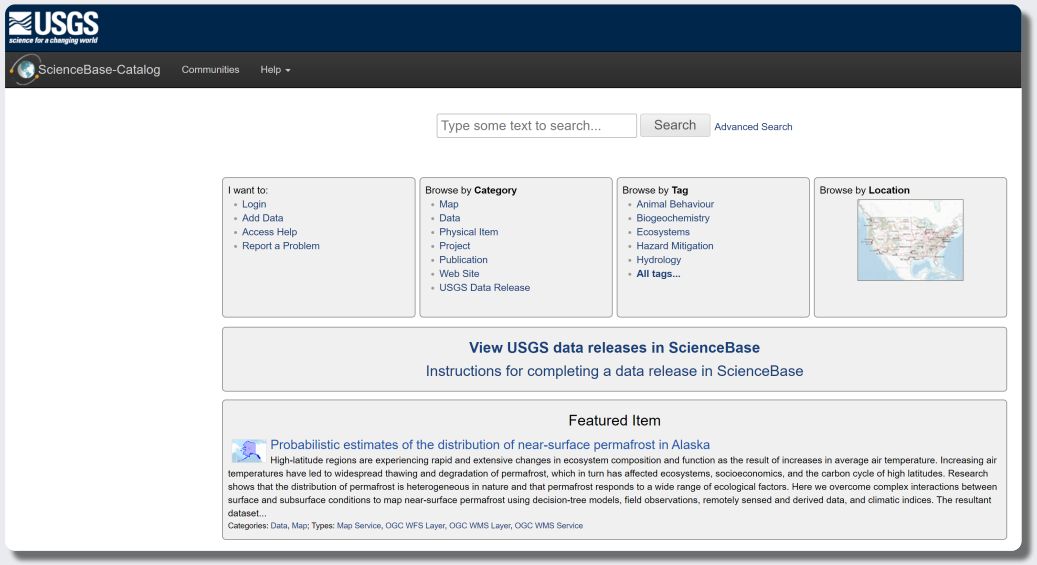
sbtools

- Allows complete access to the USGS ScienceBase API from R.
- Supports the creation, editing and access of data/metadata:



sbtools

- Allows complete access to the USGS ScienceBase API from R.
- Supports the creation, editing and access of data/metadata:



Create a record:

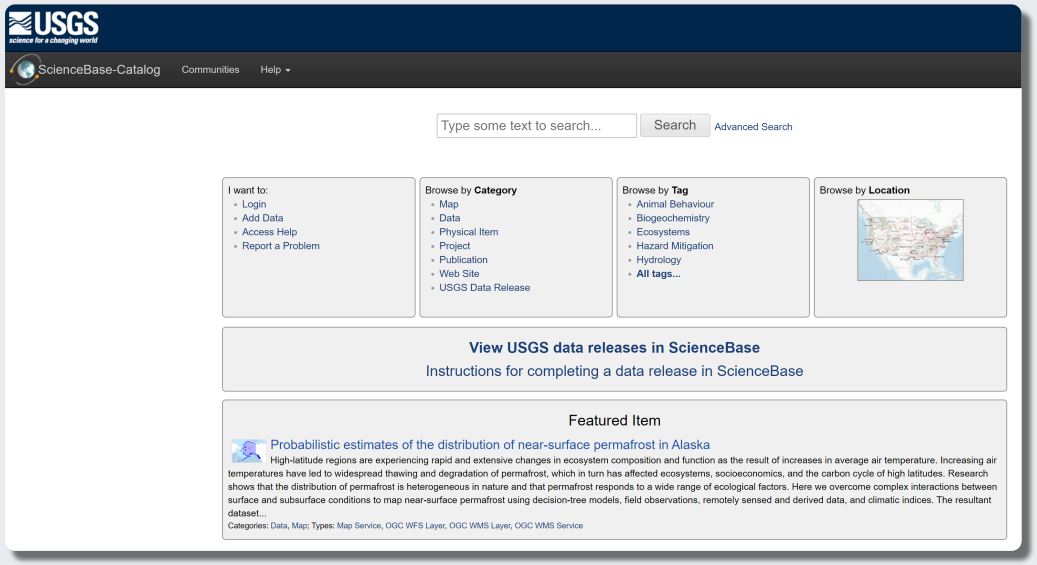
```
library(sbtools)

# Log in
authenticate_sb(username = "username@usgs.gov",
                password = password)

# Create new item (record), by default under "My
Items" parent
new_item ← item_create(title = "new test item")
```

sbtools

- Allows complete access to the USGS ScienceBase API from R.
- Supports the creation, editing and access of data/metadata:



Create a record:

```
library(sbtools)

# Log in
authenticate_sb(username = "username@usgs.gov",
                password = password)

# Create new item (record), by default under "My
Items" parent
new_item ← item_create(title = "new test item")
```

Edit a record:

```
# Give the item a new title
edited_item ←  item_update(new_item, list(title =
"new updated item"))

# Append data to the item
item_append_files(edited_item, "test.dat")
```

sbtools: Accessing data

```
# Access the record
test_item ← item_get("572a2a7fe4b0b13d391a0f6c")

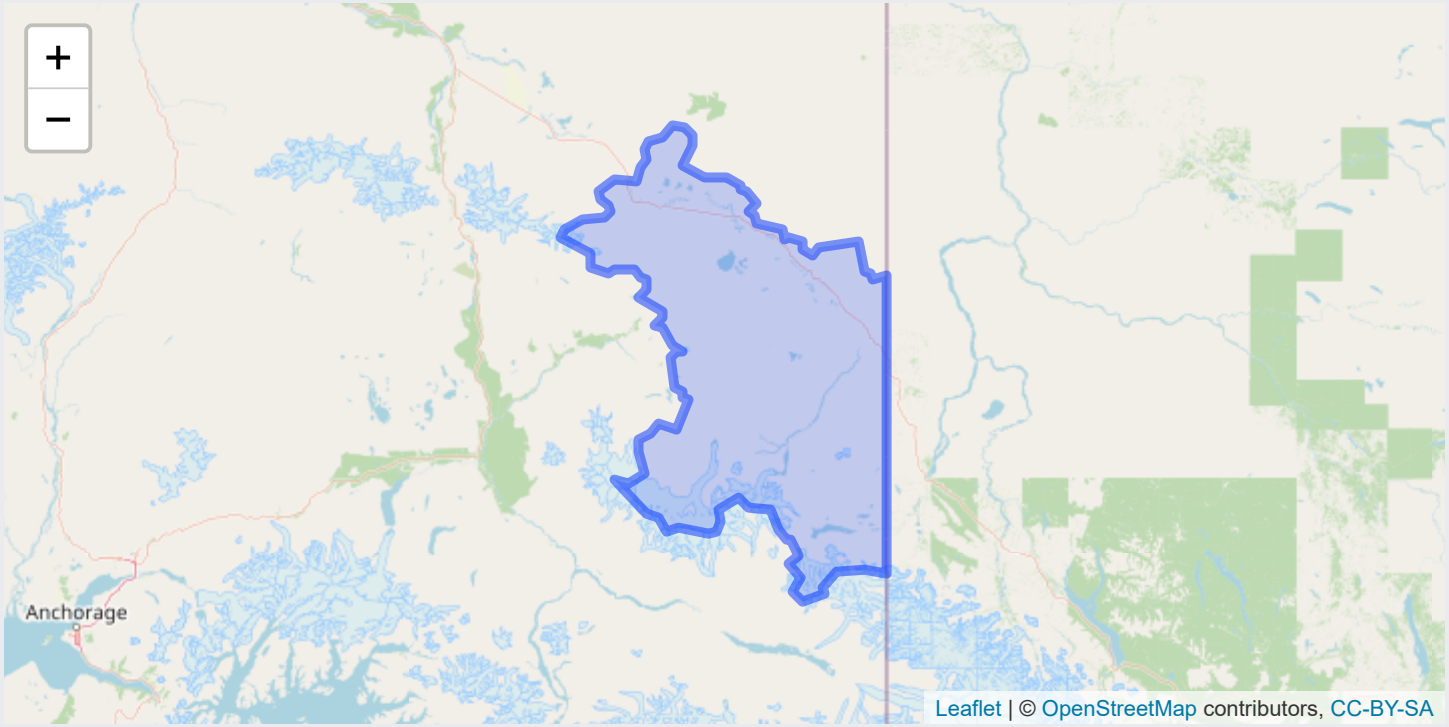
# Take a look at the citation
test_item
```

```
<ScienceBase Item>
  Title: Moose population survey: Tetlin National Wildlife Refuge, Game Management Unit 12,
   eastern Alaska
  Creator/LastUpdatedBy:      /
  Provenance (Created / Updated):  2016-05-04T16:59:43Z / 2016-11-29T21:04:15Z
  Children: FALSE
  Item ID: 572a2a7fe4b0b13d391a0f6c
  Parent ID: 5771b40fe4b07657d1a6bb5e
```

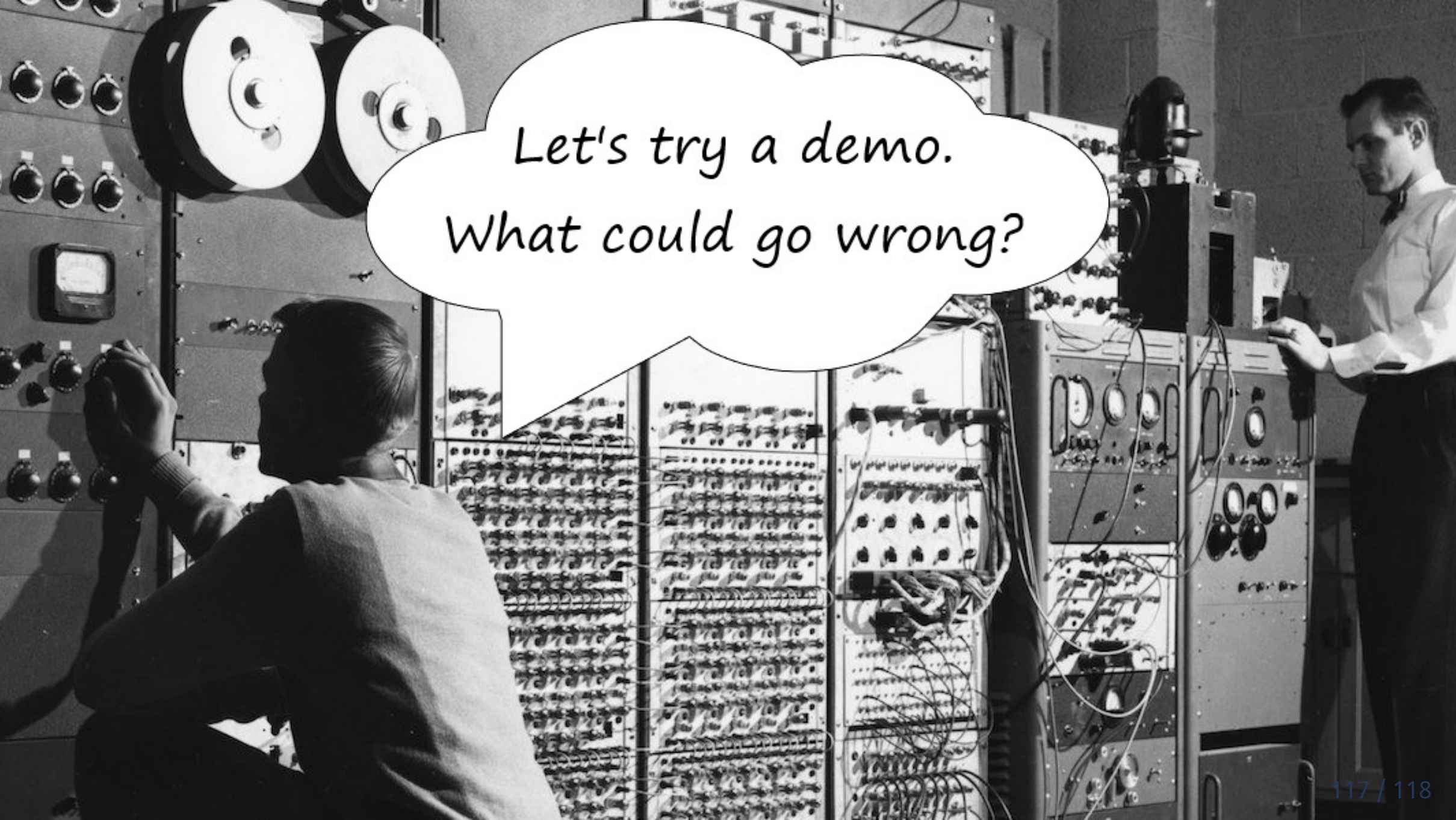
```
library(leaflet)

# Get study area boundaries
study_area ← item_get_wfs(test_item)

# Map it
leaflet() %>%
  addPolygons(data = study_area@polygons[[1]],
              label = "Game Management Unit 12") %>%
  addTiles()
```

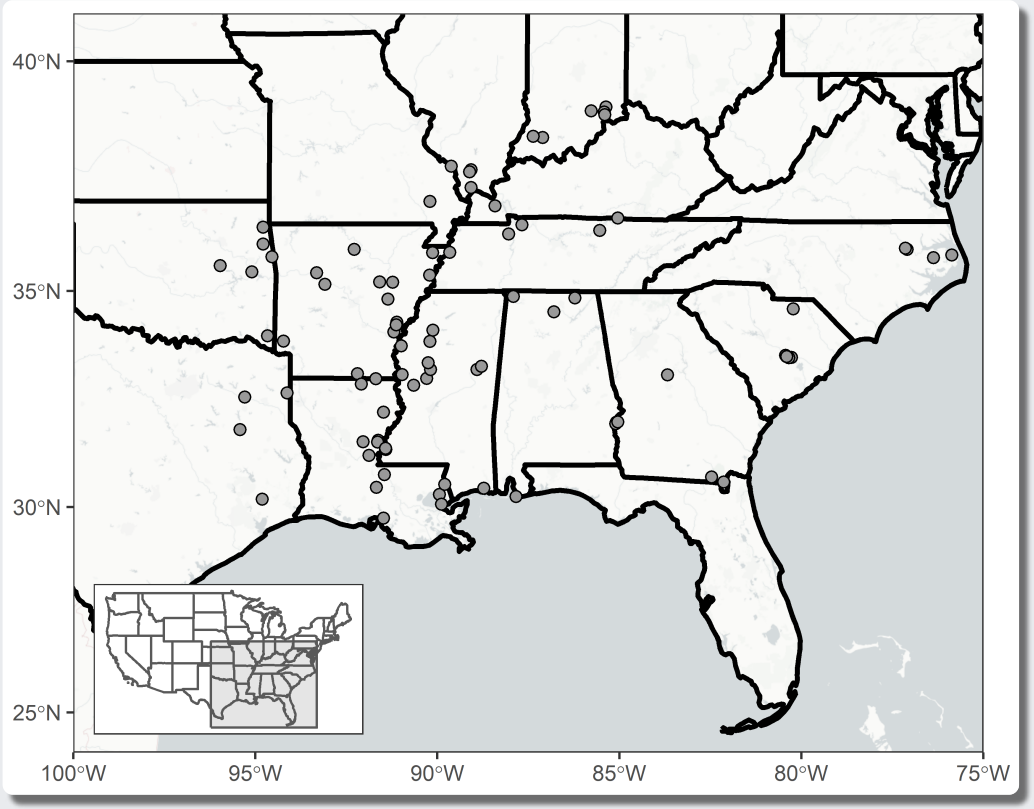


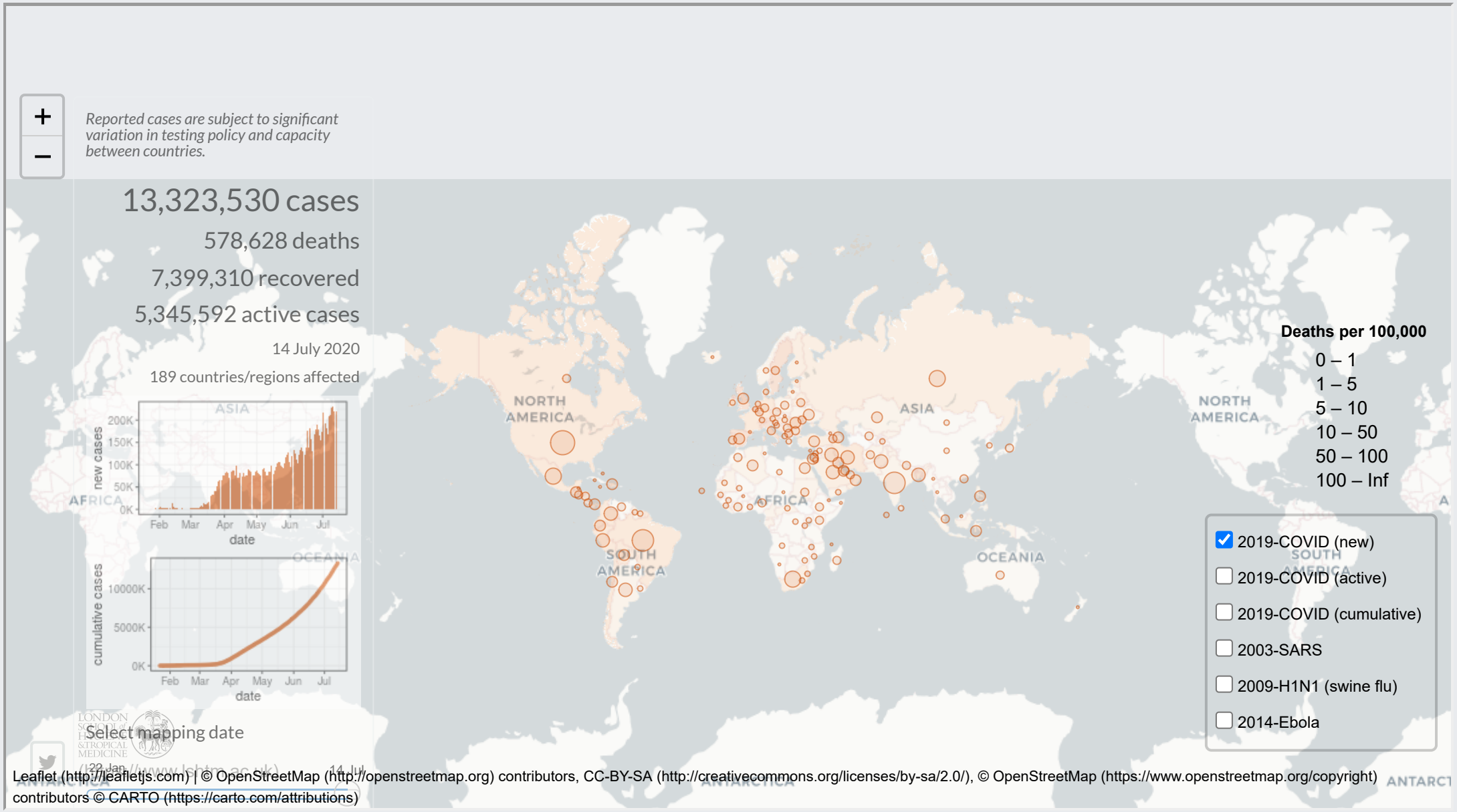
github.com/USFWS/data-mgt-with-r


Let's try a demo.
What could go wrong?

Case study: Mobile Acoustical Bat Monitoring

- 86 survey routes at 63 FWS field stations
  - ~ 40 participate annually
- Data life cycle bottlenecks
  1. process raw identification and GPS data for database import
  2. generate annual reports for each station




+
−
Reported cases are subject to significant variation in testing policy and capacity between countries.
13,323,530 cases
578,628 deaths
7,399,310 recovered
5,345,592 active cases
14 July 2020
189 countries/regions affected
ASIA
new cases
200K
150K
100K
50K
0K
Feb Mar Apr May Jun Jul
date
OCEANIA
cumulative cases
10000K
5000K
0K
Feb Mar Apr May Jun Jul
date
Select mapping date
22 Jan
14 Jul
NORTH AMERICA
SOUTH AMERICA
AFRICA
ASIA
OCEANIA
ANTARCTICA
Deaths per 100,000
0 – 1
1 – 5
5 – 10
10 – 50
50 – 100
100 – Inf
2019-COVID (new)
2019-COVID (active)
2019-COVID (cumulative)
2003-SARS
2009-H1N1 (swine flu)
2014-Ebola
Leaflet (http://leafletjs.com) | © OpenStreetMap (http://openstreetmap.org) contributors, CC-BY-SA (http://creativecommons.org/licenses/by-sa/2.0/), © OpenStreetMap (https://www.openstreetmap.org/copyright) contributors © CARTO (https://carto.com/attributions)

Accessing web services in R

CRAN Task View: Web Technologies and Services

Maintainer: Scott Chamberlain, Thomas Leeper, Patrick Mair, Karthik Ram, Christopher Gandrud
Contact: myrmecocystus at gmail.com
Version: 2020-06-08
URL: https://CRAN.R-project.org/view=WebTechnologies

This Task View contains information about to use R and the world wide web together. The base version of R does not ship with many tools for interacting with the web. Thankfully, there are an increasingly large number of tools for interacting with the web. This task view focuses on packages for obtaining web-based data and information, frameworks for building web-based R applications, and online services that can be accessed from R. A list of available packages and functions is presented below, grouped by the type of activity. The rOpenSci Task View: Open Data provides further discussion of online data sources that can be accessed from R.

If you have any comments or suggestions for additions or improvements for this Task View, go to GitHub and submit an issue , or make some changes and submit a pull request . If you can't contribute on GitHub, send Scott an email . If you have an issue with one of the packages discussed below, please contact the maintainer of that package. If you know of a web service, API, data source, or other online resource that is not yet supported by an R package, consider adding it to the package development to do list on GitHub .

Tools for Working with the Web from R

Core Tools For HTTP Requests

There are three main packages that should cover most use cases of interacting with the web from R. crul is an R6-based HTTP client that provides asynchronous

source: https://cran.r-project.org/web/views/WebTechnologies.html