Modern Geospatial Data Analysis with R

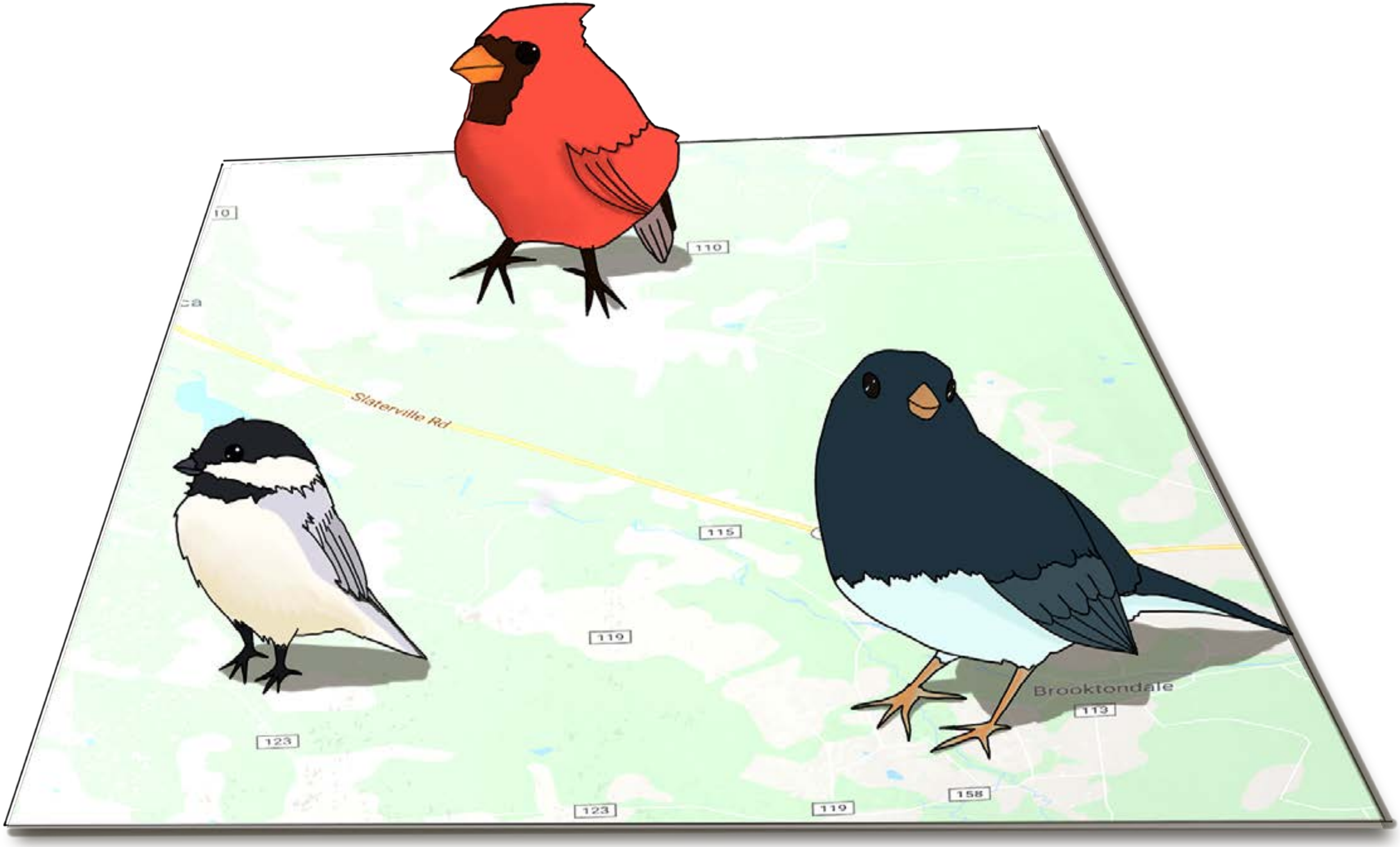
Zev Ross

rstudio::conf

SAN FRANCISCO // JANUARY 27 - 30, 2020

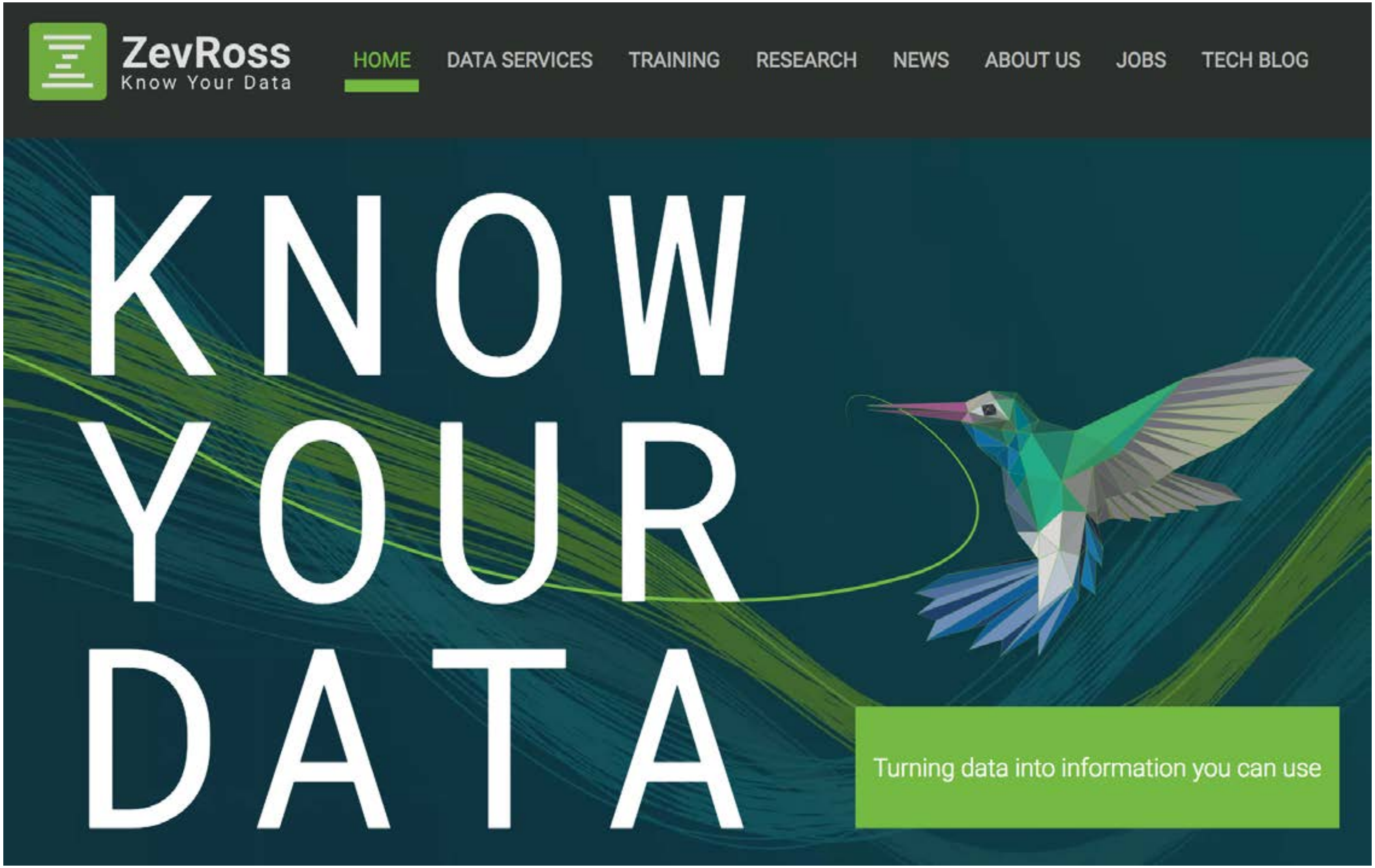
from RStudio

Spatial Birds of a Feather on Thursday at lunch



A little detail about the people in the room

Starting with me

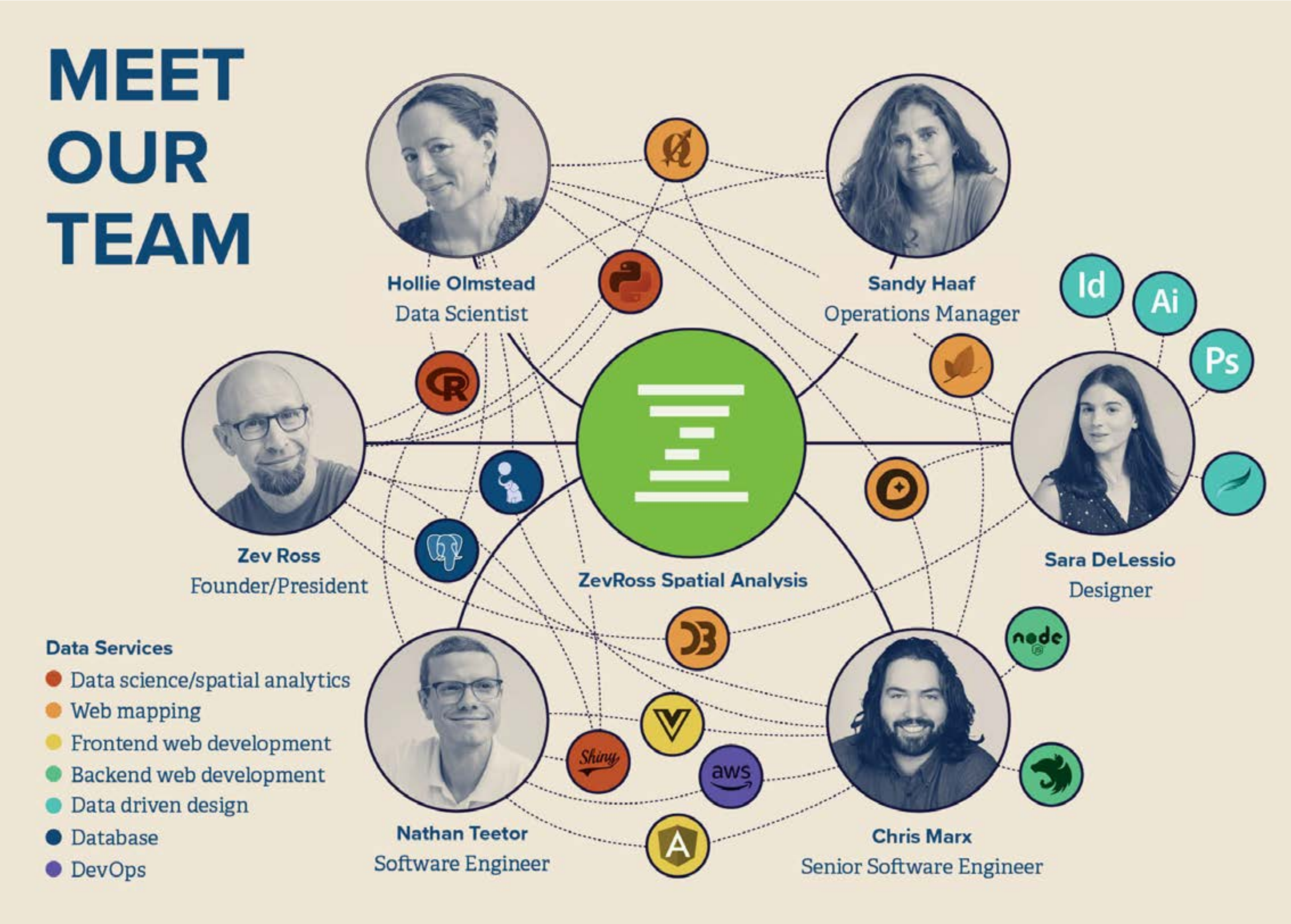


ZevRoss
Know Your Data

HOME DATA SERVICES TRAINING RESEARCH NEWS ABOUT US JOBS TECH BLOG

KNOW YOUR DATA

Turning data into information you can use



MEET OUR TEAM

Hollie Olmstead
Data Scientist

Sandy Haaf
Operations Manager

Zev Ross
Founder/President

ZevRoss Spatial Analysis

Sara DeLessio
Designer

Nathan Teetor
Software Engineer

Chris Marx
Senior Software Engineer

Data Services

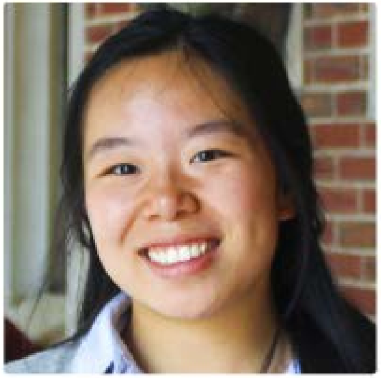
- ● Data science/spatial analytics
- ● Web mapping
- ● Frontend web development
- ● Backend web development
- ● Data driven design
- ● Database
- ● DevOps

Ithaca

Image by Matthew Conheady, nyfalls.com

Workshop assistants

Hollie Olmstead


Angela Li

Dennis Irorere

Thomas Mock

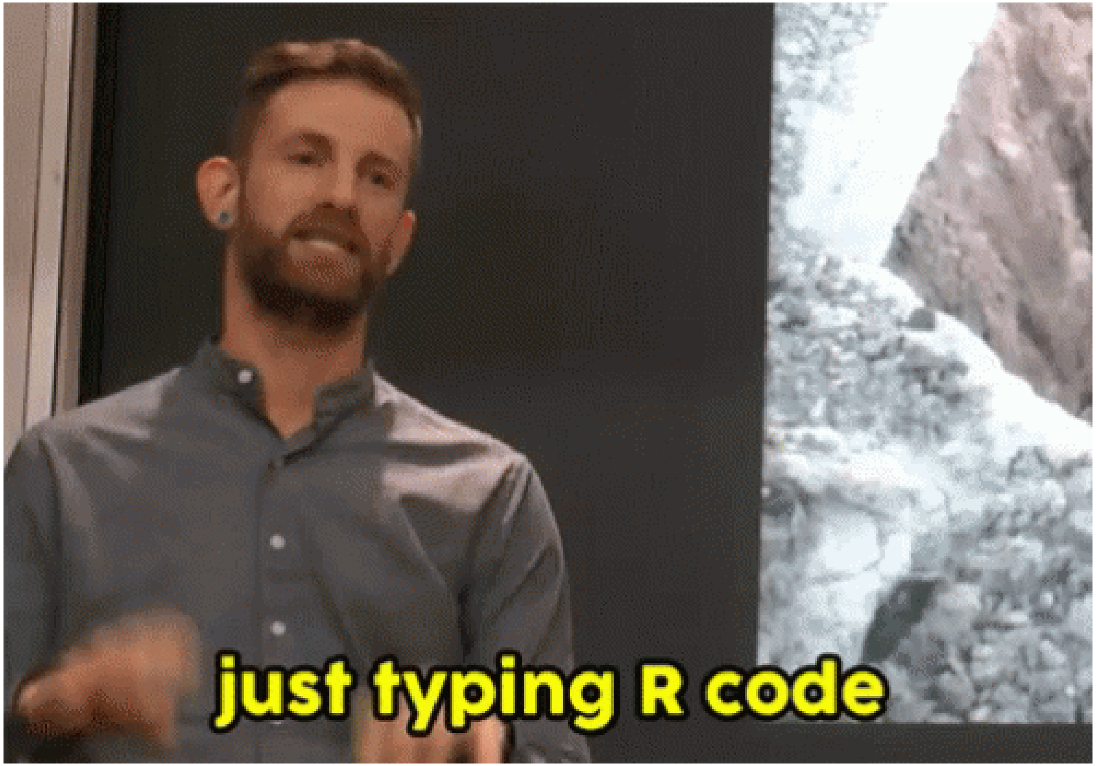
Jindra Lacko

What is your experience with spatial data?

- Have you worked with spatial data before?
- Have you worked with ArcGIS or QGIS (or related)?
- Have you made a map in R?
- Have you done any geoprocessing in R?

My expectations about what you already know about R

I'm assuming you already know R



Including {dplyr}

- `summarize()`
- `mutate()`
- `select()`
- `group_by()`

You should know the pipe

An example of the pipe

```r
starwars %>%
  group_by(hair_color) %>%
  summarize(mean_height = mean(height, na.rm = TRUE))
```

As well as the `::`

```r
readr::read_csv("data/mydata.csv")
```

If %>% and :: are unfamiliar…

This is fine, but talk with a TA and use Google during a break to get comfortable with them

If {dplyr} is a mystery to you

Talk with a TA please, the workshop won't be impossible but...

it will be a challenge

With respect to what you know about spatial data

No experience is expected and I'm guessing there is a huge range!

Workshop agenda

Workshop agenda

- Intro (this section)
- Getting your spatial data into R
- Mapping your spatial data
- Coordinate reference systems (CRS)

Workshop agenda continued

- Getting to know vector data in R
- Getting to know raster data in R
- Geoprocessing with vectors and rasters (with three pieces)
    - Single vector layer geoprocessing
    - Multi vector layer geoprocessing
    - Raster layer geoprocessing

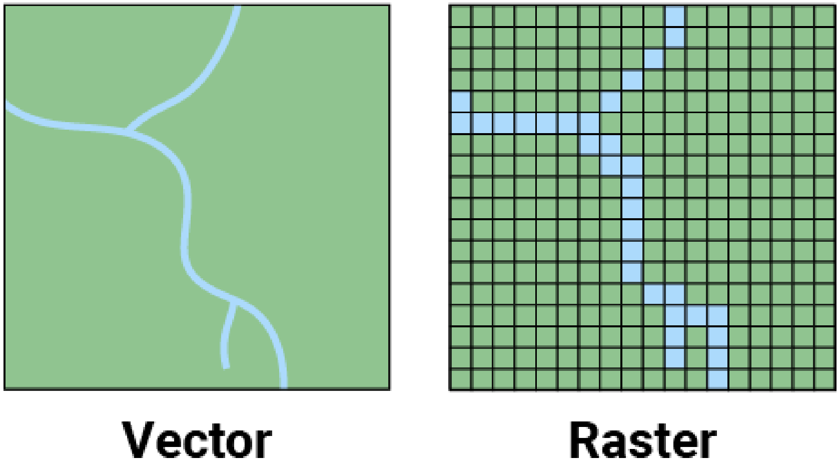
A note on why I chose to organize the way I did

What is spatial data?

A note on the terms spatial vs geospatial vs geographic

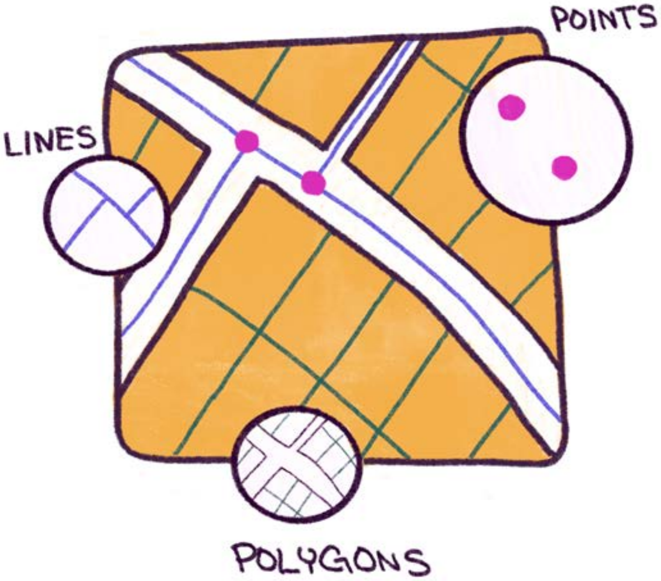
- I use them interchangeably
- Technically "spatial" can refer to non-earth based positions and geographic/geospatial is a subset

Vector vs raster spatial data



Vector data

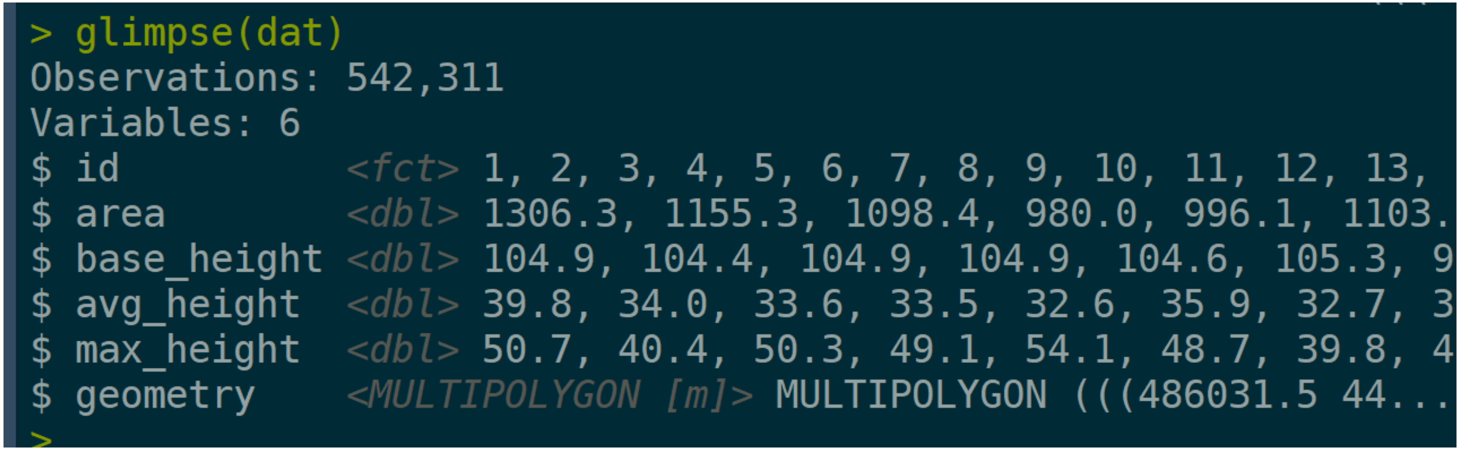
Points, lines and polygons



Vector data also can have non-spatial variables

- Points, lines and polygons can have associated, non-spatial data

- In the example below the **non-spatial** variables of building footprints in Philadelphia are `id`, `area`, `base_height`, `avg_height` and `max_height`.



```
> glimpse(dat)
Observations: 542,311
Variables: 6
$ id          <fct> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
$ area        <dbl> 1306.3, 1155.3, 1098.4, 980.0, 996.1, 1103.
$ base_height <dbl> 104.9, 104.4, 104.9, 104.9, 104.6, 105.3, 9
$ avg_height  <dbl> 39.8, 34.0, 33.6, 33.5, 32.6, 35.9, 32.7, 3
$ max_height  <dbl> 50.7, 40.4, 50.3, 49.1, 54.1, 48.7, 39.8, 4
$ geometry    <MULTIPOLYGON [m]> MULTIPOLYGON (((486031.5 44...
>
```

Vector data comes in a variety of different file formats

- Shapefiles
- Geopackages
- GeoJSON

We will cover this in more detail

Raster data

Raster data is a grid of pixels with values

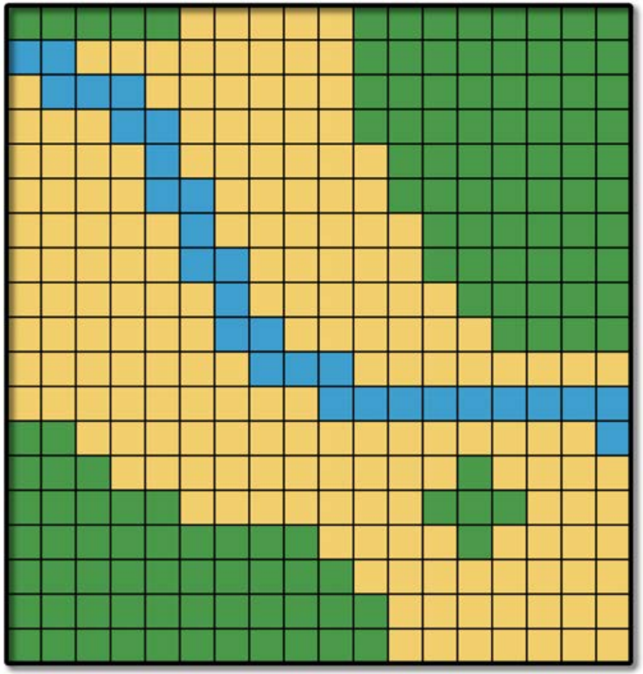
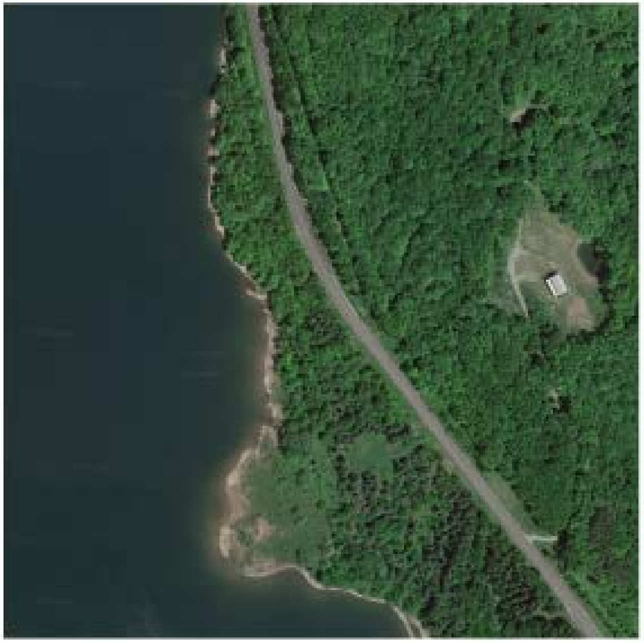
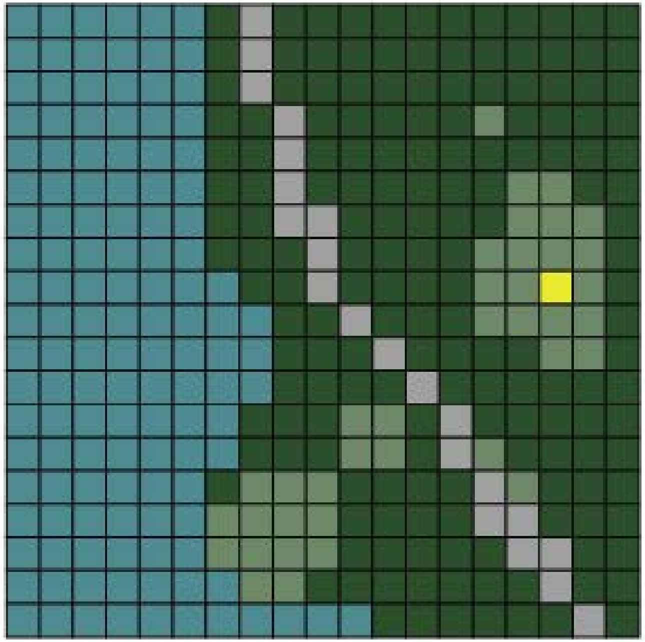


Image rasters vs data rasters





Raster data comes in a variety of different file formats

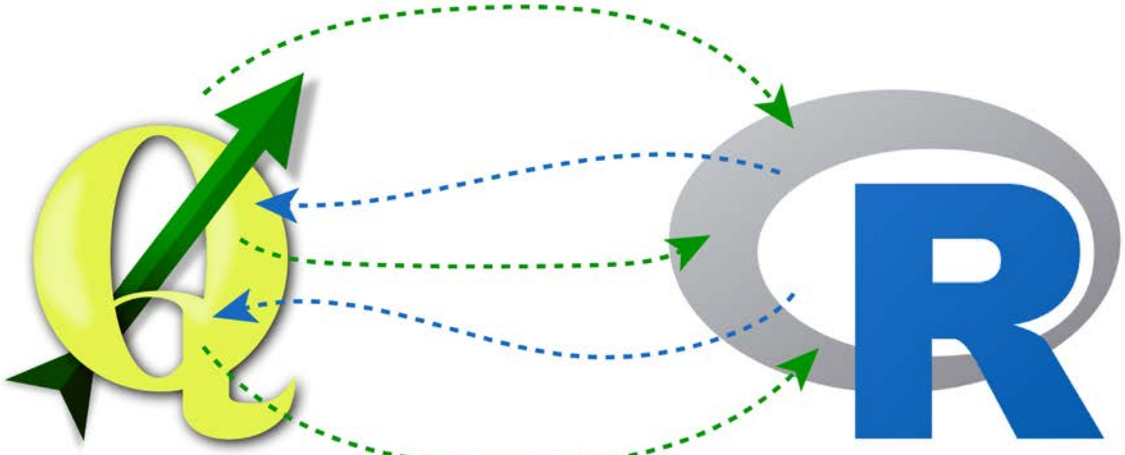
- IMG
- TIF
- SID

Working with spatial data

Traditionally, spatial data has been handled with dedicated spatial software (e.g., GIS)

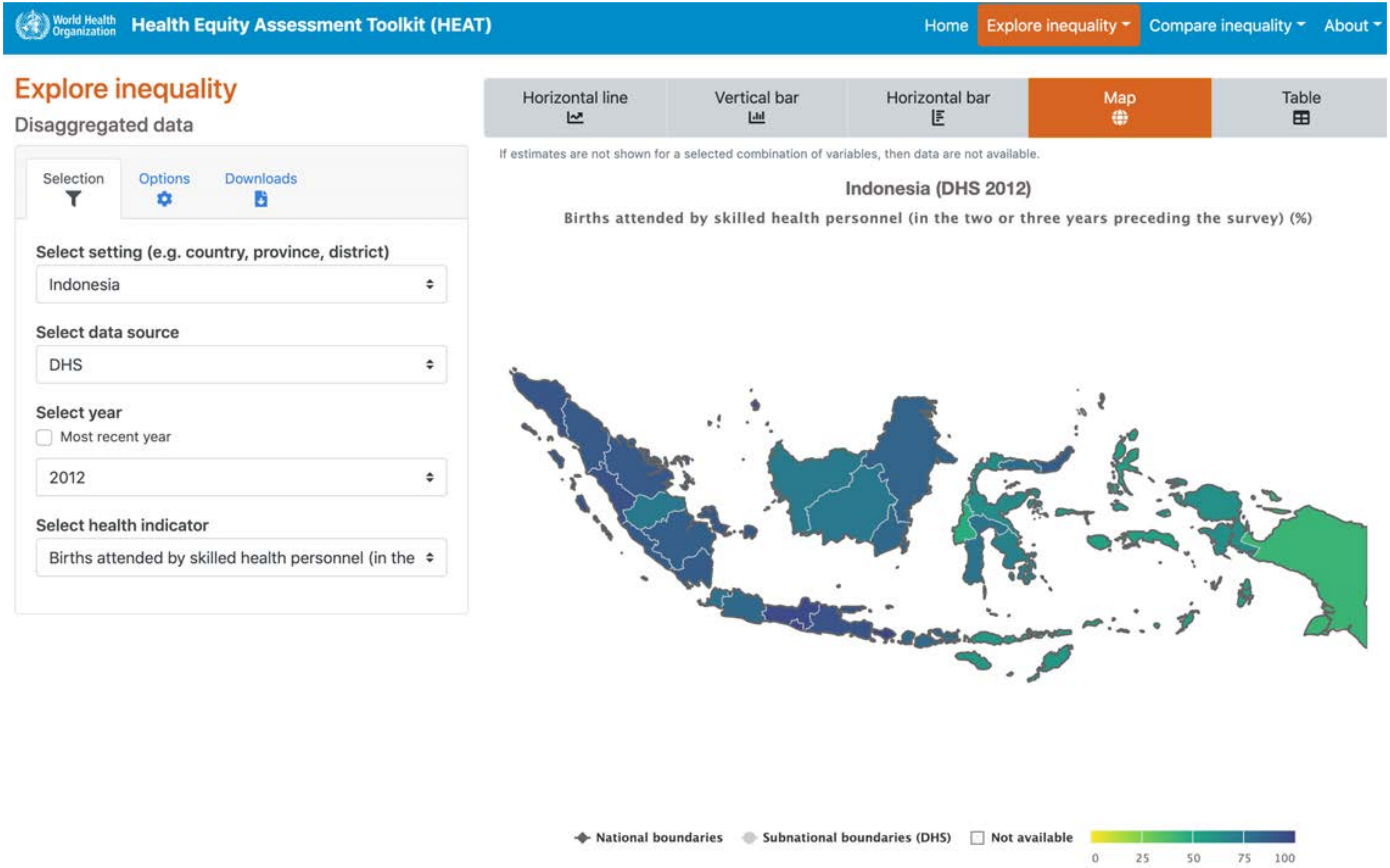
- ArcGIS
- QGIS
- ERDAS IMAGINE
- ENVI

Historically, my workflow looked something like this



But spatial analysis in R has gotten so good that I do most of my spatial work in R

Example 1: Health equity maps



Explore inequality

Disaggregated data

Select setting (e.g. country, province, district)

Indonesia

Select data source

DHS

Select year

□ Most recent year

2012

Select health indicator

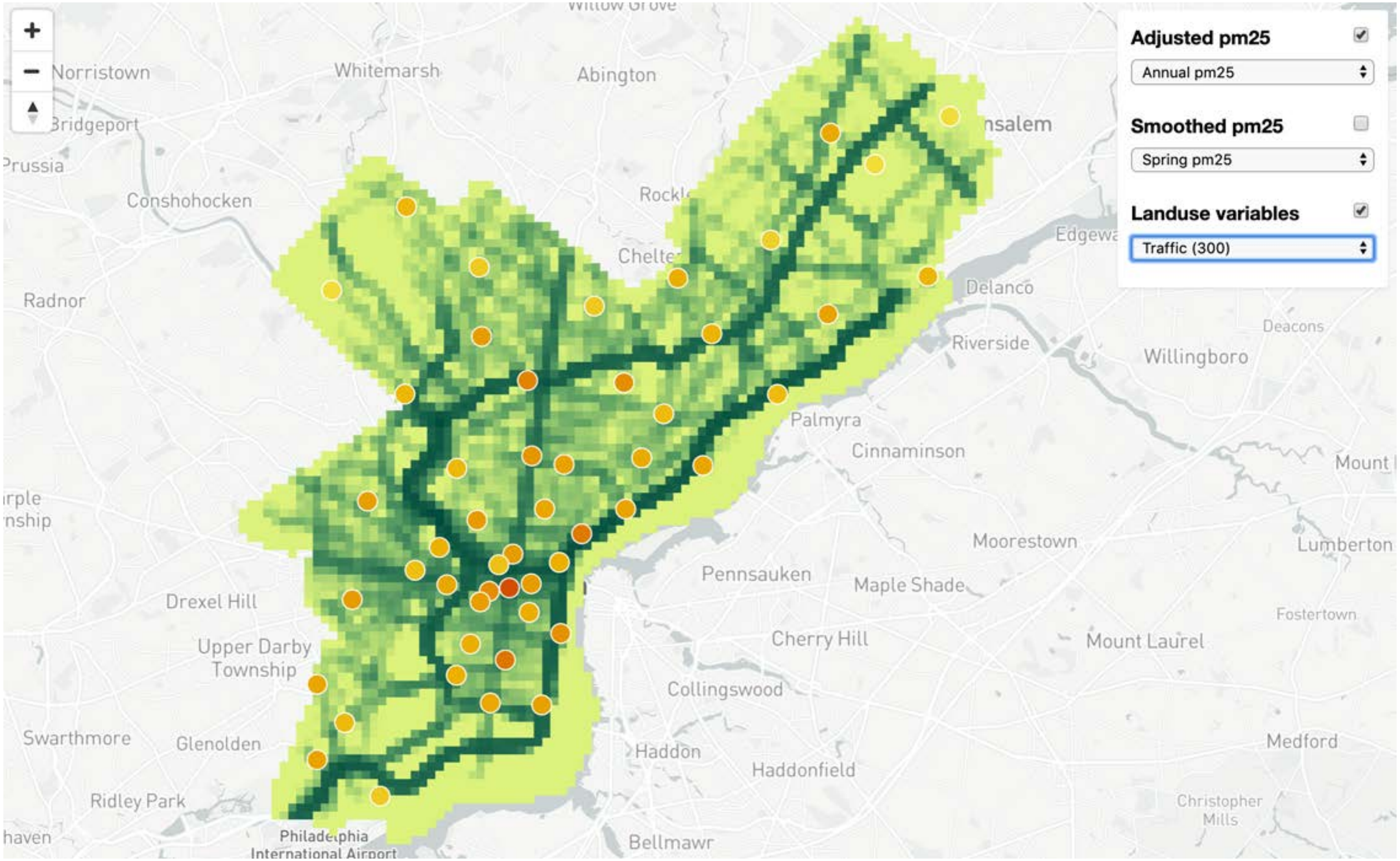
Births attended by skilled health personnel (in the

If estimates are not shown for a selected combination of variables, then data are not available.

Indonesia (DHS 2012)

Births attended by skilled health personnel (in the two or three years preceding the survey) (%)

Example 2: Air quality modeling



Adjusted pm25 ☑

Annual pm25

Smoothed pm25 □

Spring pm25

Landuse variables ☑

Traffic (300)

The #rspatial package landscape

By the way the curly brackets {} denote a package

Most spatial processing and visualization can be done with these packages

- {sf}
- {raster}
- {tmap} or {ggplot2}
- {mapview} or {leaflet}

{sf}

A package for vector data

{raster}

For working with raster data (obviously!)

{tmap}

For creating static (and interactive!) maps

{mapview}

For creating interactive maps

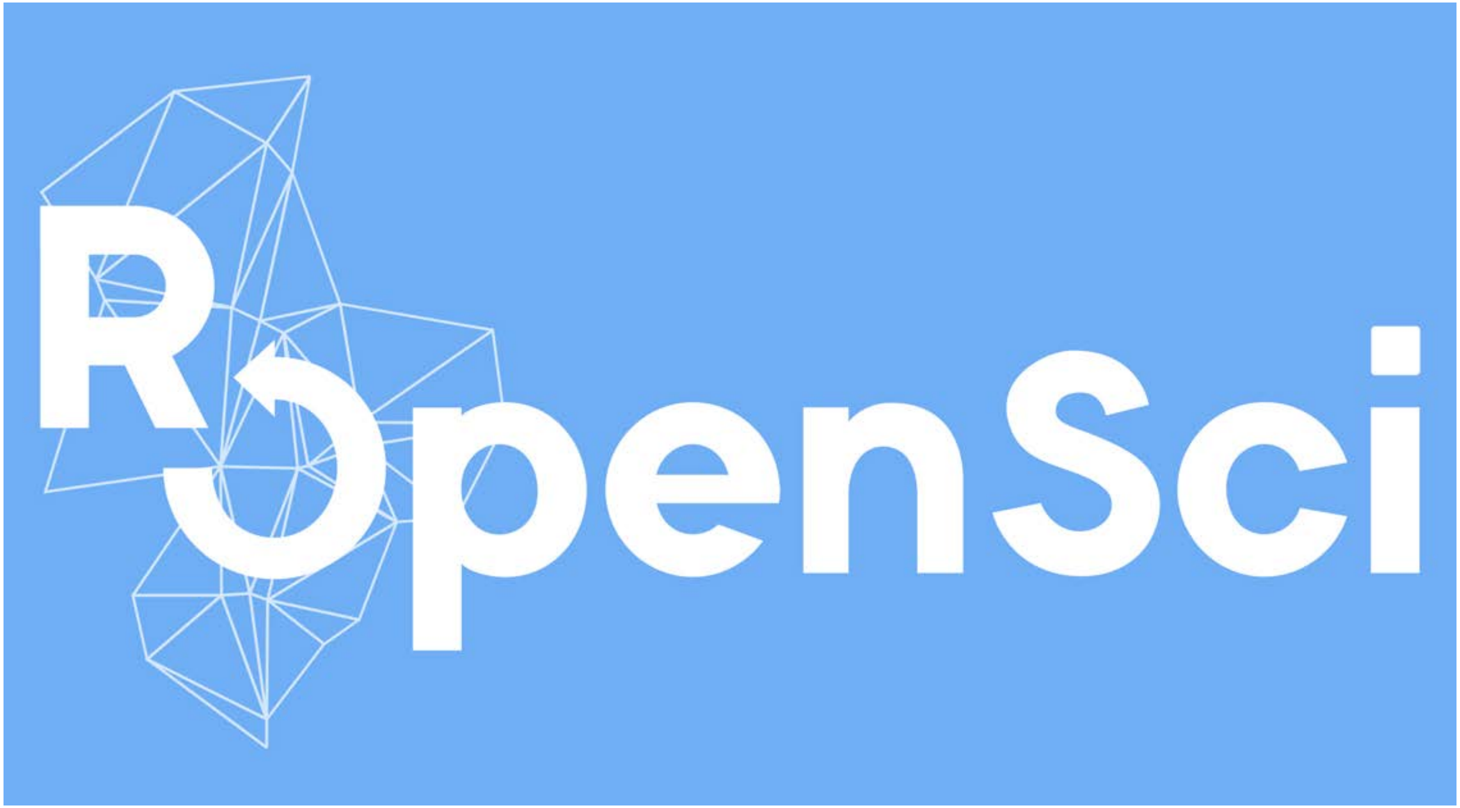
But there are dozens more spatial packages for specific needs

- {ggspatial}
- {leaflet}
- {concaveman}
- {cartography}
- {ggmap}
- {tidycensus}
- {rayshader}
- {rgrass7}
- {stars}
- {geogrid}
- {arcgisbridge}

Many of these packages (including {sf}) have non-R dependencies

Key authors of spatial (and spatial-adjacent) packages

rOpenSci sponsors a lot of great (spatial) work!



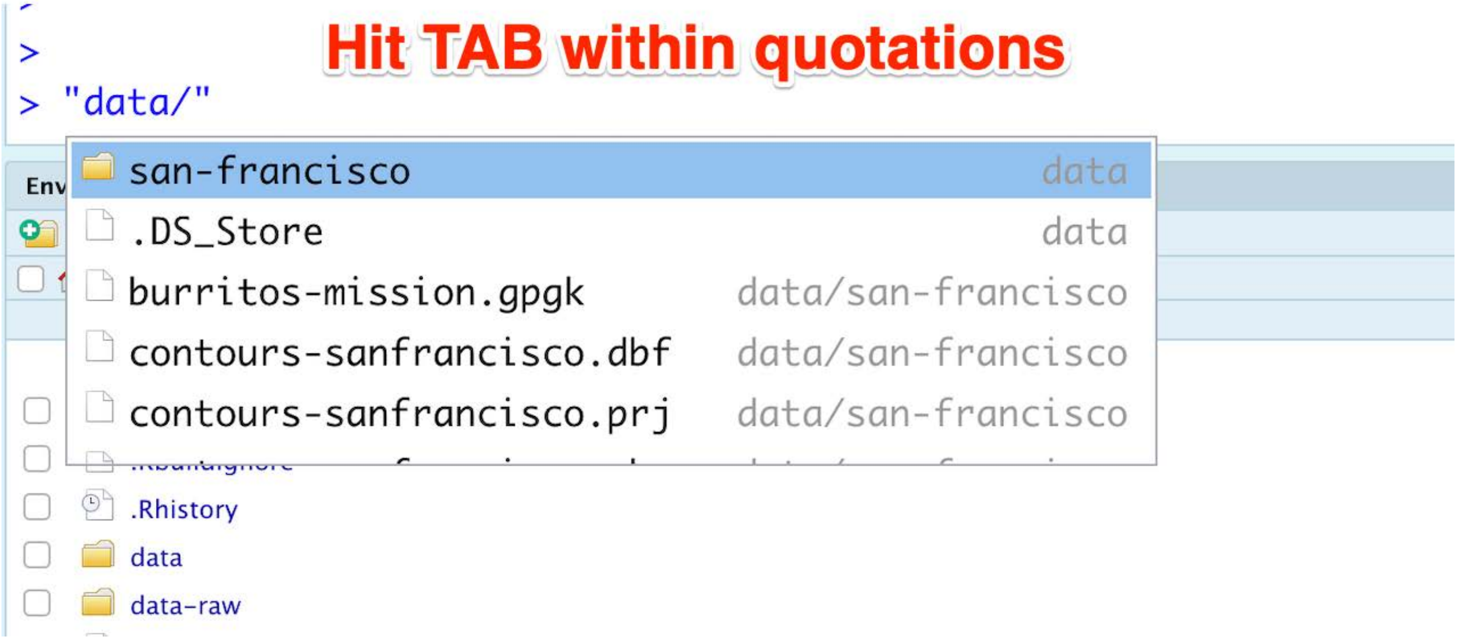
Let's dive in...

Code along with me

Goal here is to give you a sense of what we'll be doing in the workshop

Open a new R script

By the way -- use tab to autocomplete paths



Hit TAB within quotations

```
>
> "data/"
```

```
san-francisco                               data
.DS_Store                                   data
burritos-mission.gpgk          data/san-francisco
contours-sanfrancisco.dbf      data/san-francisco
contours-sanfrancisco.prj      data/san-francisco
```

Load {dplyr}

```r
library(dplyr)
```

Read in US counties near San Francisco

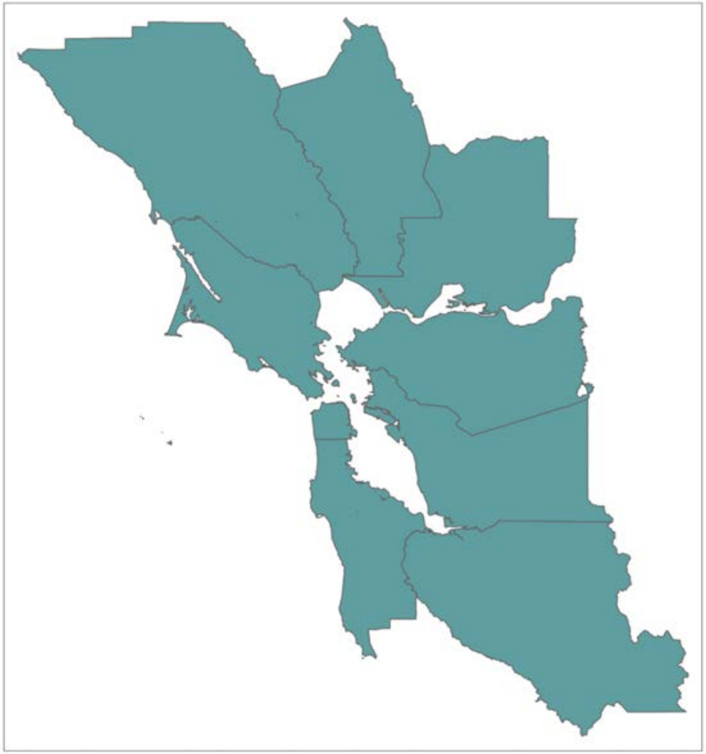
```
library(sf)
bayarea <- read_sf("data/san-francisco/counties-bayarea.shp"
```

```
glimpse(bayarea)
```

```
## Observations: 9
## Variables: 4
## $ county   <chr> "Alameda", "Contra Costa", "Marin",…
## $ fipsstco <chr> "06001", "06013", "06041", "06055",…
## $ objectid <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9
## $ geometry <MULTIPOLYGON [°]> MULTIPOLYGON (((-122.2…
```

Make a quick static map with {tmap}

```r
library(tmap)
tm_shape(bayarea) +
  tm_polygons(col = "cadetblue")
```

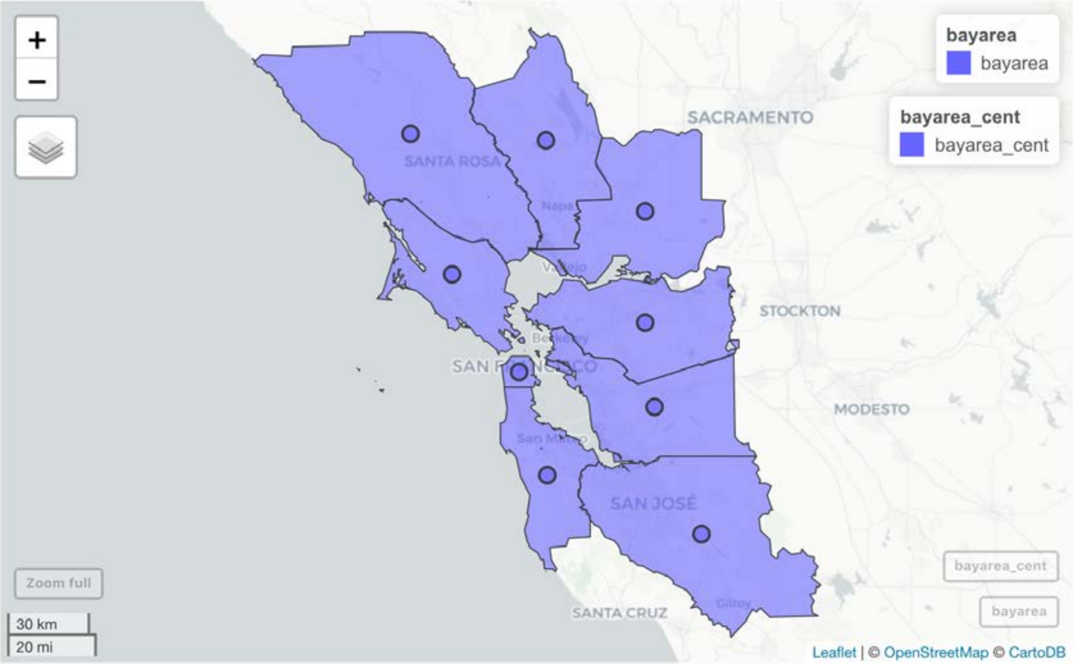


Compute centroids with {sf}

```
bayarea_cent <- st_centroid(bayarea)
```

Make a quick interactive map with {mapview}

```r
library(mapview)
mapview(list(bayarea, bayarea_cent))
```



Pre-created examples make it look easy but...

Working with spatial data is not always smooth sailing

Spatial data is more complex than "standard" tabular data

- Coordinate systems
- List columns
- Different geometry types
- Vector + Raster

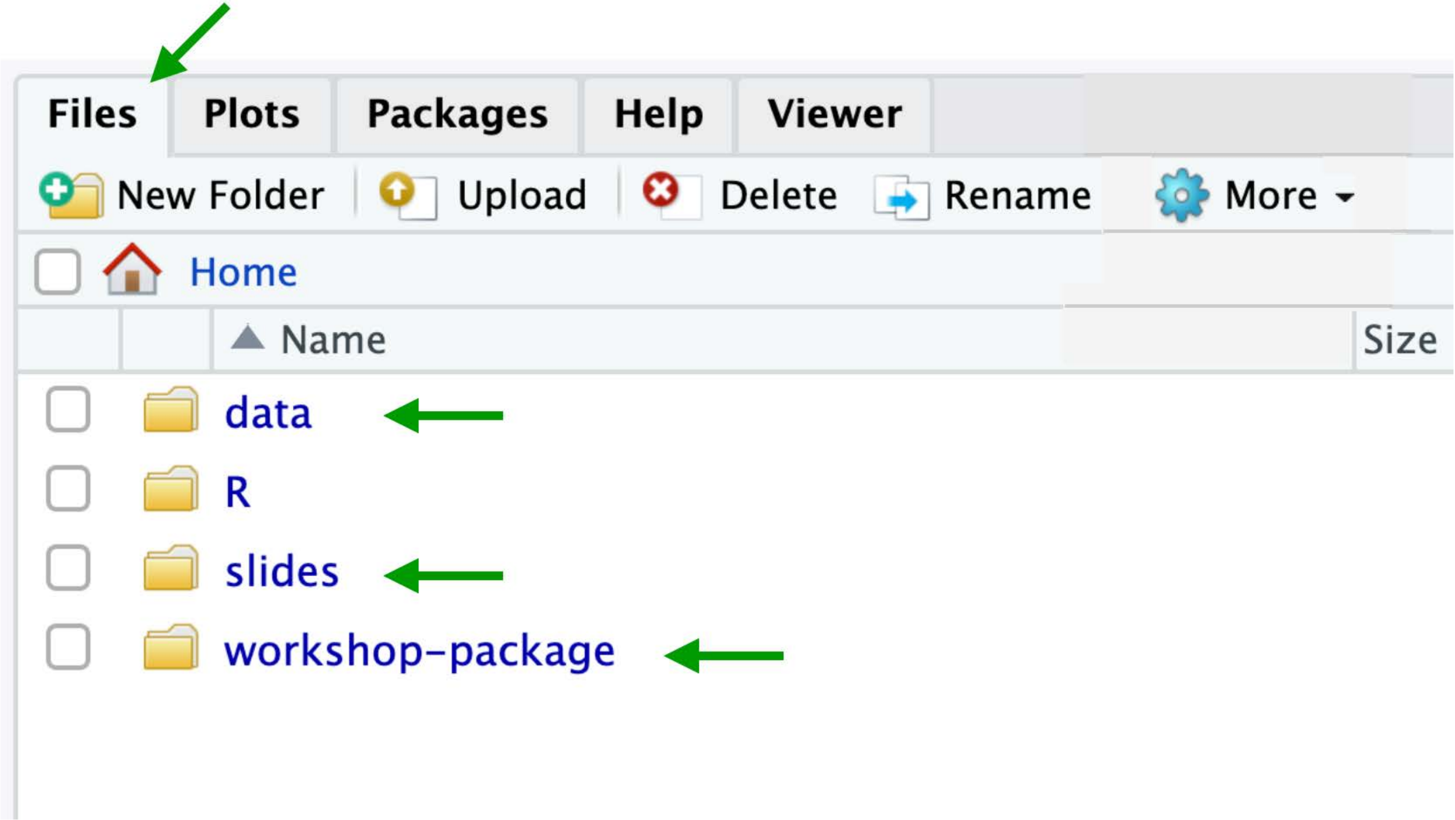
The goal for the workshop: smooth[er] sailing with spatial data in R

Workshop materials

There is a package for this workshop

```r
library(zrsaSpatialWorkshop)
```

Slides, data and the package are in folders in the Files tab



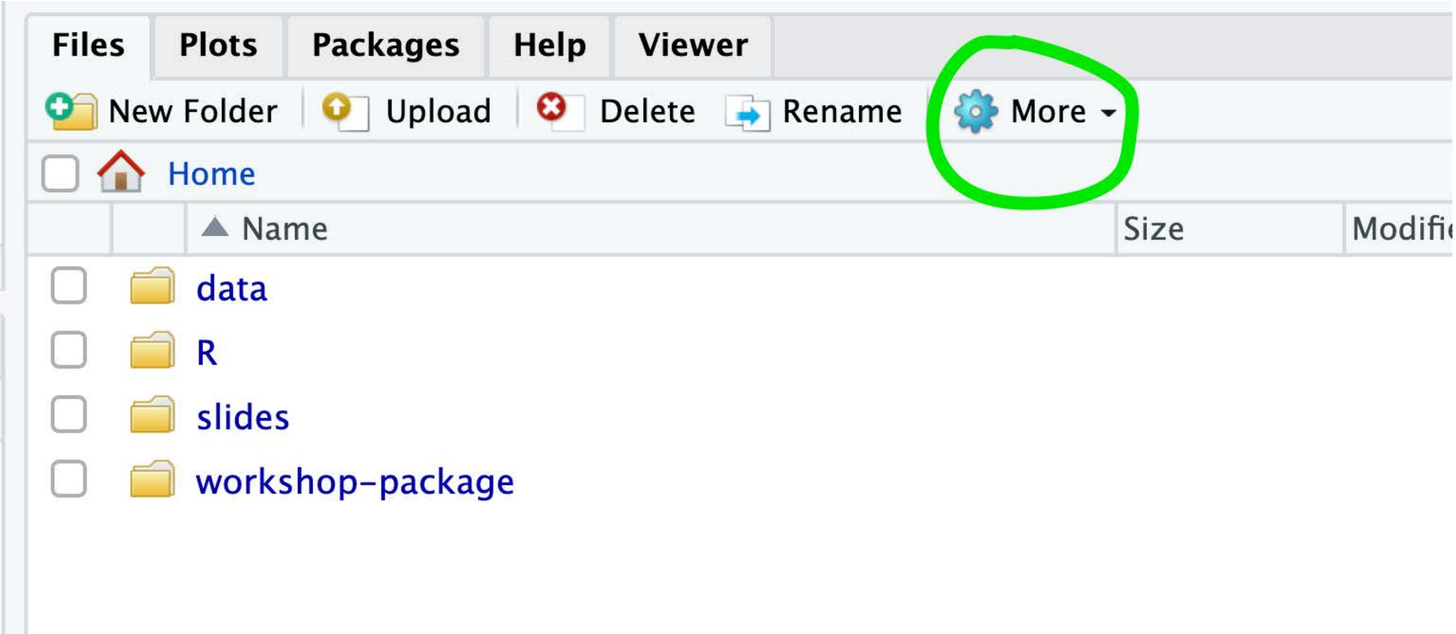
□ data

□ R

□ slides

□ workshop-package

You can download the package (and slides etc)



The workshop exercises data paths expect that you're using the server

If you install the package on your own machine you'll need to change the paths

```
system.file("data-raw/san-francisco/counties-bayarea.shp",
            package = "zrsaSpatialWorkshop")
```

But you don't need to worry about this working on the server

To open an exercise:

- `open_exercise(1)`

To open a solution

- `open_solution(1)`

A note on exercises and solutions

If you run `open_exercise(1)` a second time you will get an error (to prevent you from overwriting the existing file). You can:

- Find the exercise script in the RStudio file explorer and open it there
- Overwrite the existing file and start over with

```
open_exercise(1, overwrite = TRUE)
```

Some exercises have code with `---`

The `---` is a placeholder for something you need to fill in

```
# st_buffer(---)
```

Ready for Exercise 1

- This exercise throws you in to the deep end with explanations to come!

```
library(zrsaSpatialWorkshop)

open_exercise(1)
```