SAE using

Nairobi Workshop:

Ann-Kristin Kreutzmann

Augu

Introduction to geospatia

- One estimate says that 100 TB of only weathe
  - This means there is a lot of data to work w
  - Note that this is also problematic, since it
- Geospatial data is used in a variety of fields
  - Agriculture
  - Urban planning
  - Environmental science
  - Public health
  - Transportation
  - And many more!

The amount of geospatia

- Geospatial data can be highly predictive of e.
  - Urbanity
  - Land class/cover
  - Vegetation indices
  - Population counts
  - etc. etc.
- More importantly: it’s available everywhere!

Think of what you need f

- You need a sample, e.g. a household survey
  - This will only cover some of the country

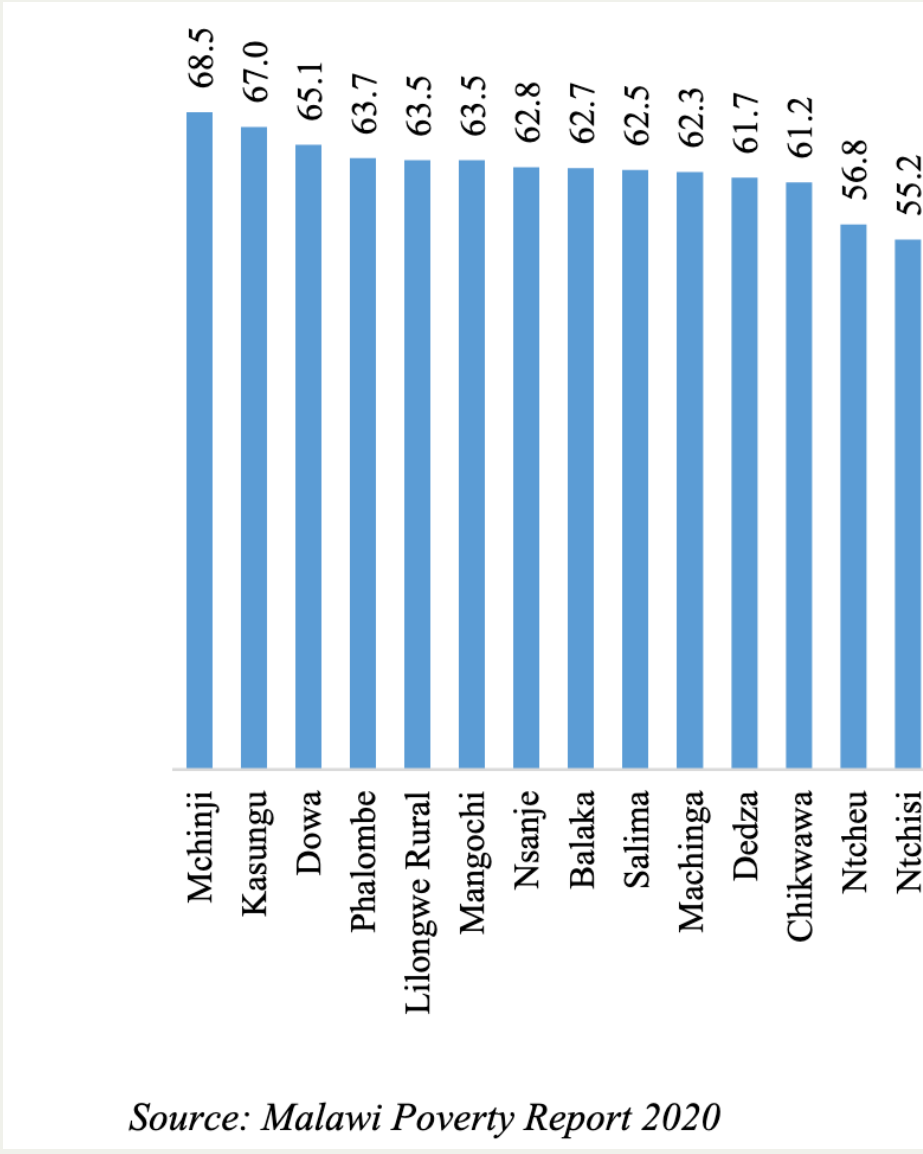
- You need auxiliary data that is:
  - Predictive of the outcome you care about
  - Available throughout the entire country

- Some countries, use administrative data
  - But, importantly, it's often not available or

A quick example

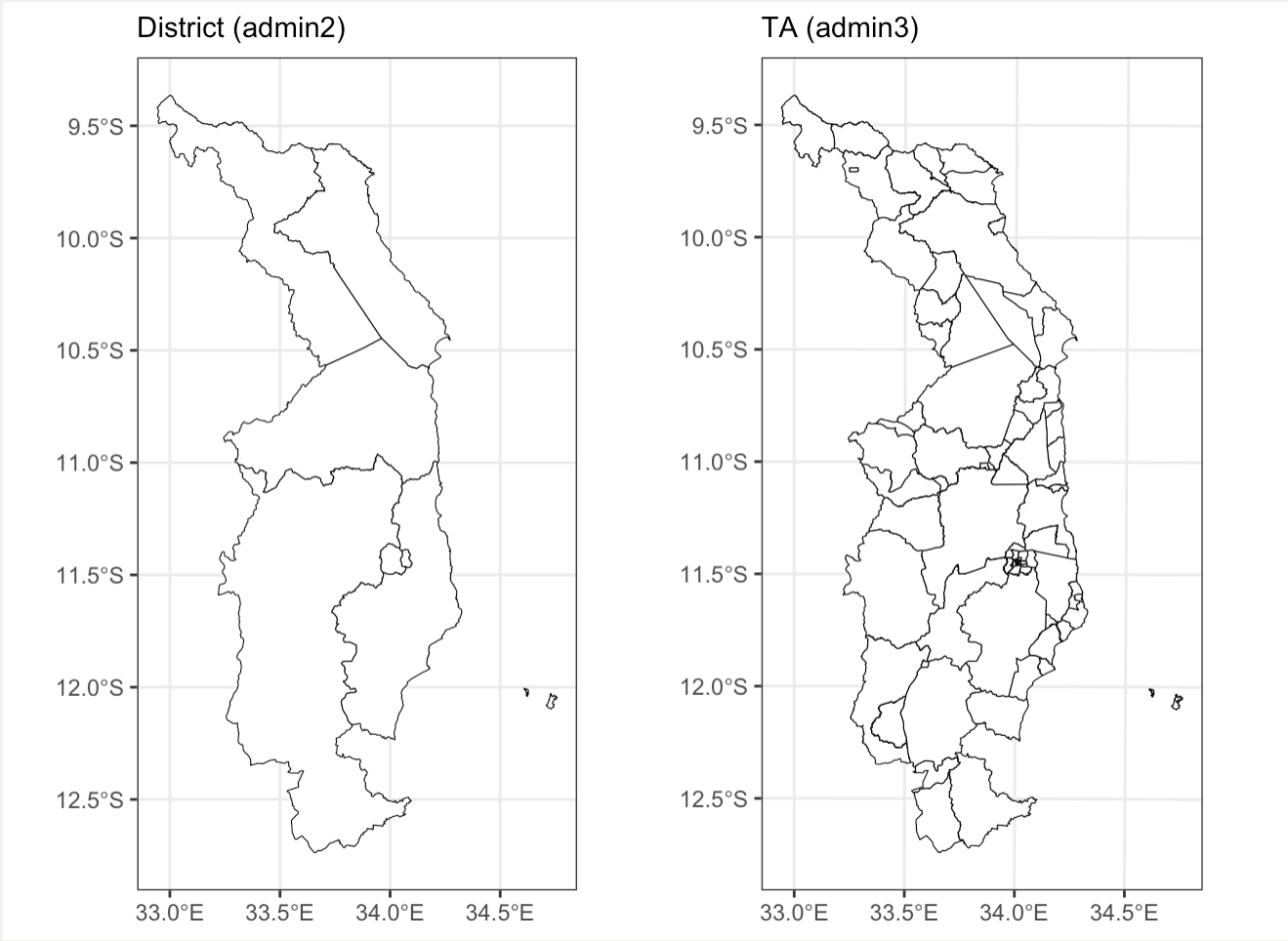
- Let's take a look at Malawi
- Why Malawi?
  - I have survey data you can use 😃
  - Only going to use part of Malawi for this e
- Consider the 2019/2020 Integrated Household
  - Was used for the Malawi Poverty Report 2
  - Can say things about poverty at the distric
  - If you want to split by urban/rural, only at

A quick example

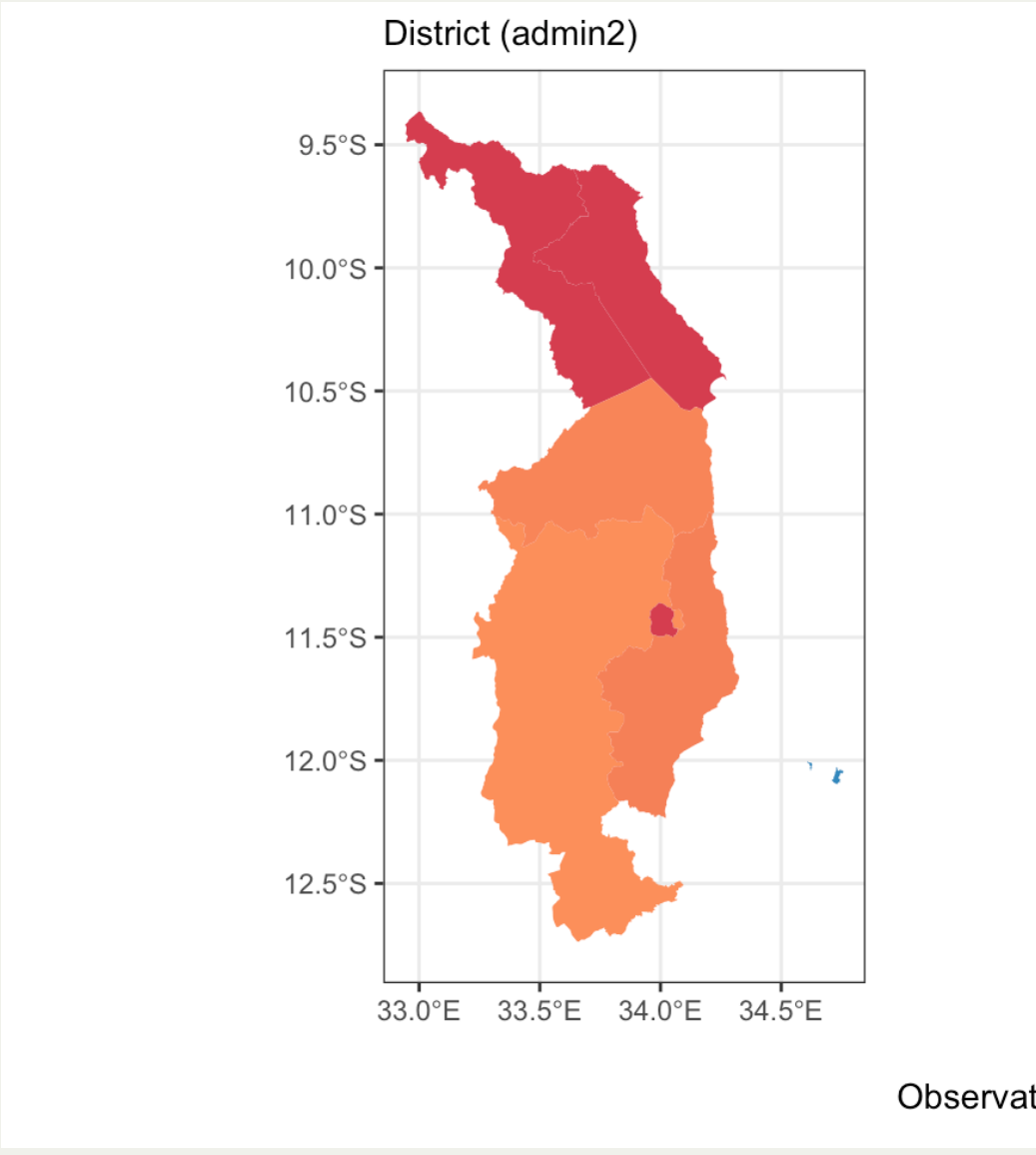


Source: Malawi Poverty Report 2020

Malawi admin areas - No



Observations at the distr



Observat

Sub-area model with sec

- One option: estimate a sub-area model at the
- Steps:
  - Collapse survey data to the EA level
  - Extract geospatial data at the EA level
  - Estimate the model

Getting started with geospatial data

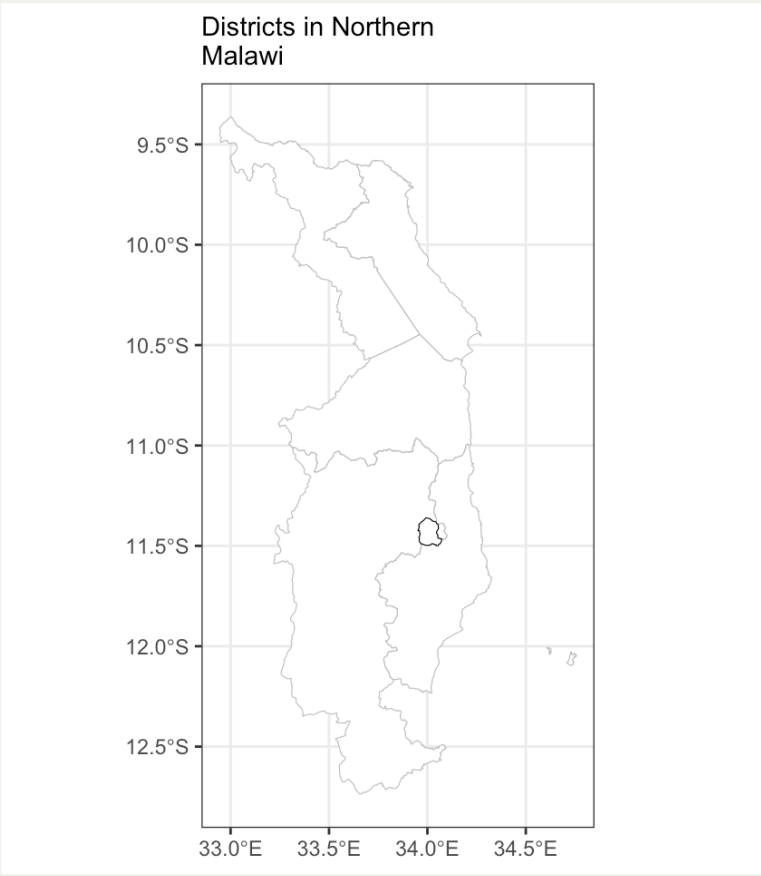
Getting started with geos

- Due to time, this introduction will be necessar
- We are going to learn about the following:
  - Shapefiles
  - Rasters
  - Extracting data

Shapefiles

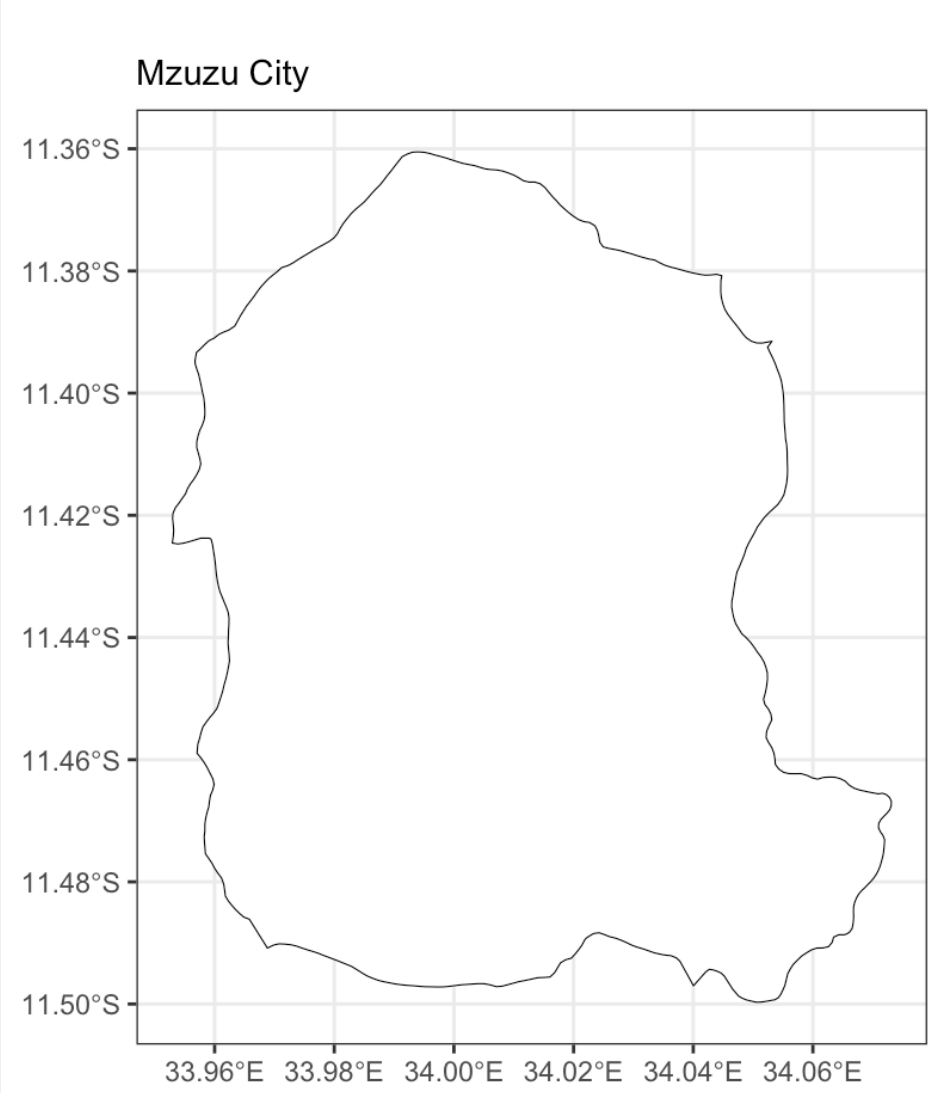
- The maps I just showed you are `shapefiles`
- Shapefiles are a common format for geospati
    - They are a form of `vector` data
- Shapefiles are made up of *at least* four files:
    - `.shp` - the shape itself
    - `.shx` - the index
    - `.dbf` - the attributes
    - `.prj` - the projection
    - What these all mean isn't important for no folder on github.

Let's look at Northern Ma

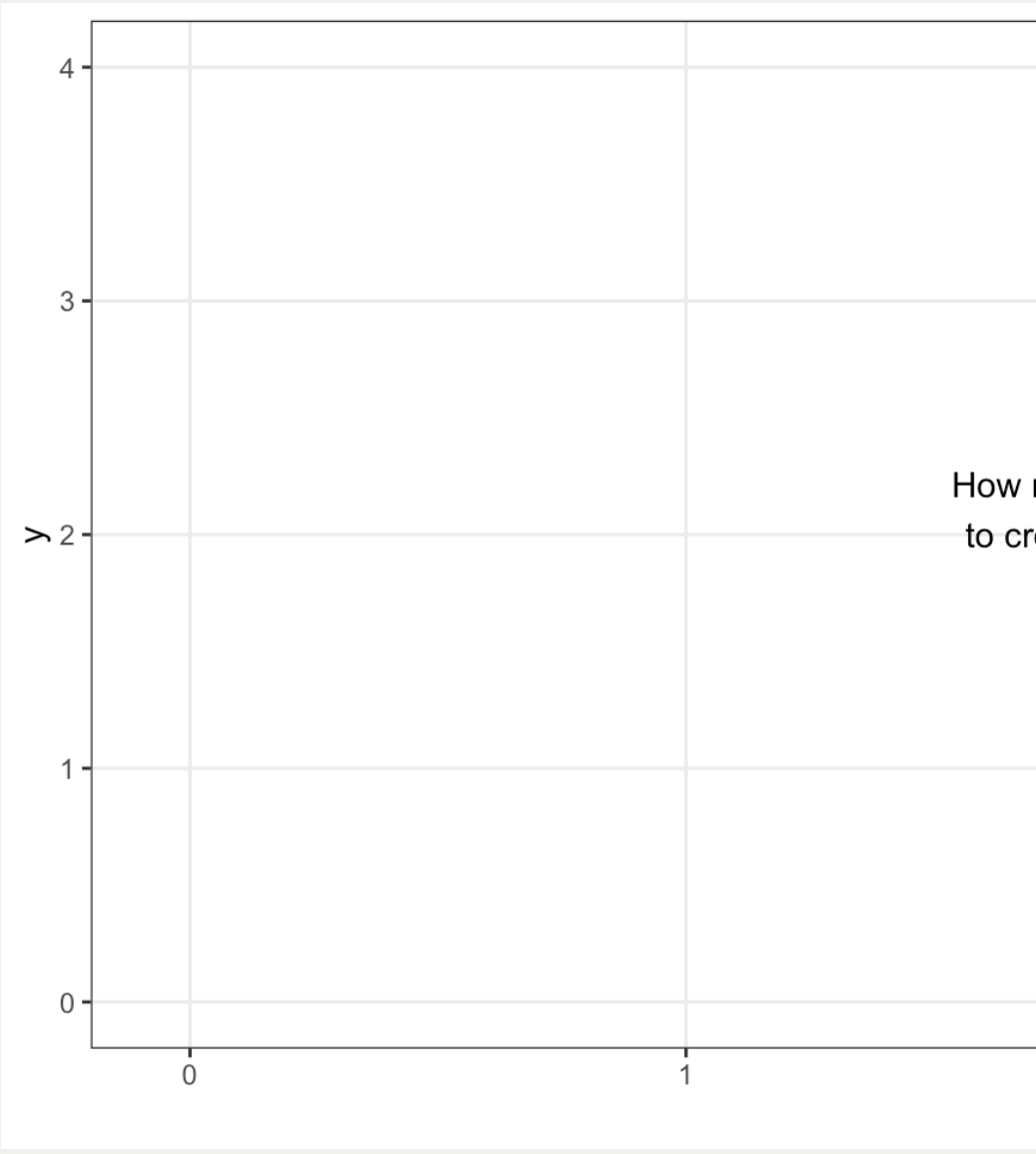


- Left: collection of `features`
- Right: one `feature`

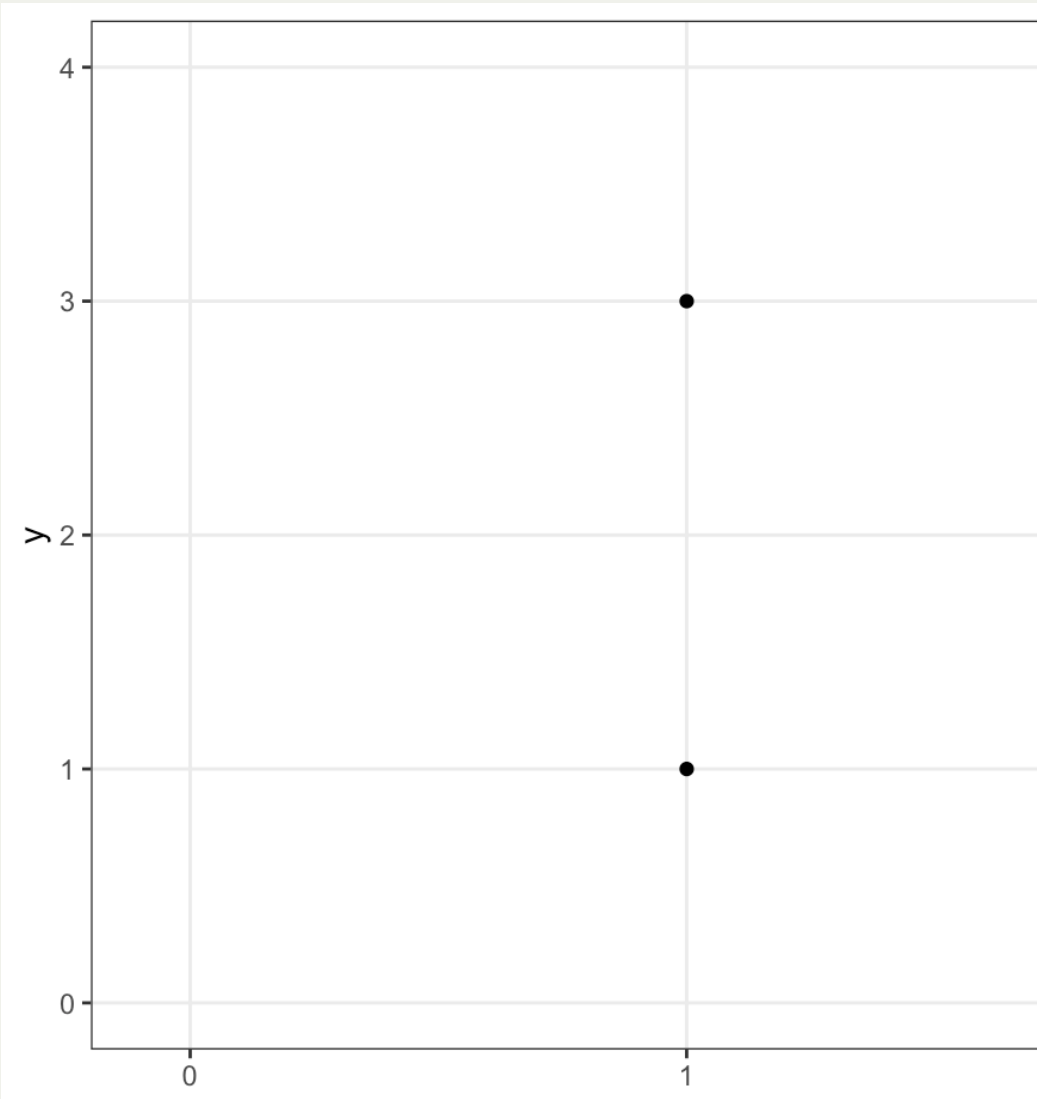
Features are made of ve



Imagine a rectangle, on



Imagine a rectangle, on



Imagine a rectangle, on

- We need four points.
- But features in shapefiles are a little different.
    - We have to “close” the feature
- We do this by adding a fifth point: the same a the first point!

Five points (vertices) in our feature

	X value	Y value
Point 1	1	1
Point 2	3	1
Point 3	3	3
Point 4	1	3
Point 5	1	1

Features are made of ve

- So we have all our vertices (489 of them!)
- The question:
  - What is the coordinate system here?

Latitude and longitude o

- The most common `coordinate reference`
  - Latitude: North/South
  - Longitude: East/West
  - The equator is at 0° latitude
  - The prime meridian is at 0° longitude

- But there's a problem with using latitude/longi
  - The Earth is a sphere (well, more or less;


Source: wikipedia

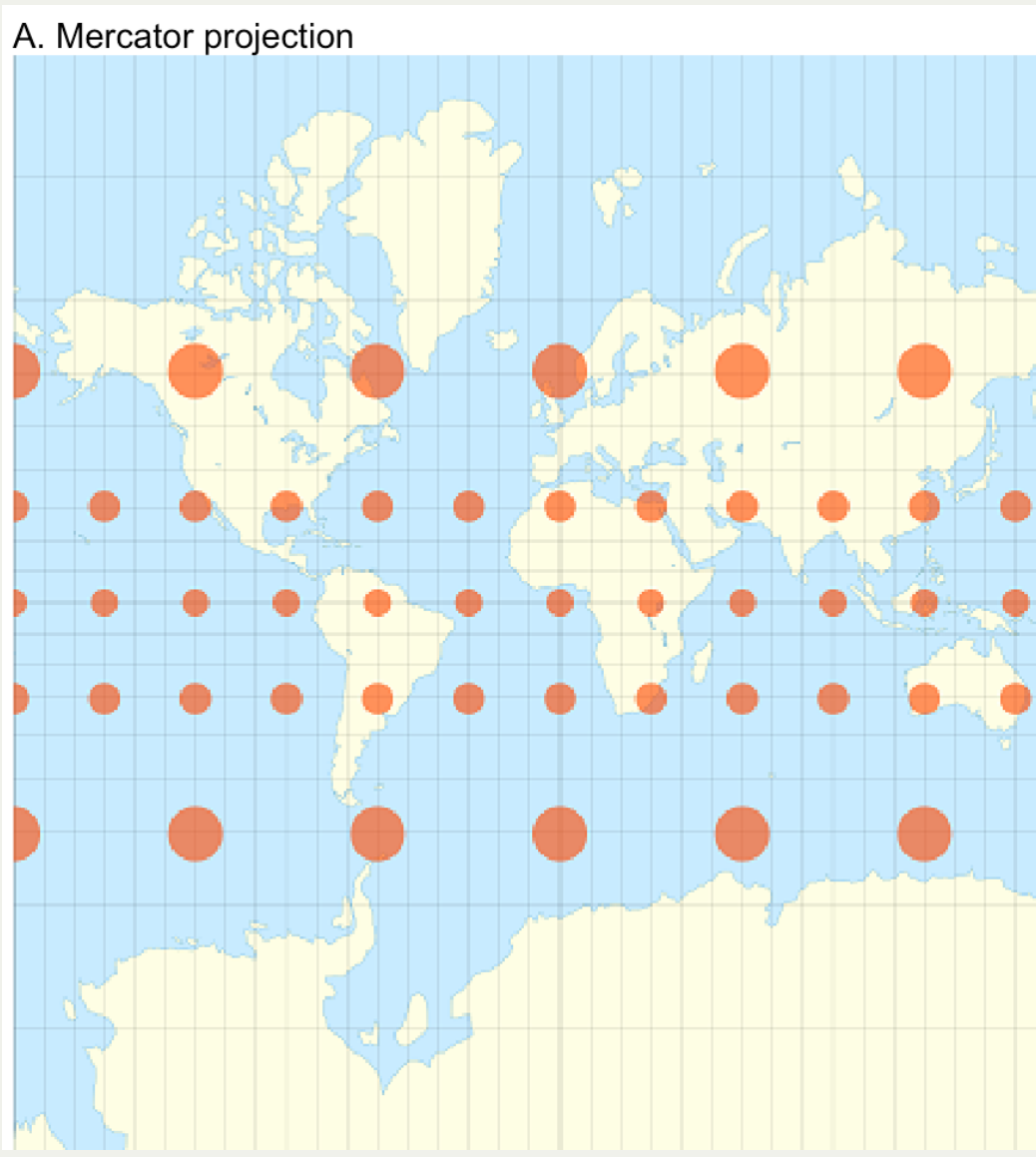

Source: wikipedia
2 million
sq km

The basic problem

- The basic problem is that one degree of longi
  - At the equator, one degree of longitude is
  - At 15N/S, one degree of longitude is abou
  - At 30N/S, one degree of longitude is abou
  - At 45N/S, one degree of longitude is abou
  - At 60N/S, one degree of longitude is abou
    - This explains Greenland!

- It's not an easy problem to solve, as all soluti

Preserve shape, give up



Long story short…

- Using lat/lon is generally fine as long as you d
  - But if you do, you need to use a different
- Today we will focus on things that do not requ
  - So we will generally use lat/lon

Reading shapefiles in R

- My go-to package for shapefiles in R is `sf`
- Reading shapefiles is VERY easy! And you ca

▼ Code

```
1 library(sf)
2 # this is the shapefile for the northern region of M
3 northmw <- read_sf("day4data/mw2.shp")
4 northmw
```

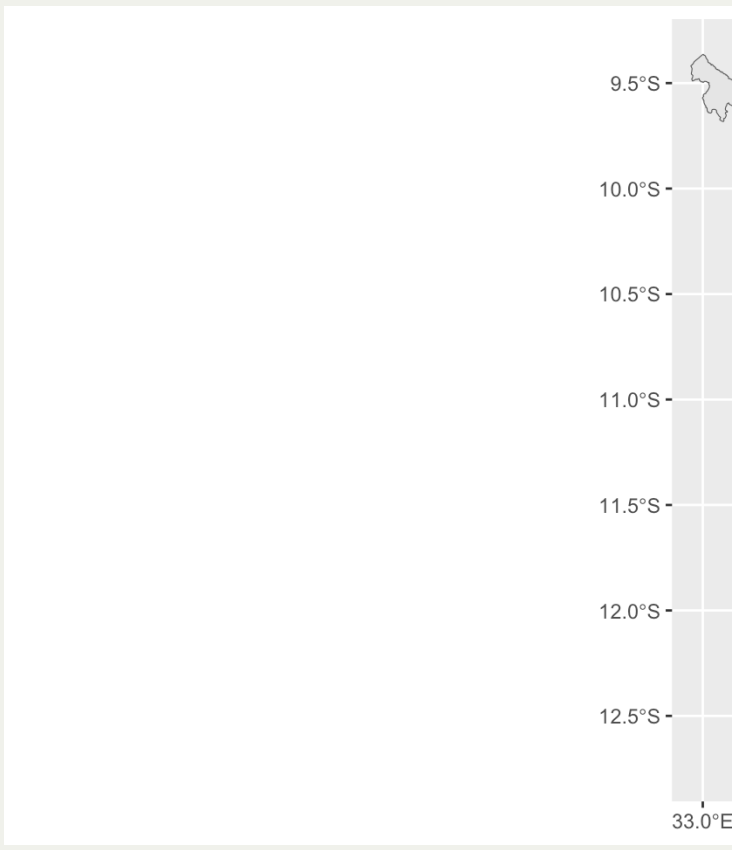
```
Simple feature collection with 7 features and 1 field
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 32.9424 ymin: -12.73669 xmax: 34.75834 ymax:
Geodetic CRS:  WGS 84
# A tibble: 7 × 2
  DIST_CODE
  <chr>
1 101       (((33.00618 -9.367976, 33.00641 -9.368012, 33.00666 -
2 102       (((33.73312 -9.581763, 33.73382 -9.582024, 33.73453 -
3 106       (((34.73705 -12.04239, 34.7378 -12.04253, 34.73849 -12
4 105       (((34.079 -11.38773, 34.07986 -11.38818, 34.08067 -11
```

```
5 107       (((34.01161 -11.36523, 34.01255 -11.36546, 34.0135 -1
6 103       (((34.21913 -11.05184, 34.21931 -11.05268, 34.21969 -
7 104       (((34.09466 -10.57433, 34.09563 -10.57449, 34.09763 -
```

Plotting is also very easy

▼ Code

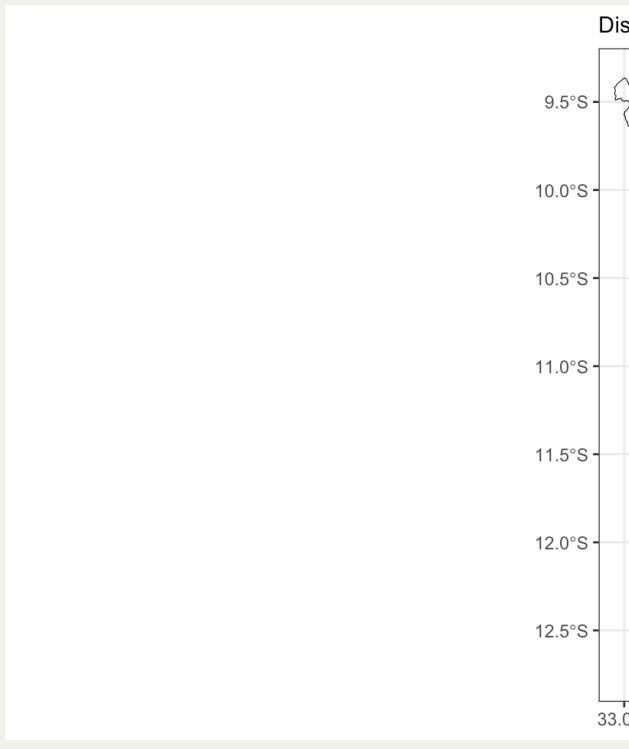
```
1  ggplot() +
2    geom_sf(data = northmw)
```



My go-to theme

▼ Code

```
1  ggplot() +
2    geom_sf(data = northmw, fill = NA, color = "black"
3    theme_bw() +
4    labs(subtitle = "Districts in Northern Malawi")
```



Give it a try with TAs (mw

▶ Code

▶ Code

One more example - ma

▼ Code

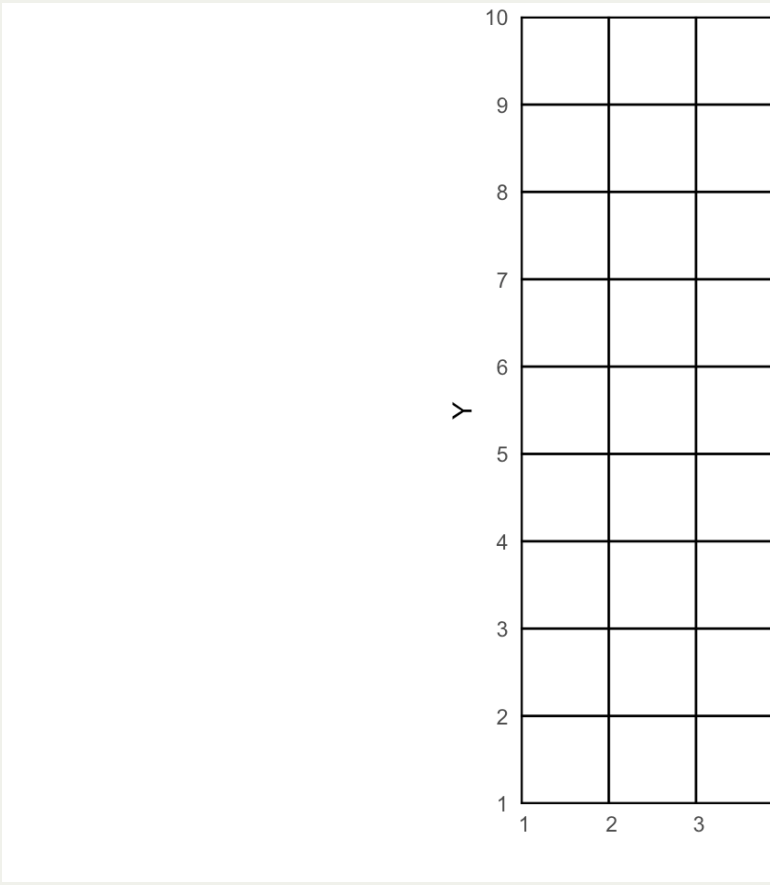
```
admin2 <- read_sf("day4data/mw2.shp")

ggplot() +
  geom_sf(data = admin2,
    fill = "white", color = "gray") +
  geom_sf(data = admin2 |> filter(DIST_CODE=="10
    fill = "white", color = "black") +
  theme_bw() +
  labs(subtitle = "Districts in Northern Malawi"
```

Rasters

- We've discussed shapefiles
  - Now let's talk about rasters!

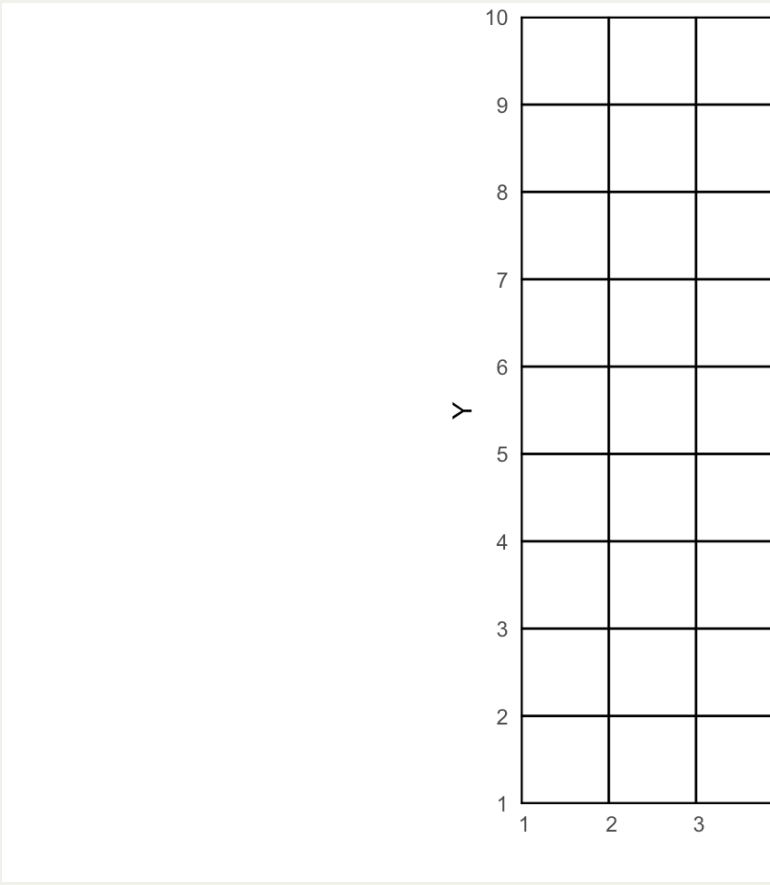
- Rasters are a different type of geospatial data
  - They are made up of a grid of cells
  - Each cell has a value

Example raster grid - ho



- Here's a grid.
  - How many points do we need?

Example raster grid - ho



- Need to know location of one grid cell…
  - And the size of each grid!

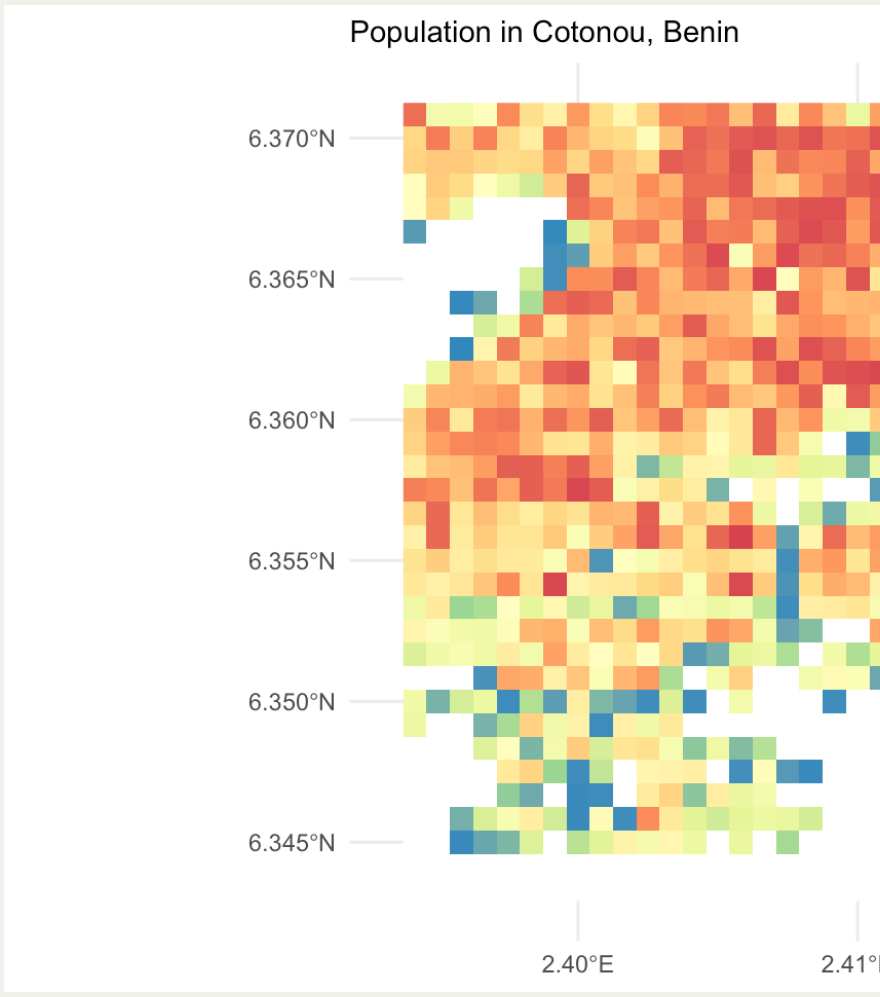
How much info do we ne

- In other words, we do not need a point for eve

- We just need to know:
  - The location of one cell
  - The size of each cell
    - This is called the `resolution` of the

      4
- Example:
  - I know the first grid cell in bottom left is at
  - I know each grid cell is 1 meter by 1 mete
  - Then I know the exact location of every si

Population in Cotonou, B



- What are the white values?

Population in Cotonou, B

- Here's the information for this raster
  - What's the resolution? What are the units?

```
class       : SpatRaster
dimensions  : 34, 46, 1  (nrow, ncol, nlyr)
resolution  : 0.0008333333, 0.0008333333  (x, y)
extent      : 2.39375, 2.432083, 6.342917, 6.37125  (xmin, xmax, y
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : beninpop.tif
name        : beninpop
```

Rasters

- Rasters are defined by the grid layout and the
  - Grid cells are sometimes called pixels (jus

- There are many different file types for rasters
  - `.tif` or `.tiff` (one of the most common
  - `.nc` (NetCDF, common for very large rast
  - Image files, e.g. `png`, `jpg`, etc.

Reading rasters in R

- Reading rasters is also quite easy!
  - Going to use the `terra` package for it
    - Note: can use `terra` for shapefiles, t
  - `day4data/beninpop.tif` is a raster of

▼ Code

```
1 library(terra)
2
3 # this is the raster for Cotonou, Benin
4 cotonou <- rast("day4data/beninpop.tif")
5 cotonou
```

```
class       : SpatRaster
dimensions  : 34, 46, 1  (nrow, ncol, nlyr)
resolution  : 0.0008333333, 0.0008333333  (x, y)
extent      : 2.39375, 2.432083, 6.342917, 6.37125  (xmin, xmax, y
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : beninpop.tif
```

Plotting rasters

- Plotting rasters only with `terra` is a bit of a pain
  - Can't use `ggplot`
  - So, I load another package that lets me use `ggplot` with rasters
    - `tidyterra`

▼ Code

```
library(tidyterra)

ggplot() +
  geom_spatraster(data = cotonou)
```

Making it nicer

▼ Code

```
library(tidyterra)

ggplot() +
  geom_spatraster(data = cotonou) +
  # distiller is for continuous values
  # but we can use palettes!
  # I like spectral a lot
  scale_fill_distiller("Population\ncount",
    palette = "Spectral", na.value = "white") +
  theme_minimal() +
  labs(subtitle = "Population in Cotonou, Benin"
```

Extracting raster data to

- Let's go back to our use case:
  - We want to estimate a sub-area model at
  - This means we need to extract raster data
  - We can do this with `terra`, `sf`, and `exac`
    - `terra` has its own method, but i find

- Let's start by looking at the raster I've uploade

Give it a try

- Try to load it into R using terra, then plot it with tidyterra and ggplot

▶ Code

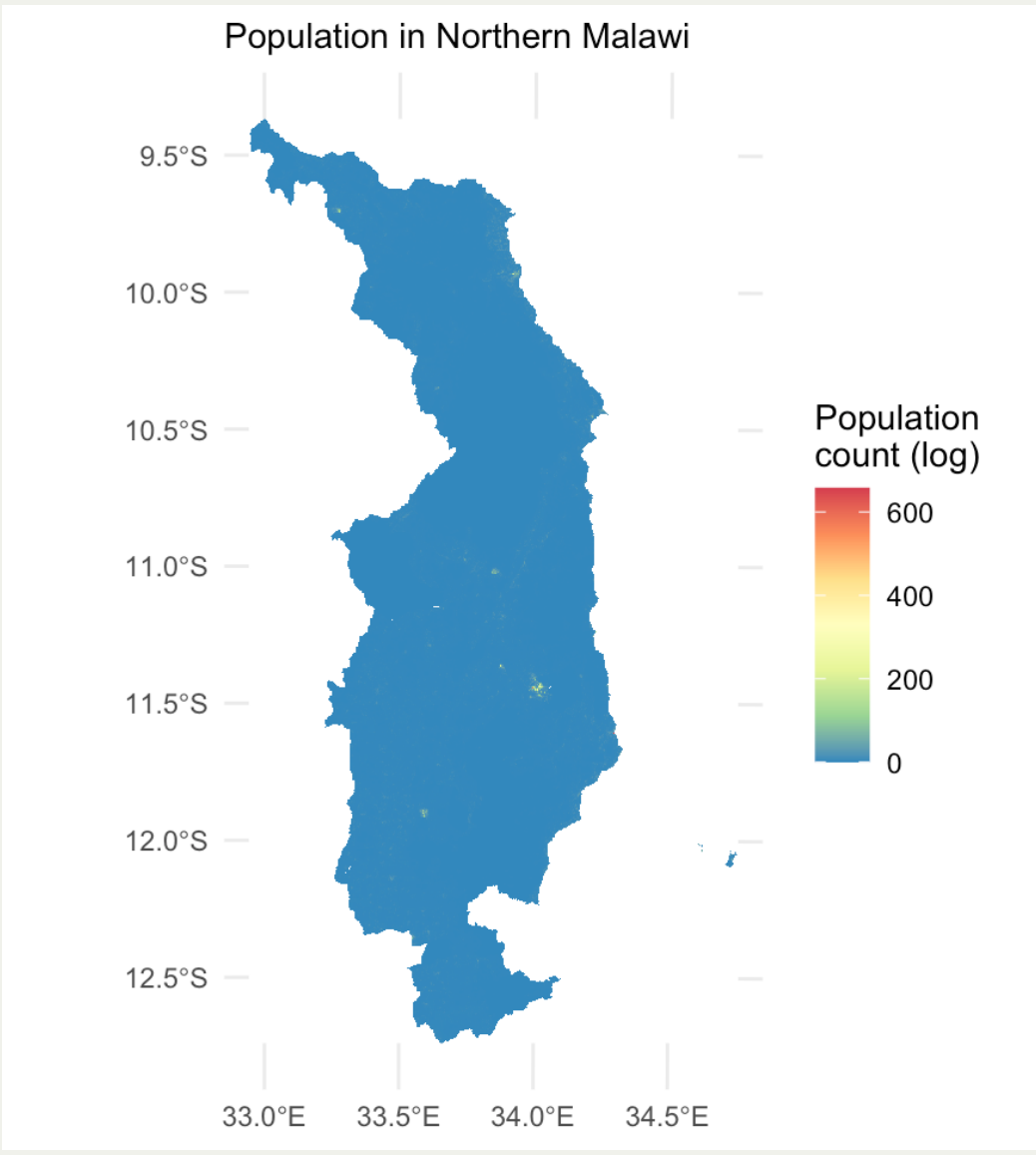
Give it a try

- I actually don't like that map! It's too hard to see because of all the low values.
- So let's take logs, instead!
    - Note that all the zeros become missing (can't log zero)

▼ Code

```
tif <- rast("day4data/mwpop.tif")

ggplot() +
  geom_spatraster(data = log(tif)) +
  scale_fill_distiller("Population\ncount (log)"
    palette = "Spectral", na.value = "white") +
  theme_minimal() +
  labs(subtitle = "Population in Northern Malawi
```

We want to extract the .ti



Let's do it with exactex

▼ Code

```
 1 library(exactextractr)
 2
 3 tif <- rast("day4data/mwpop.tif")
 4 adm4 <- read_sf("day4data/mw4.shp")
 5 # make sure they are in the same CRS! (they already
 6 # st_transform is for the sf object
 7 adm4 <- st_transform(adm4, crs = crs(tif))
 8
 9 # extract the raster values to the shapefile
10 # we are going to SUM, and add the EA_CODE from the
11 extracted <- exact_extract(tif, adm4, fun = "sum", a
```

▼ Code

```
1 head(extracted)
```

```
   EA_CODE       sum
1 10507801 1068.7787
2 10507072  695.8013
3 10507010  938.1139
```

```
4 10507001  749.6960
5 10507009  597.5432
6 10507033  474.3934
```

Now we can join the extr

▼ Code

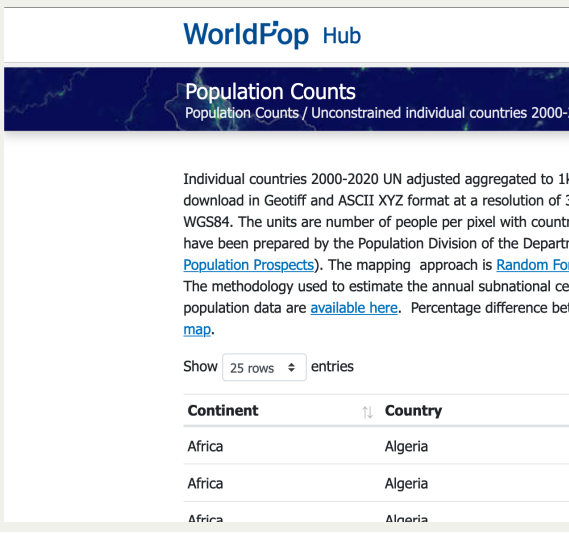
```r
# join
adm4 <- adm4 |>
  left_join(extracted, by = "EA_CODE")

# plot it!
ggplot() +
  geom_sf(data = adm4, aes(fill = sum),
    color = "black", lwd = 0.01) +
  scale_fill_distiller("Population\ncount",
    palette = "Spectral", na.value = "white") +
  theme_minimal() +
  labs(subtitle = "Population in EAs")
```

Now it’s your turn

- Here’s your task:
  - Search for “worldpop population counts”
    - Should be the first result (link: https://h
  - Scroll down the page, click on “unconstra
    resolution)

Now it’s your turn

- Here’s your task:
  - Search for “worldpop population counts”
    - Should be the first result (link: https://h
  - Scroll down the page, click on “unconstra
    resolution)
  - Then, search for a country (maybe yours?



Now it’s your turn

- Here’s your task:
  - Search for “worldpop population counts”
    - Should be the first result (link: https://h
  - Scroll down the page, click on “unconstra
    resolution)
  - Then, search for a country (maybe yours?
  - Click on “Data & Resources” for 2020
  - Scroll down to the bottom of the page and

Now it’s your turn

- Load the .tif into R using `terra`
- Plot the raster using `tidyterra` and `ggplot`
    - Make it look nice!

Let’s keep going!

- Now you need to find a shapefile for the same
- This will be a bit less straightforward
  - Search for “shapefile COUNTRY humdata
  - You should find a link to the Humanitarian
  - Click on it and see if it has shapefiles for y
  - If so, download a shapefile (it can be at a
    - If not, raise your hand and I’ll come he
  - Load it into R and plot it!

One last thing

- You have the population tif and the shapefile
- Extract the population data (using sum, don't
  - Use `append_cols` and make sure you c
- Join the data to the shapefile
- Plot the shapefile with the population data
  - Make it look nice!

What can you do with tha

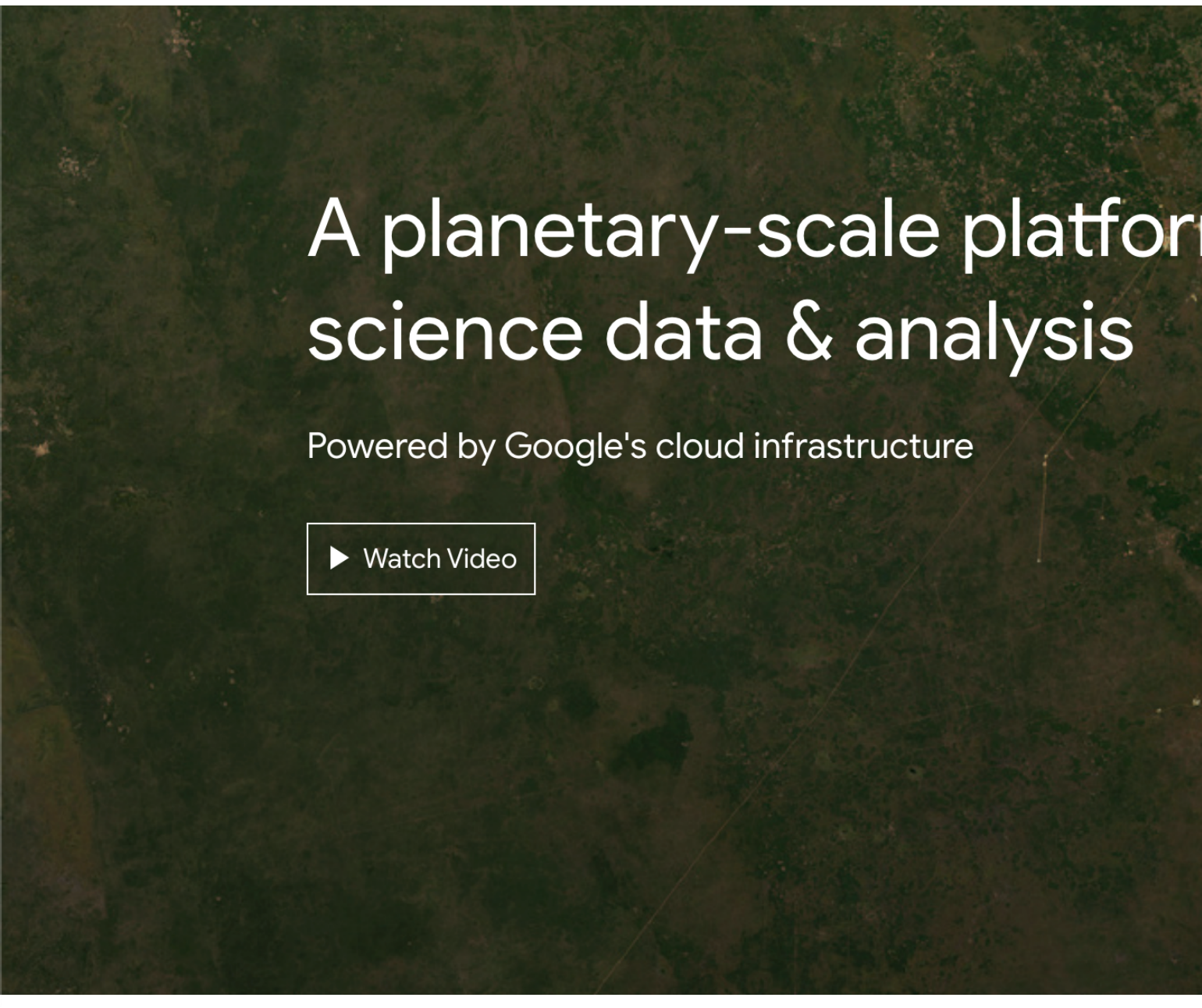
- Now you have a shapefile with population dat
- You can save it as a `.csv` and use it in your a
  - We'll get to this point eventually.
  - We will also discuss adding the survey da

Finding rasters

Where can you find raste

- Depending on the variable, rasters are somet
  - We already saw one example: WorldPop (

- There are two large online repositories:
  - Google Earth Engine
  - Microsoft Planetary Computer
    - This one is newer and has less data (f

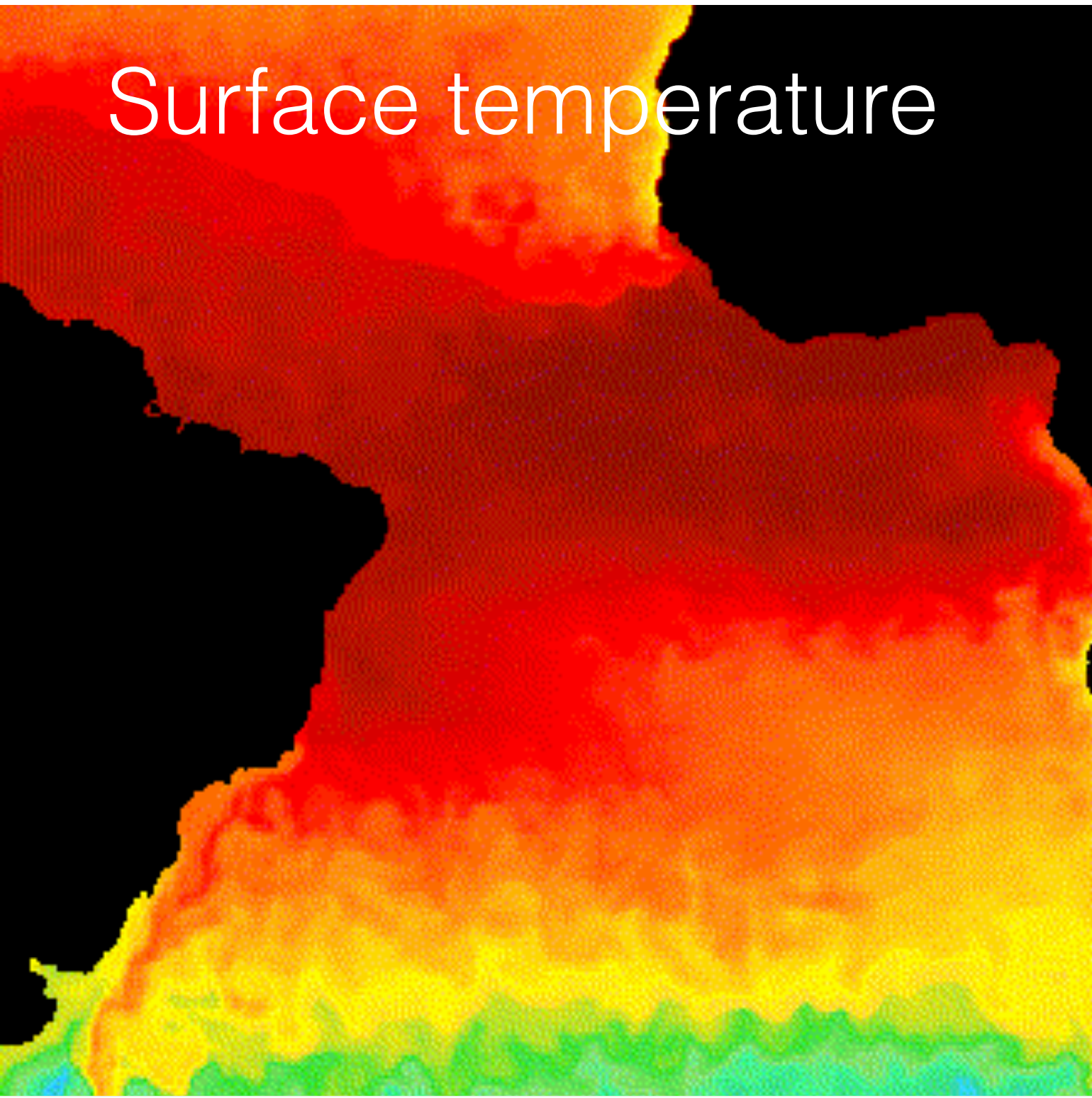

A planetary-scale platfor
science data & analysis
Powered by Google's cloud infrastructure
Watch Video

Meet E

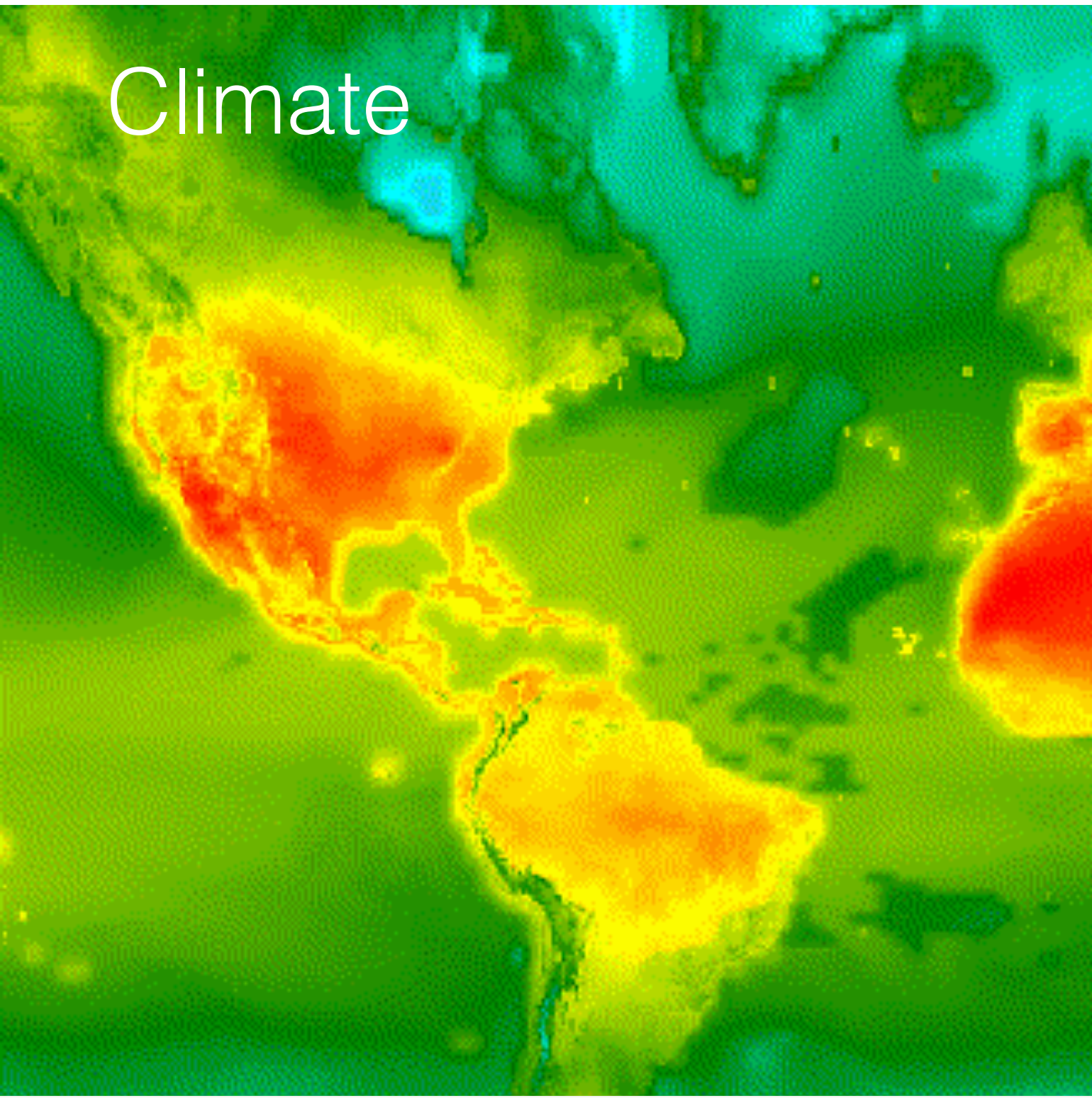
Google Earth Engine

- Google Earth Engine has *a lot* of data.
- Let's see some examples

Surface temperature



Climate



Land cover

Imagery

Google Earth Engine

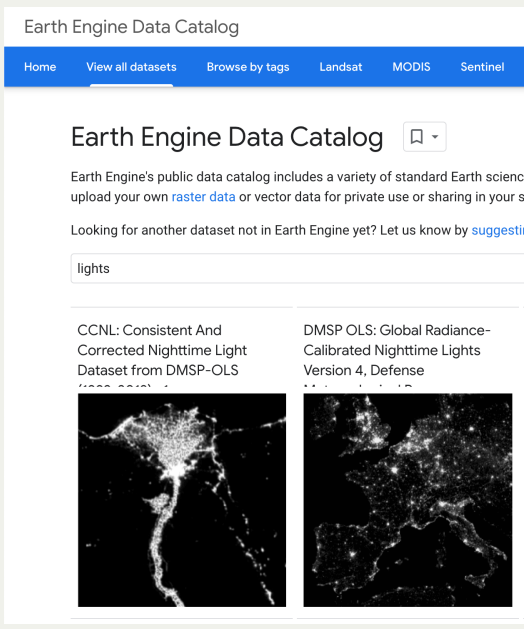
- First things first: you need an account!

- Go to https://earthengine.google.com/ and sig
  - Top-right corner: `Get Started`
  - Next page: `Create account`

- I'll give you all a couple minutes to get this se

Let's look at a dataset

- On the https://earthengine.google.com/ page
  - Click `View all datasets` (near the top
  - Search for `lights`
  - We want `VIIRS Nighttime Day/Nigh`



Basic information about

Earth Engine Data Catalog

Home | View all datasets | Browse by tags | Landsat | MODIS | Sentinel | Publisher

VIIRS Nighttime Day/Night Annual Band Com

Dataset Availability

2012-04-01T00:00:00Z–2021-01-01T00:0

Dataset Provider

Earth Observation Group, Payne Institute f

Earth Engine Snippet

```
ee.ImageCollection("NOAA/VIIRS/DNB
```

Tags

annual | dnb | eog | lights | nighttim

Raster bands - They can

Description | Bands | Terms of Use | Citations | DOIs

Resolution
463.83 meters

Bands

Name	Units	Description
`average`	nanoWatts/sr/cm^2	Average DNB radiance values.
`average_masked`	nanoWatts/sr/cm^2	Average Masked DNB radiance values
`cf_cvg`		Cloud-free coverages; the total number of observations quality is reduced.
`cvg`		Total number of observations free of sunlight and moo
`maximum`	nanoWatts/sr/cm^2	Maximum DNB radiance values.
`median`	nanoWatts/sr/cm^2	Median DNB radiance values
`median_masked`	nanoWatts/sr/cm^2	Median masked DNB radiance values.
`minimum`	nanoWatts/sr/cm^2	Minimum DNB radiance values

Downloading the data is

- Actually getting the data is a bit of a pain
  - Unless you know Javascript!

- A lot of people use libraries in R (or python) to
  - All of them are a bit cumbersome.
    - Especially true in R, because we need
  - We are going to use `rgeedim`
  - Go ahead and install it using `install.p`
  - Load it using `library(rgeedim)`
    - Type `Yes` when asked to create a def

The code

▼ Code

```
1 library(rgeedim)
2 gd_install()
3 gd_authenticate(auth_mode = "notebook")
```

- After `gd_authenticate()`, your browser sh
  - You'll need to sign in to your Google acco
  - Continue through the prompts and make s

The code

▼ Code

```
1 library(rgeedim)
2 gd_install()
3 gd_authenticate(auth_mode = "notebook")
```

- After `gd_authenticate()`, your browser sh
  - You'll need to sign in to your Google acco
  - Continue through the prompts and make s

The code

- You'll arrive at this page.
- Click the **Copy** button

▼ Code

```
1 library(rgeedim)
2 gd_install()
3 gd_authenticate(auth_mode = "notebook")
```

- Then go back to RStudio, and paste (ctrl + v) the code into the console

The code

▼ Code

```
1 library(rgeedim)
2 #gd_install() # You SHOULD NOT need to do this on ea
3 gd_authenticate(auth_mode = "notebook") # need to do
4 gd_initialize() # and you need to do this
```

- After you do `gd_install()` once, you shoul
  - You will need to do `gd_authenticate(` session

Downloading the data - s

- First, we need to create a “bounding box”
  - This is the area of the globe we want to se
  - We will use the Malawi shapefile for this
  - The “bounding box” is a rectangle that co

▼ Code

```
# load shapefile
malawi <- read_sf("day4data/mw4.shp")
# this creates the bounding box
bbox <- st_bbox(malawi)
bbox
```

```
     xmin       ymin       xmax       ymax
32.942417 -12.740578  34.758878  -9.367346
```

Basic information about

- Remember I said we'd need this again?

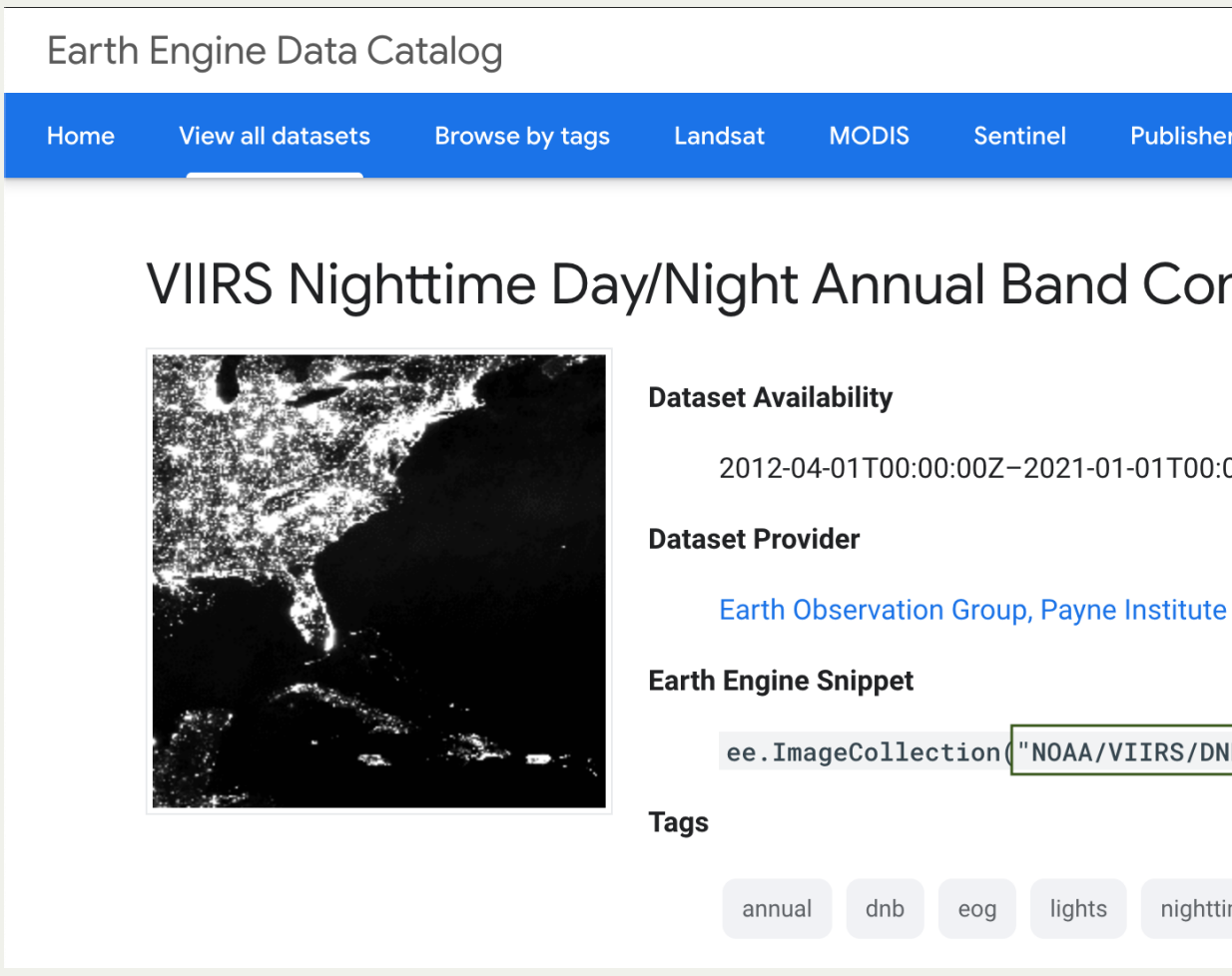


Image collections vs. ima

- One key thing to understand about GEE is the

- An image collection is what it sounds like: a c
  - The key is that we won't download image
  - We'll download individual images
  - So we need to find the images!

Get images from the coll

▼ Code

```
1 x <- gd_collection_from_name("NOAA/VIIRS/DNB/ANNUAL_
2   gd_search(region = bbox)
3 gd_properties(x)
```

```
                                    id                 date
1 NOAA/VIIRS/DNB/ANNUAL_V21/20130101 2013-01-01 09:00:00
2 NOAA/VIIRS/DNB/ANNUAL_V21/20140101 2014-01-01 09:00:00
3 NOAA/VIIRS/DNB/ANNUAL_V21/20150101 2015-01-01 09:00:00
4 NOAA/VIIRS/DNB/ANNUAL_V21/20160101 2016-01-01 09:00:00
5 NOAA/VIIRS/DNB/ANNUAL_V21/20170101 2017-01-01 09:00:00
6 NOAA/VIIRS/DNB/ANNUAL_V21/20180101 2018-01-01 09:00:00
7 NOAA/VIIRS/DNB/ANNUAL_V21/20190101 2019-01-01 09:00:00
8 NOAA/VIIRS/DNB/ANNUAL_V21/20200101 2020-01-01 09:00:00
9 NOAA/VIIRS/DNB/ANNUAL_V21/20210101 2021-01-01 09:00:00
```

- The survey data I have from Malawi is 2019/2
  - We want to use the id: **NOAA/VIIRS/DNB**

We can FINALLY downlo

▼ Code

```
x <- gd_image_from_id("NOAA/VIIRS/DNB/ANNUAL_V21/201
  gd_download(
    filename = "temp.tif",
    region = bbox, # region is our bbox
    scale = 500, # resolution of raster is only 500,
    crs = 'EPSG:4326', # lat/lon
    overwrite = TRUE, # overwrite if it exists
    silent = FALSE
  )
# we downloaded the raster and called it x
# so let's load it using terra!
x <- rast(x)
# here it is!
x
```

```
class       : SpatRaster
dimensions  : 752, 405, 9  (nrow, ncol, nlyr)
resolution  : 0.004491576, 0.004491576  (x, y)
extent      : 32.94122, 34.76031, -12.7426, -9.364937  (xmin, xmax
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : temp.tif
names       : average, avera~asked, cf cvg, cvg, maximum, median,
```

Quick note: we downloa

▼ Code

```
names(x)
```

```
[1] "average"        "average_masked" "cf_cvg"
```

▼ Code

```
# but we really only want average nightlights
# so here's how you can download just the average
x <- gd_image_from_id("NOAA/VIIRS/DNB/ANNUAL_V21/201
  gd_download(
    filename = "temp.tif",
    region = bbox, # region is our bbox
    scale = 500, # resolution of raster is only 500,
    crs = 'EPSG:4326', # lat/lon
    overwrite = TRUE, # overwrite if it exists
    silent = FALSE,
    bands = list("average"),
  )
# we downloaded the raster and called it x
# so let's load it using terra!
x <- rast(x)
```

```
# here it is!
x
```

```
class       : SpatRaster
dimensions  : 752, 405, 1  (nrow, ncol, nlyr)
resolution  : 0.004491576, 0.004491576  (x, y)
extent      : 32.94122, 34.76031, -12.7426, -9.364937
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : temp.tif
name        : average
```

What does it look like?

▼ Code

```
adm4 <- read_sf("day4data/mw4.shp")
ggplot() +
  geom_spatraster(data = x) +
  scale_fill_distiller("Nightlights",
    palette = "Spectral") +
  geom_sf(data = adm4,
    color = "white",
    lwd = 0.01,
    alpha = 0.5,
    fill = "transparent") +
  theme_minimal() +
  labs(subtitle = "Nightlights in Malawi")
```

- Note what the “bounding box” does!
  - `st_bbox` as a reminder

Now it’s your turn!

- We want to download NDVI data for Malawi
  - We want the `Terra Vegetation Indi` this)
  - Download the first observation from 2019