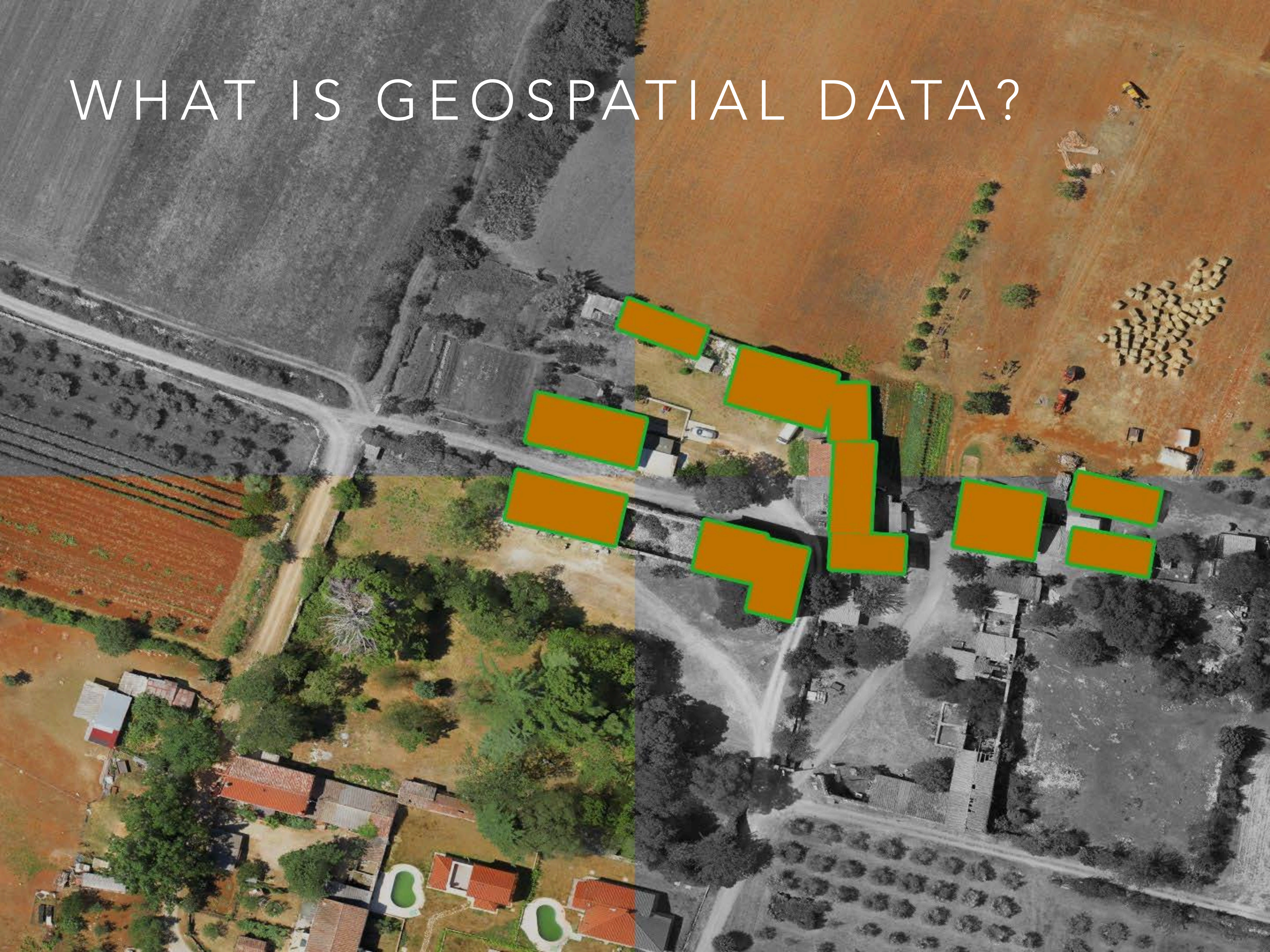
ANT 6973: DATA VISUALIZATION AND EXPLORATION

VISUALIZING GEOSPATIAL DATA

WHAT IS GEOSPATIAL DATA?



WHAT IS GEOSPATIAL DATA?

- A simple definition: you can show it on a map
- Geospatial data have *spatial referents* (e.g., lat/long coordinates) that anchor them to precise locations on the Earth's surface

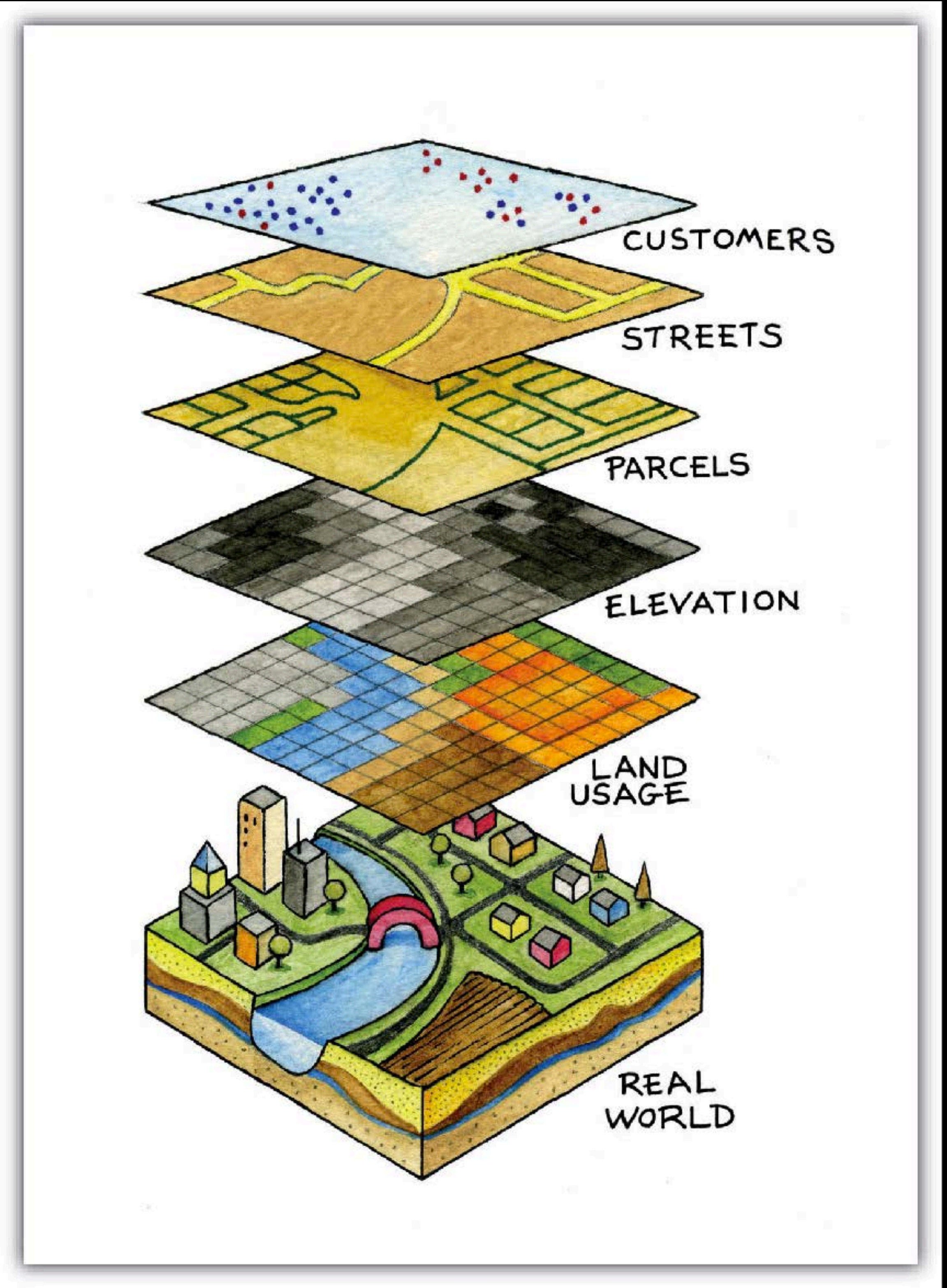
SPATIAL DATA IN R

- ggplot2 has robust spatial plotting capabilities.
- Advantages are much the same as with other kinds of plots that we have been making.
    - Draw maps programmatically.
    - Elements of a map can be added or removed with a few keystrokes.
    - Easy to reproduce with different data sets.
    - Same ggplot2 syntax that is now familiar.

GEOSPATIAL DATA MODEL

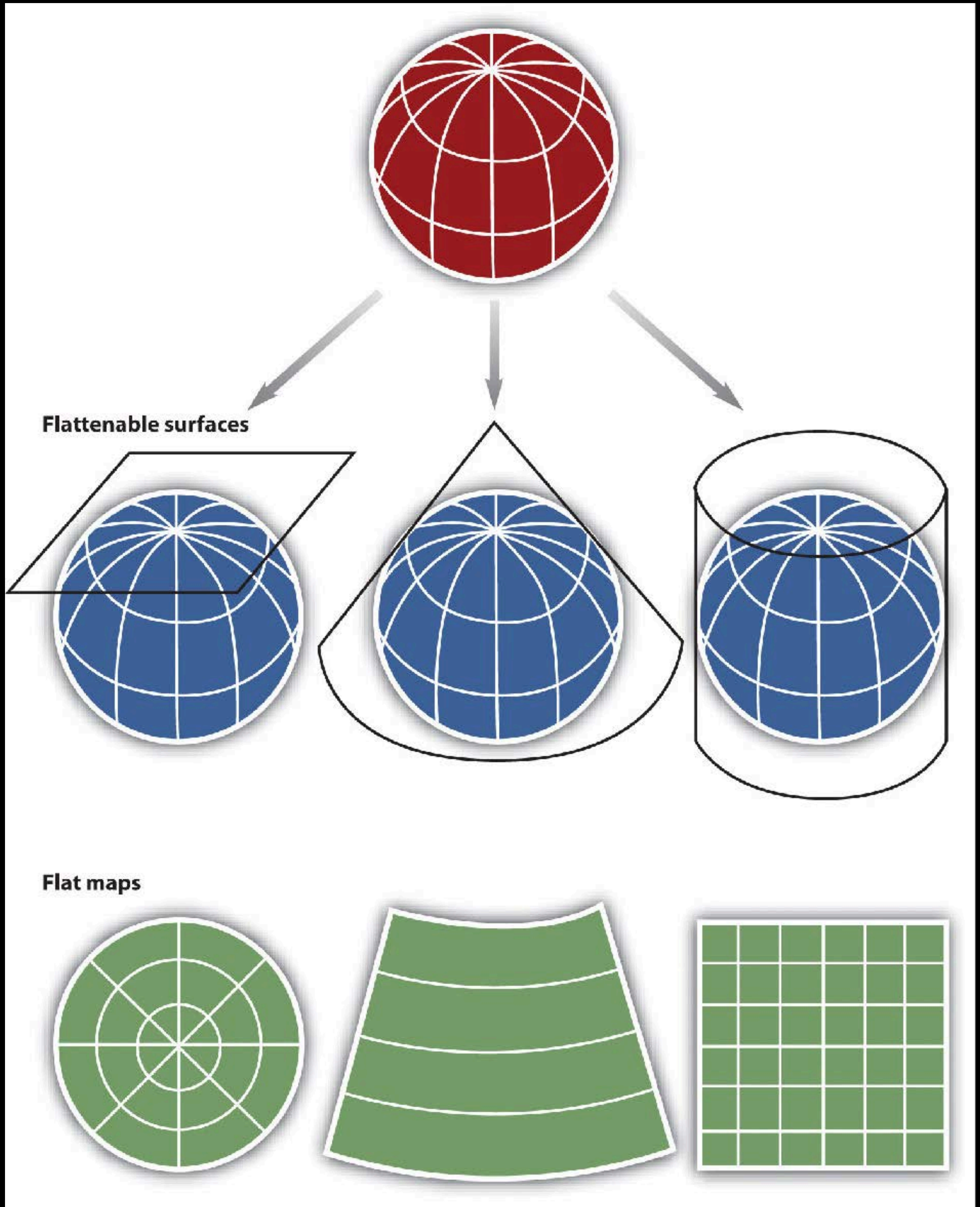
GEOSPATIAL DATA MODEL: IMPLEMENTATION

- Data are organized by layers, with each layer representing a common feature.
- Layers are integrated using explicit location on the earth's surface, thus geographic location is the organizing principal.



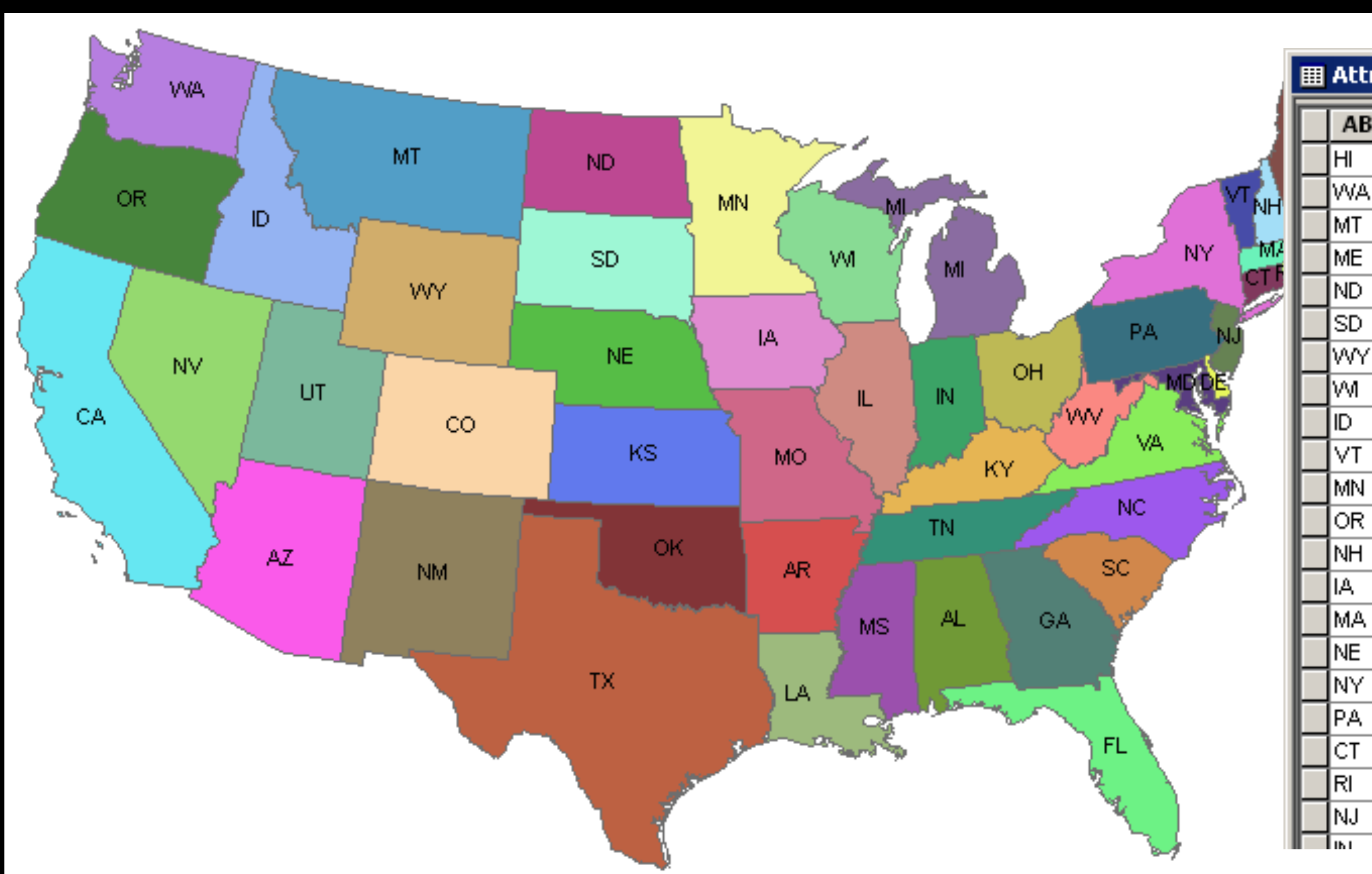
COORDINATE REFERENCE SYSTEMS

- Datum: tells where something is located on earth's surface
- Projection: tells how to represent the curved 3-D surface of the earth by X,Y coordinates on a 2-D flat map/screen
- Distortion is inevitable with any projection



GEOSPATIAL DATA COMPONENTS

- Layers consist of two data types
  - Spatial data that describe location (where)
  - Attribute data specifying what, how much, and when

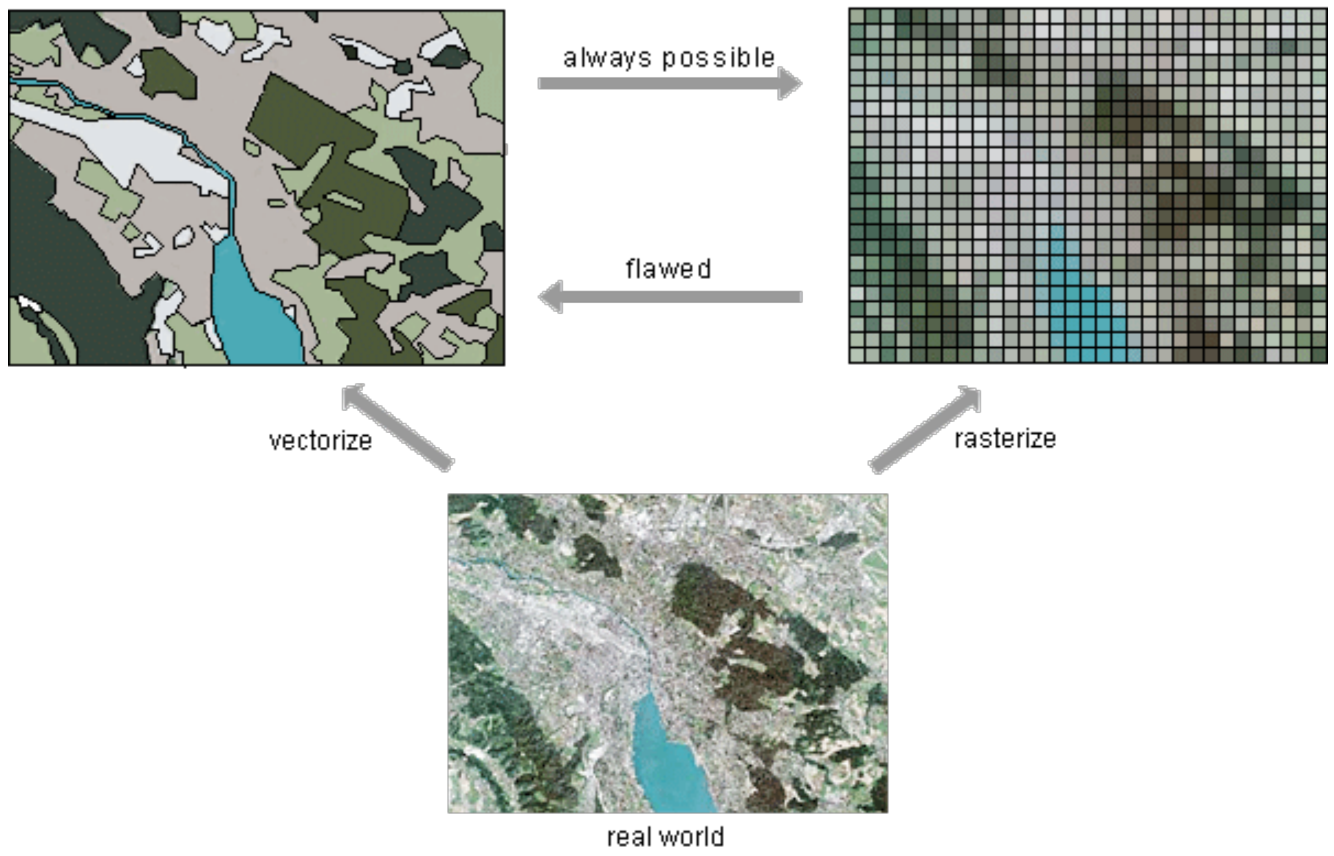


Attributes of States_Table

ABBR	NAME	AREA	SUB_REGI_1	POP1990	POP2000_1
HI	Hawaii	6428.217	Pacific	1108229	1184688
WA	Washington	67566.127	Pacific	4866692	5835089
MT	Montana	147043.116	Mtn	799065	885795
ME	Maine	32495.312	N Eng	1227928	1257219
ND	North Dakota	70700.125	W N Cen	638800	631032
SD	South Dakota	77116.662	W N Cen	696004	734993
WY	Wyoming	97813.807	Mtn	453588	479673
WI	Wisconsin	56050.459	E N Cen	4891769	5277833
ID	Idaho	83570.06	Mtn	1006749	1273309
VT	Vermont	9614.299	N Eng	562758	596714
MN	Minnesota	84383.092	W N Cen	4375099	4820250
OR	Oregon	96954.726	Pacific	2842321	3356108
NH	New Hampshire	9265.998	N Eng	1109252	1215100
IA	Iowa	56271.701	W N Cen	2776755	2877060
MA	Massachusetts	8118.475	N Eng	6016425	6206482
NE	Nebraska	77353.859	W N Cen	1578385	1672199
NY	New York	48623.646	Mid Atl	17990455	18223519
PA	Pennsylvania	45301.263	Mid Atl	11881643	11986139
CT	Connecticut	4975.458	N Eng	3287116	3289062
RI	Rhode Island	1088.882	N Eng	1003464	992011
NJ	New Jersey	7545.009	Mid Atl	7730188	8192386
IN	Indiana	[illegible]	E N Cen	[illegible]	[illegible]

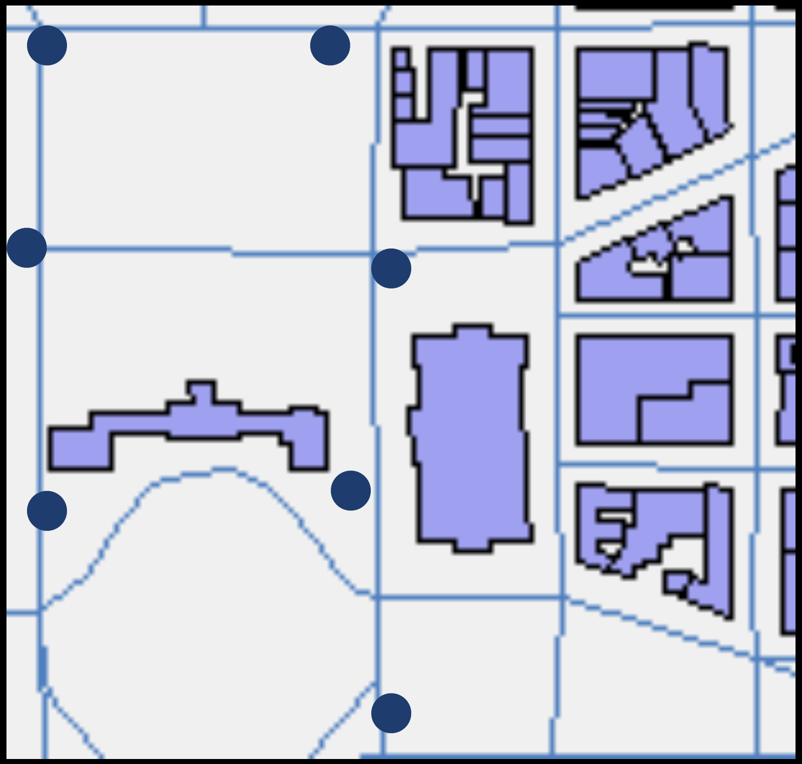
VECTOR DATA VS. RASTER DATA

- Geographic information in the real world may be represented in two ways
    - Vector format
    - Raster format

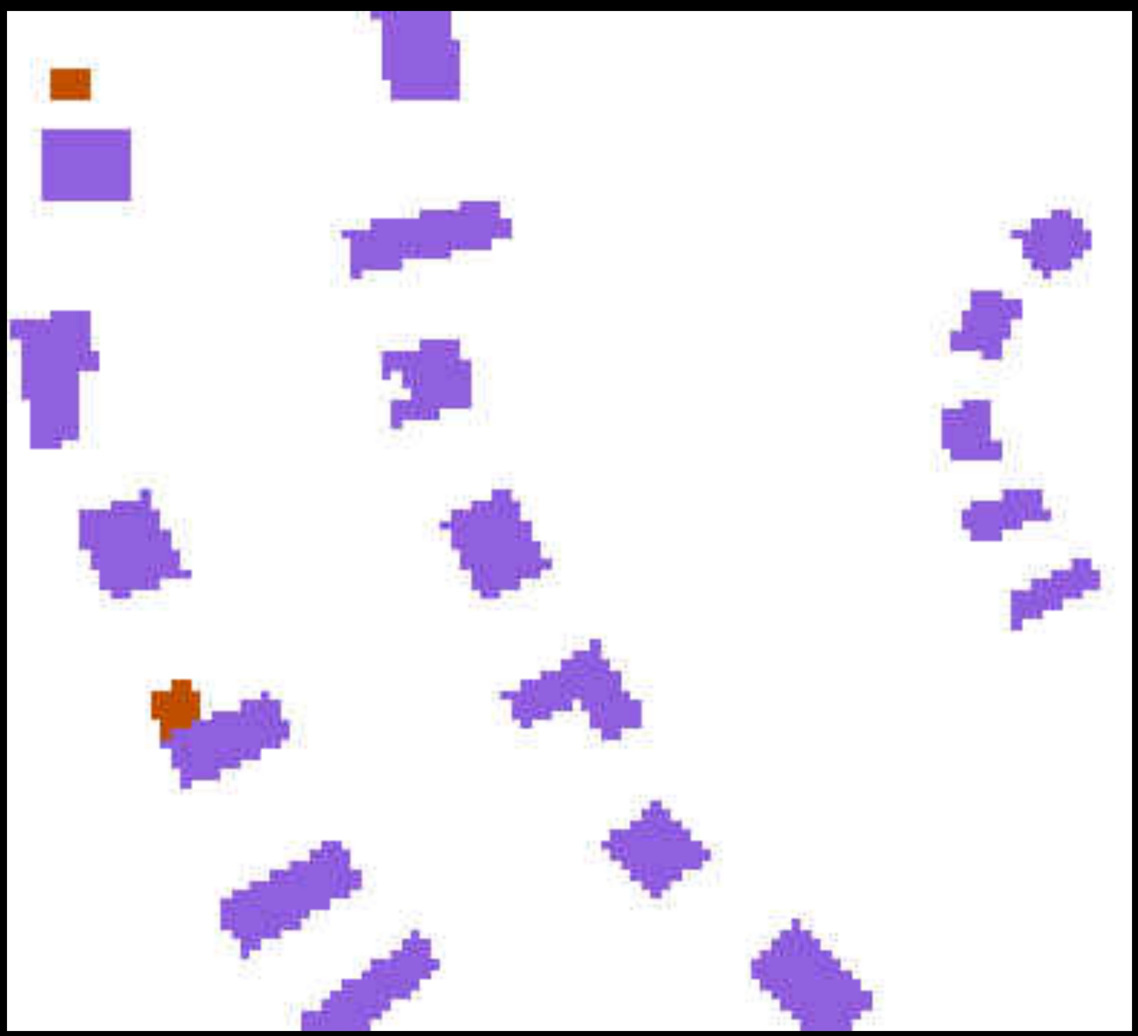


VECTOR DATA VS. RASTER DATA

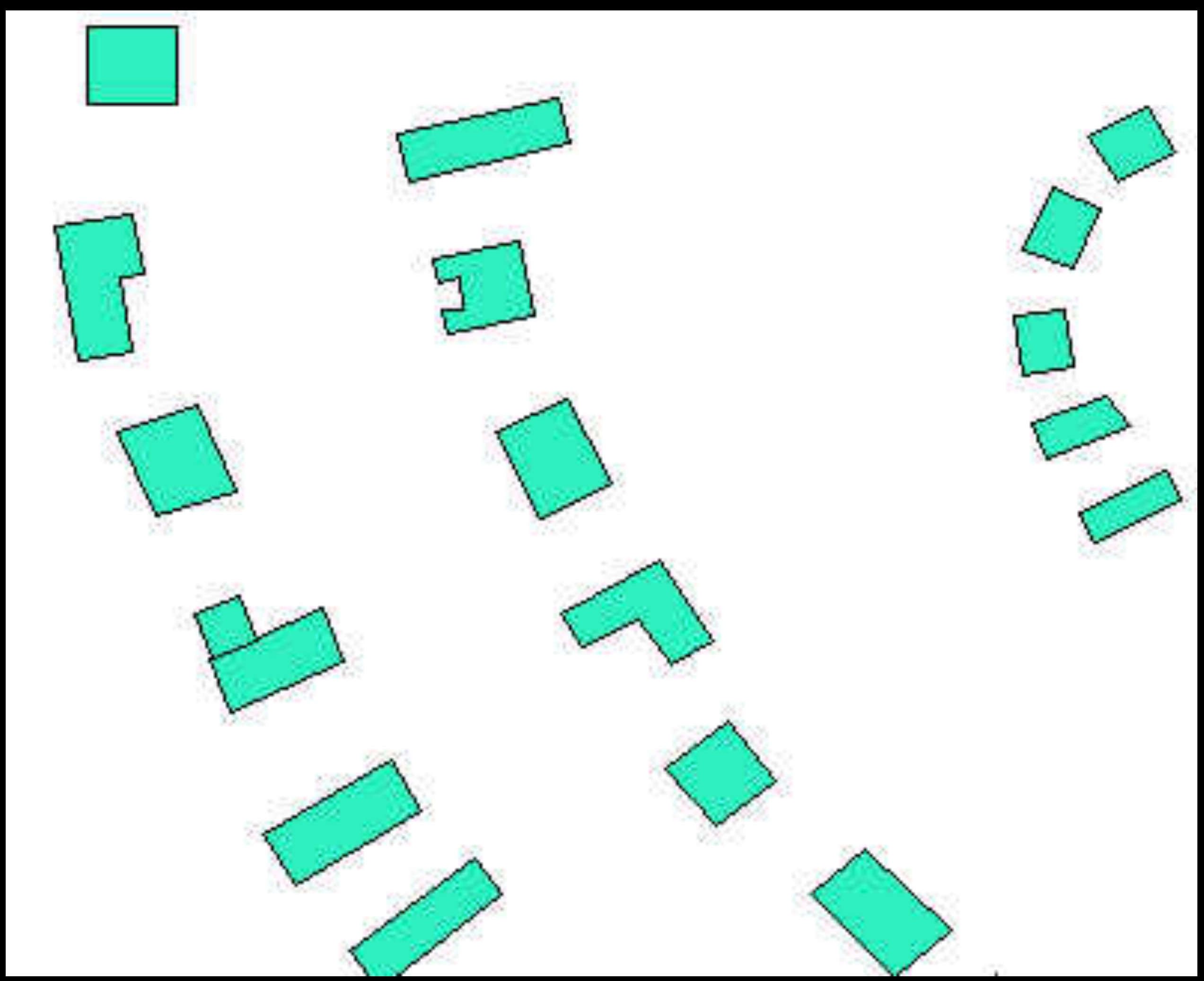
- Vector
  - Ideal for discrete themes with definite boundaries (roads, buildings, political borders)
- Raster
  - Ideal for continuous themes of change (elevation, rainfall)



VECTOR DATA VS. RASTER DATA

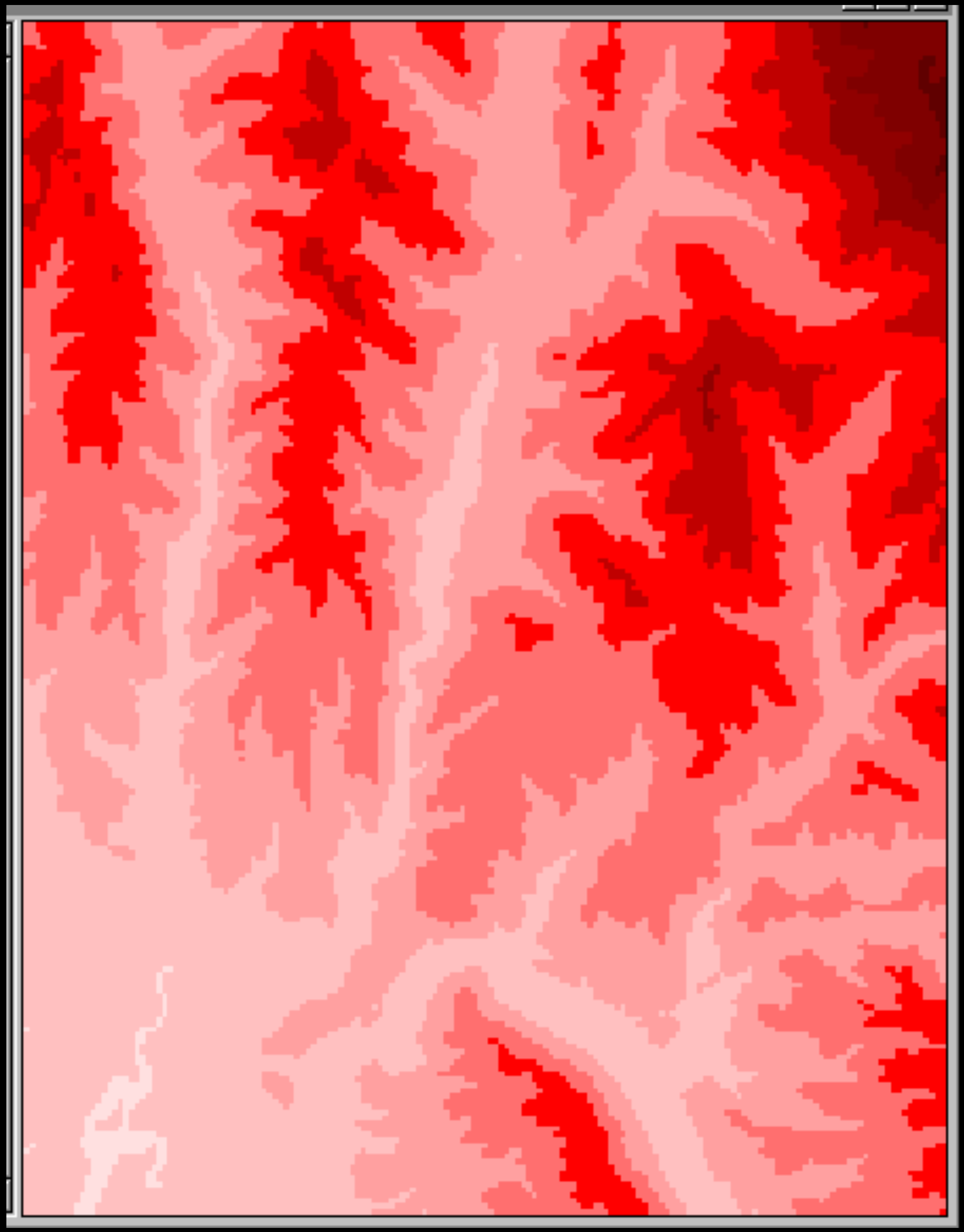


Raster: bad with bounding

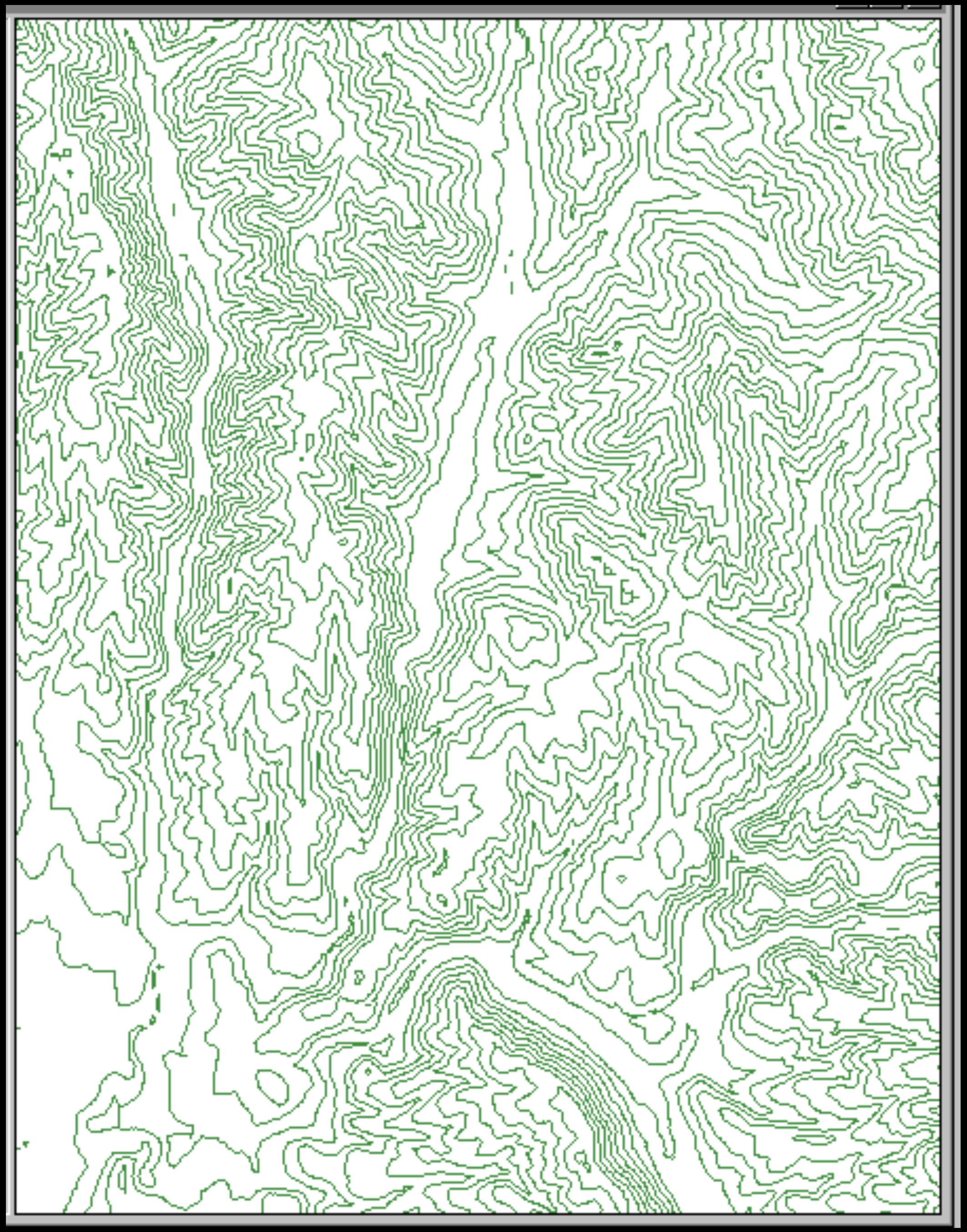


Vector: boundary precision

VECTOR DATA VS. RASTER DATA



Raster: great for surfaces

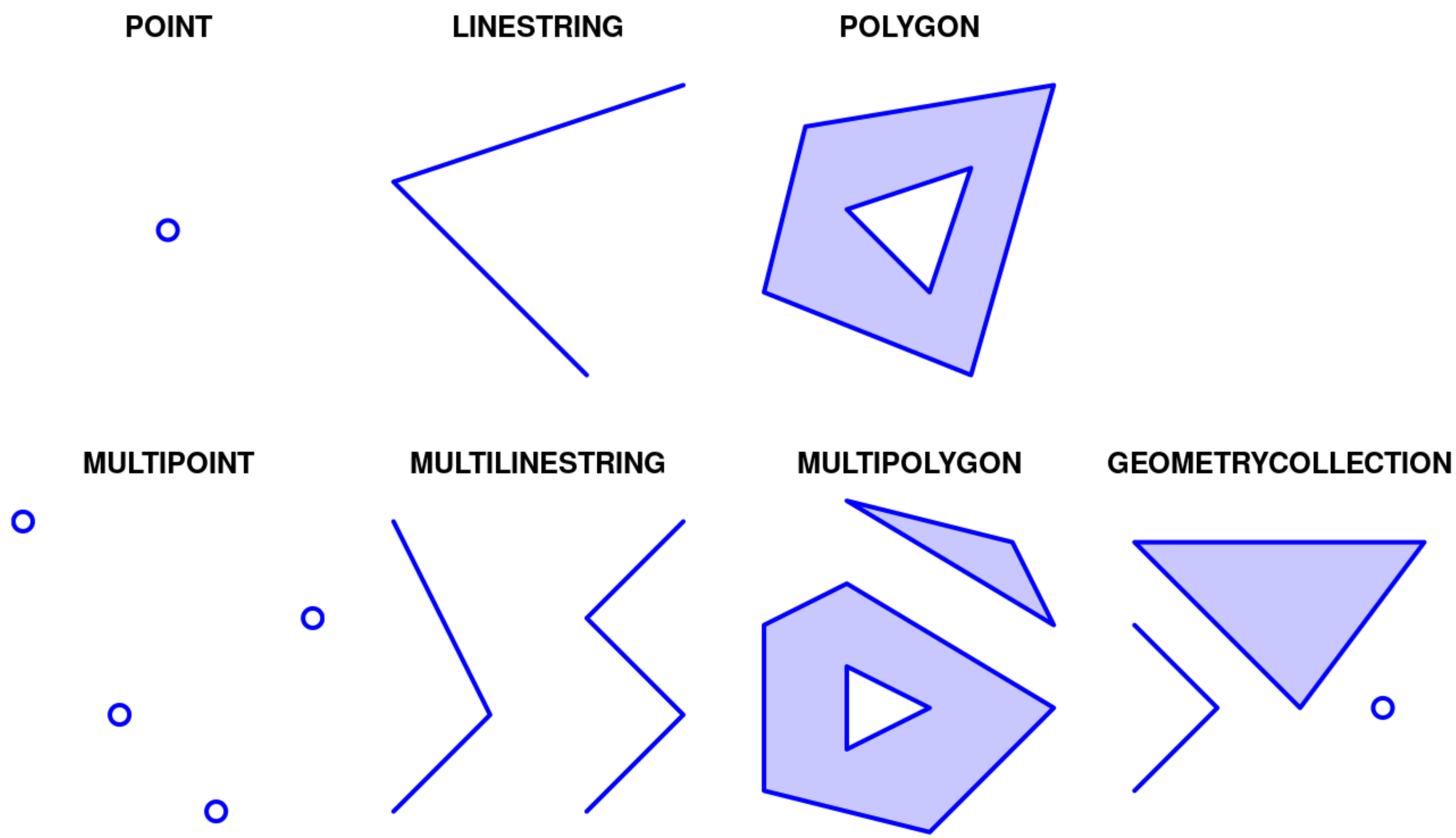


Vector: limited with surfaces

VECTOR DATA LAYERS

VECTOR DATA LAYERS

- Vector objects and include basic geometry types like points, lines, and polygons.

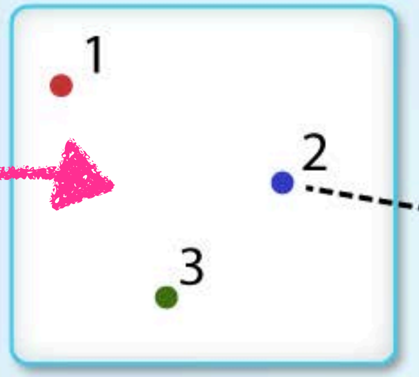


VECTOR DATA LAYERS

Geometries

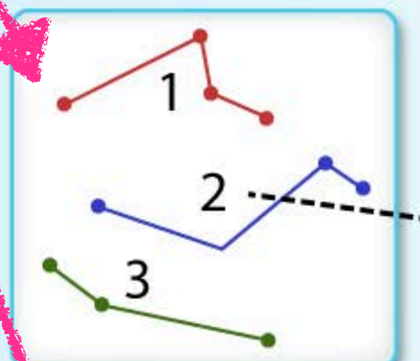
Non-spatial attributes

Example Attributes for Point Data



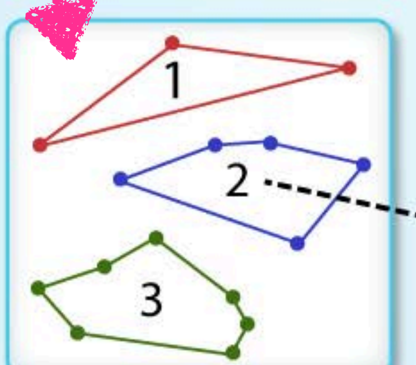
ID	Plot Size	Type	VegClass
1	40	Vegetation	Conifer
2	20	Vegetation	Deciduous
3	40	Vegetation	Conifer

Example Attributes for Line Data



ID	Type	Status	Maintenance
1	Road	Open	Year Round
2	Dirt Trail	Open	Summer
3	Road	Closed	Year Round

Example Attributes for Polygon Data



ID	Type	Class	Status
1	Herbaceous	Grassland	Protected
2	Herbaceous	Pasture	Open
3	Herbaceous / Woody	Grassland	Protected

THE SF PACKAGE

- The sf stands for "simple features."
- Geometric objects stored in tables that include attributes and a "geometry" column for each feature.
- Plays nicely with tidyverse (treated as normal data frame).
- Read/write spatial objects; geocomputation

geoid	name	variable	estimate	moe	geometry
48007	Aransas County, Texas	B19013_001	41690	3678	MULTIPOLYGON (((1811769 712…
48025	Bee County, Texas	B19013_001	42302	3403	MULTIPOLYGON (((1686520 717…
48035	Bosque County, Texas	B19013_001	44674	3329	MULTIPOLYGON (((1688481 754…
48067	Cass County, Texas	B19013_001	37352	2430	MULTIPOLYGON (((1999018 765…
48083	Coleman County, Texas	B19013_001	35156	4158	MULTIPOLYGON (((1526295 749…
48091	Comal County, Texas	B19013_001	65833	3291	MULTIPOLYGON (((1630729 729…

THE SF PACKAGE

- sf spatial objects can be plotted in ggplot2 like normal `geom_` layers using the `geom_sf()` function.
    - Don't need to map x and y (it's taken from the geometry)
- `geom_sf()` is an unusual geom because it will draw different geometric objects depending on what simple features are present in the data
    - Can draw points, lines, or polygons.

ANOTHER USEFUL PACKAGE: RNATURALEARTH

- The rnaturalearth R package makes it easy to download and use free, high quality map data in R.
- Three scales
  - 1:10m – most detailed, suitable for zoomed in maps within countries
  - 1:50m – medium level of detail, suitable for maps of countries or regions.
  - 1:110m – least detailed, suitable for global maps.
- Data come from Natural Earth: https://www.naturalearthdata.com/

Downloads

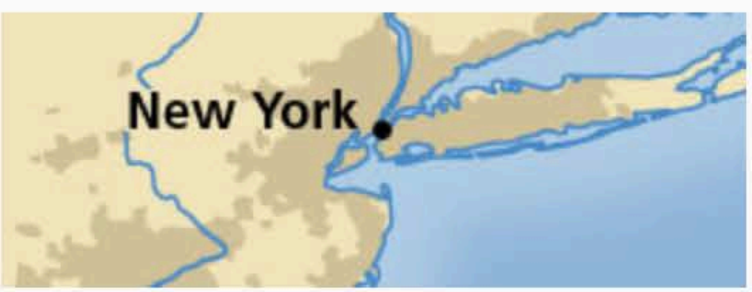
Data themes are available in three levels of detail. For each scale, themes are listed on Cultural, Physical, and Raster category pages.

Stay up to date! Know when a new version of Natural Earth is released by subscribing to our announcement list.

Overwhelmed? The Natural Earth quick start kit (227 mb) provides a small sample of Natural Earth themes styled in an ArcMap .MXD document and a QGIS document. Download all vector themes as SHP (279 mb), SQLite (222 mb), or GeoPackage (260 mb).

Natural Earth is the creation of many volunteers and is supported by NACIS. It is free for use in any type of project. Full Terms of Use »

Large scale data, 1:10m



Cultural | Physical | Raster

The most detailed. Suitable for making zoomed-in maps of countries and regions. Show the world on a large wall poster.

1:10,000,000
1″ = 158 miles
1 cm = 100 km

Medium scale data, 1:50m

Cultural | Physical | Raster

Suitable for making zoomed-out maps of countries and regions. Show the world on a tabloid size page.

1:50,000,000
1″ = 790 miles
1 cm = 500 km

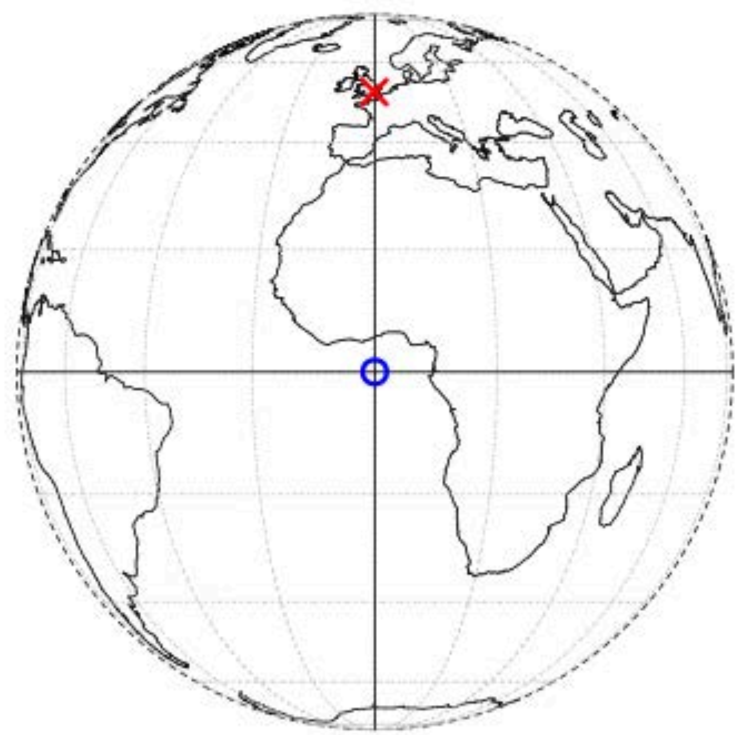
Small scale data, 1:110m

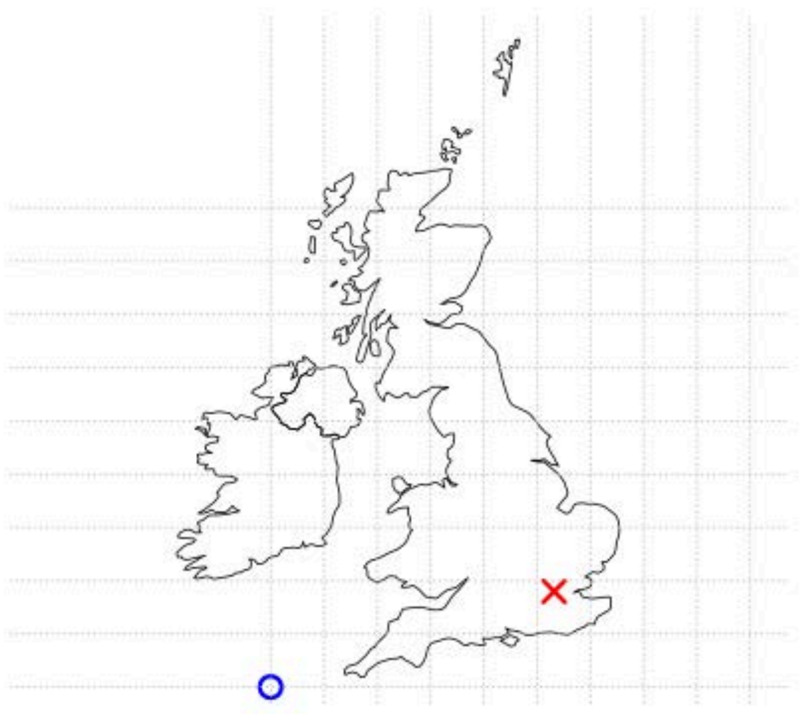
Cultural | Physical

Suitable for schematic maps of the world on a postcard or as a small locator globe.

1:110,000,000
1″ = 1,736 miles
1 cm = 1,100 km

VECTOR DATA FILE FORMATS





Common file formats:

- ESRI Shapefile (.shp)
- GeoJSON (.json)
- Keyhole Markup Language (.kml)
- GPX Exchange Format (.gpx)
- Spatial database: PostGIS / PostgreSQL

YOUR TURN

- Install the `sf` package, if you haven't already.
- Install the packages `rnaturalearth` and `rnaturalearthdata`.
- Create a new R markdown file for this interactive activity (nothing to turn in).
- Load the packages `tidyverse`, `rnaturalearth`, and `sf`

YOUR TURN

- Let's create a map mixing spatial and non-spatial data using the `storms` dataset that comes with `tidyverse`

name	year	month	day	hour	lat	long	status	category	wind	pressure	ts_diameter	hu_diameter
Amy	1975	6	27	0	27.5	-79.0	tropical depression	-1	25	1013	NA	NA
Amy	1975	6	27	6	28.5	-79.0	tropical depression	-1	25	1013	NA	NA
Amy	1975	6	27	12	29.5	-79.0	tropical depression	-1	25	1013	NA	NA
Amy	1975	6	27	18	30.5	-79.0	tropical depression	-1	25	1013	NA	NA
Amy	1975	6	28	0	31.5	-78.8	tropical depression	-1	25	1012	NA	NA
Amy	1975	6	28	6	32.4	-78.7	tropical depression	-1	25	1012	NA	NA

- Specifically, we want to plot the progression of hurricane Katrina (2005), showing its path and windspeed at each reading.

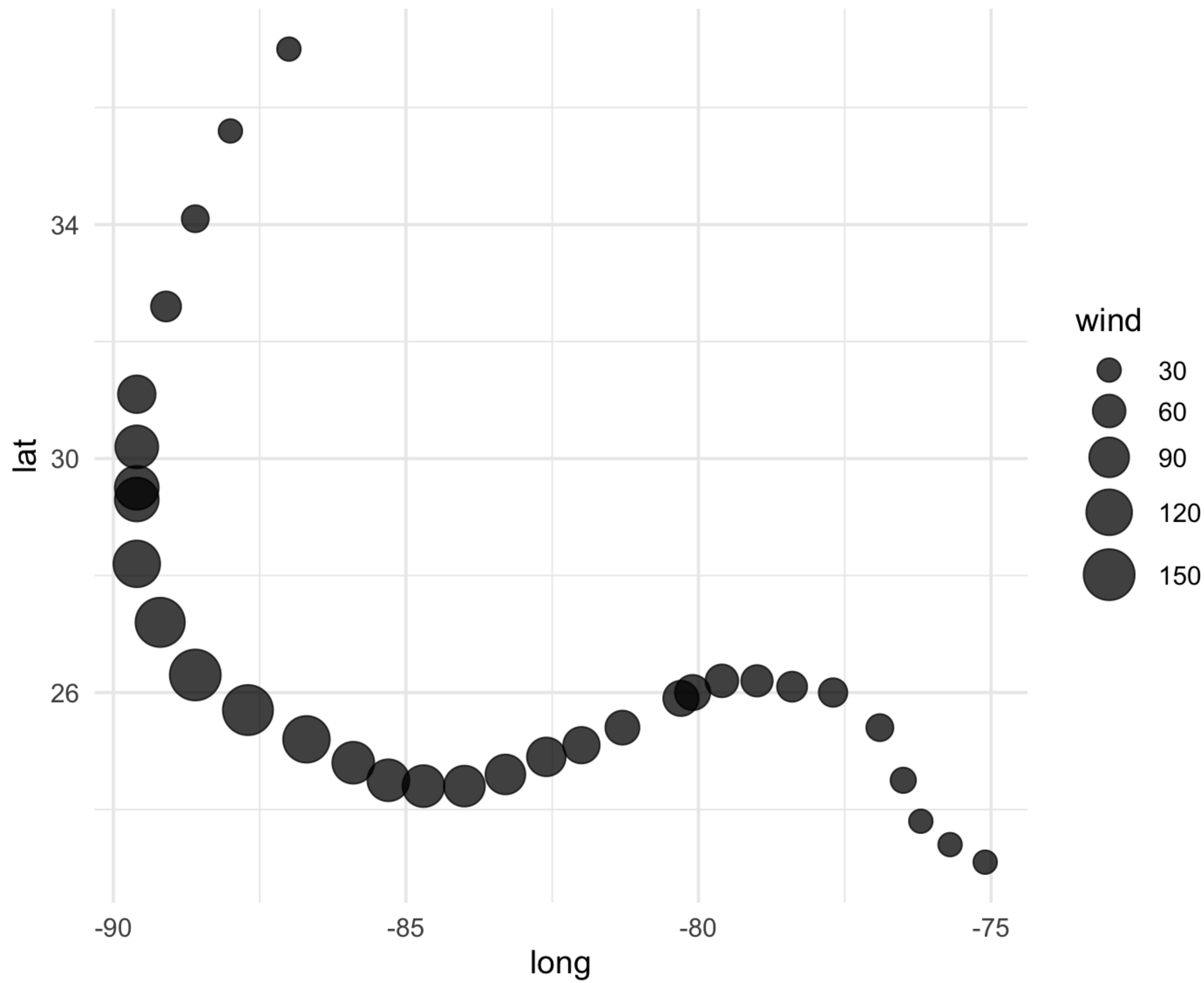
STEP 1: GET THE DATA

- Obtain the data for 2005 hurricane Katrina only

```
katrina <- storms %>%
           filter(name == "Katrina" & year == 2005)
```

STEP 2: BUBBLE CHART

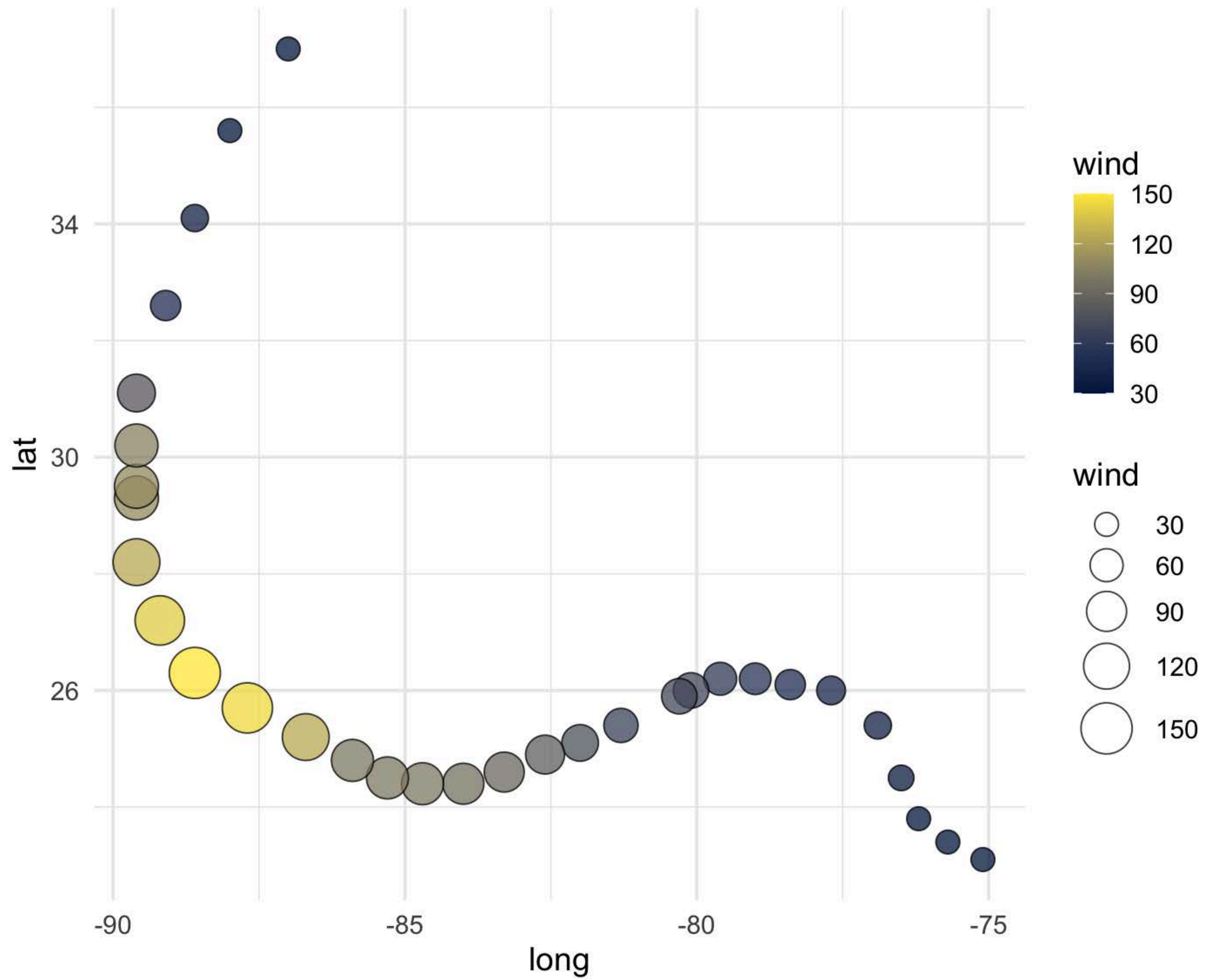
- Create a bubble chart using longitude and latitude for our x-y, and wind for size.
- Make the points partially transparent.
- Use a scale function to make the max point size 10.
- Use `coord_equal()` to make x and y axis equal units.




```r
ggplot(katrina, aes(x = long, y = lat, size = wind)) +
  geom_point(alpha = 0.75) +
  scale_size_area(max_size = 10) +
  coord_equal()
```

STEP 3: ADD COLOR

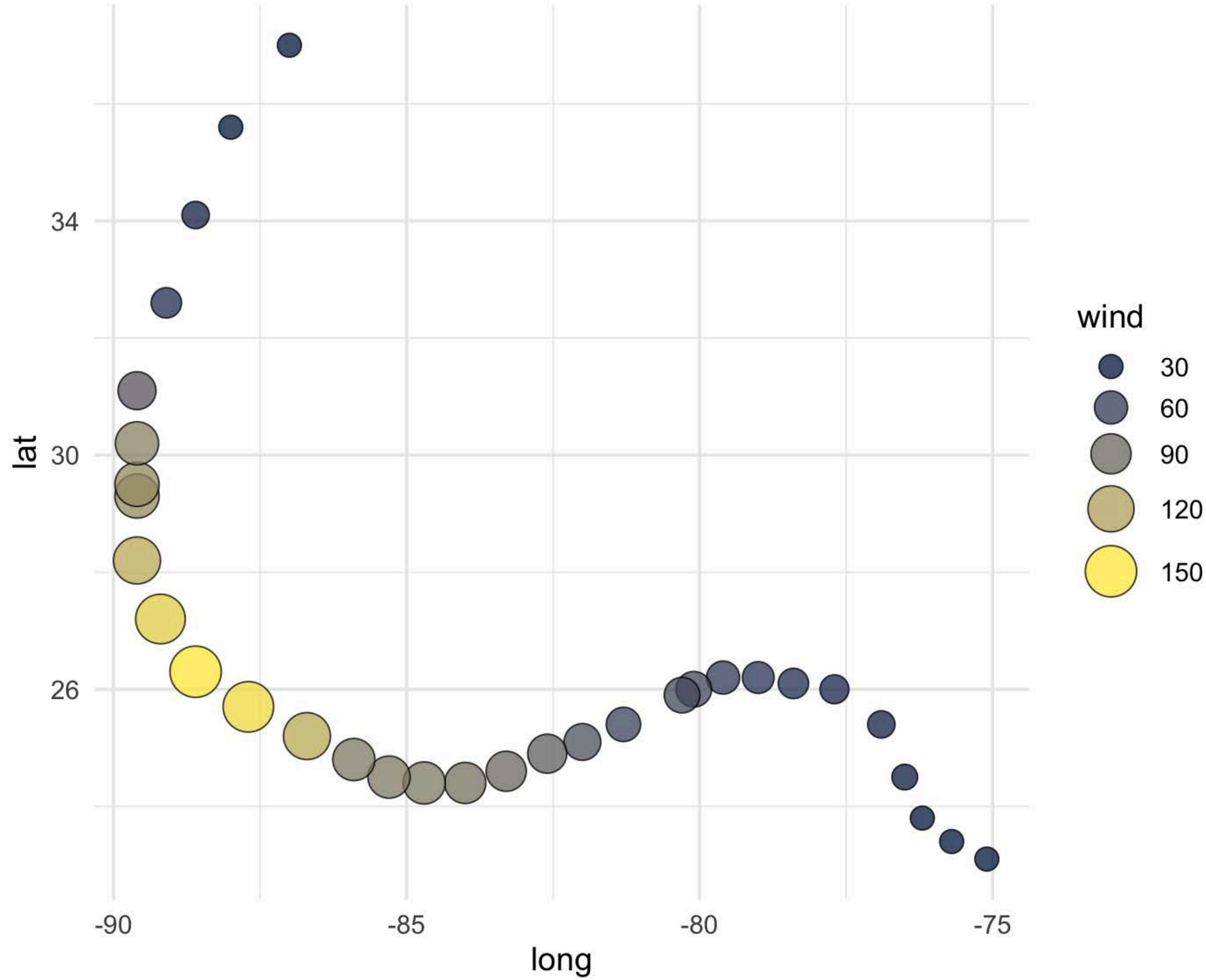
- Use point shape 21
- Map the interior fill to wind speed as well.
- Use the "cividis" palette.




```r
ggplot(katrina, aes(x = long, y = lat, size = wind, fill = wind)) +
  geom_point(alpha = 0.75, shape = 21) +
  scale_fill_viridis_c(option = "cividis") +
  scale_size_area(max_size = 10) +
  coord_equal()
```

STEP 4: COMBINE LEGENDS

- Combine the legend and color bar using the `guides()` function
    - By default, `scale_fill_viridis_c()` uses a colorbar, but we want it to use a legend (like size)




```
ggplot(katrina, aes(x = long, y = lat, size = wind, fill = wind)) +
  geom_point(alpha = 0.75, shape = 21) +
  scale_fill_viridis_c(option = "cividis", guide = "legend") +
  scale_size_area(max_size = 10) +
  coord_equal()
```

STEP 5: OBTAIN MAP DATA

- Load the rnaturalearth package and, when prompted, download the data.

- Maps include:

  1. `ne_countries()` for country boundaries
  2. `ne_states()` for boundaries within countries
  3. `ne_coastline()` for world coastline

Type of object to return

Medium resolution

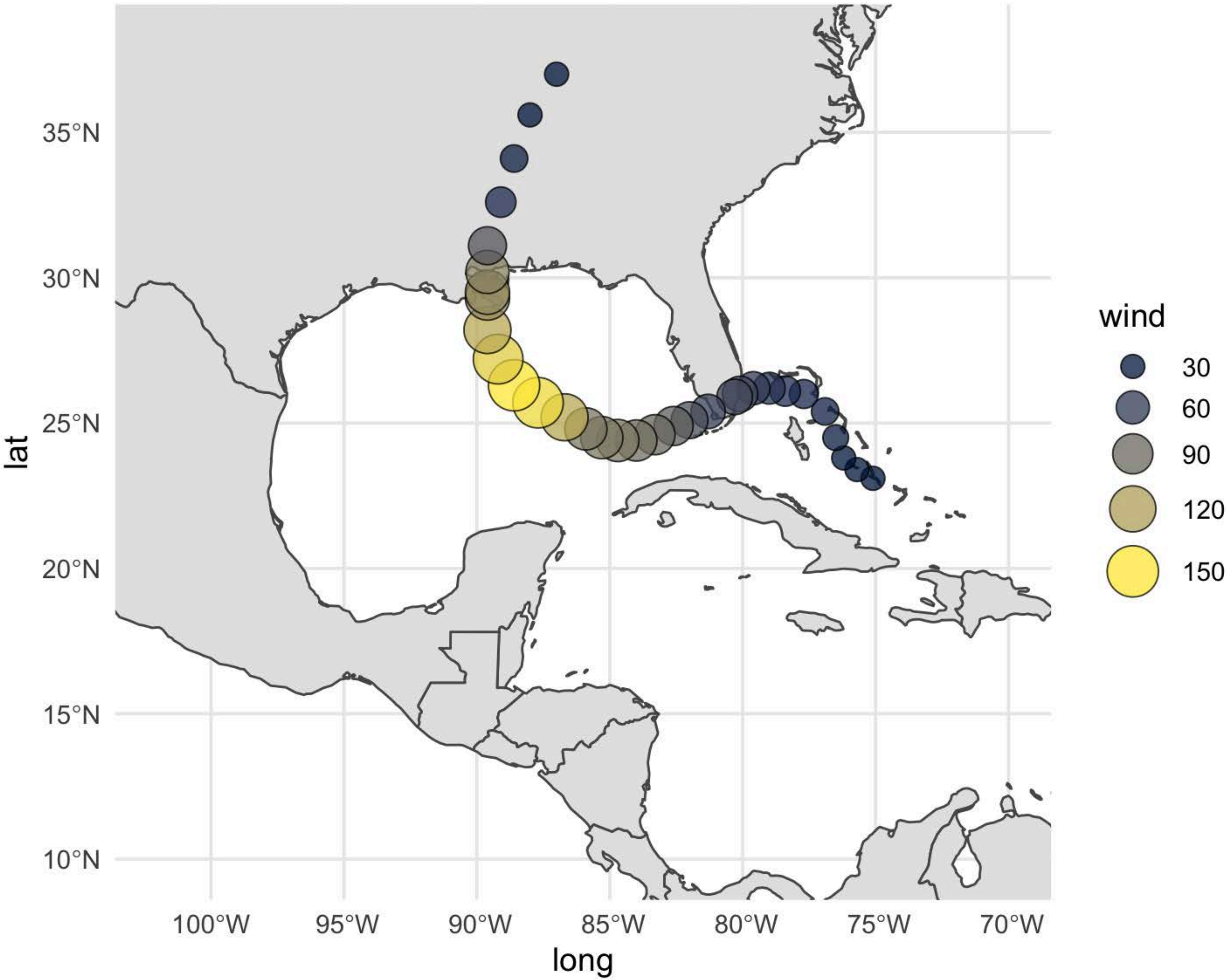
```
world <- ne_countries(scale = "medium", returnclass = "sf")
```

STEP 6: ADD THE MAP

- `geom_sf(data = world)` beneath other layers.
- `coord_sf()` function allows us to "crop" the world map to our area of interest, and provides nice lat/long formatting.

Because we're now mixing datasets, provide data at geom level

```
ggplot() +
  geom_sf(data = world) +
  geom_point(data = katrina,
             aes(x = long, y = lat, size = wind, fill = wind),
             alpha = 0.75, shape = 21) +
  scale_fill_viridis_c(option = "cividis", guide = "legend") +
  scale_size_area(max_size = 10) +
  coord_sf(xlim = c(-102, -70), ylim = c(10, 38))
```



35°N
30°N
25°N
20°N
15°N
10°N
lat
100°W
95°W
90°W
85°W
80°W
75°W
70°W
long
wind
30
60
90
120
150

STEP 7: THEME TWEAKS

- We have not talked much about customizing the built-in themes.

- But the appearance of just about every non-data element of the plot can be customized using the `theme()` function.

- Lots of examples in the online documentation: https://ggplot2.tidyverse.org/reference/theme.html

- Warning: tons of options; sort of tedious to learn

STEP 7: THEME TWEAKS

- Just as an ex

Fill color of land

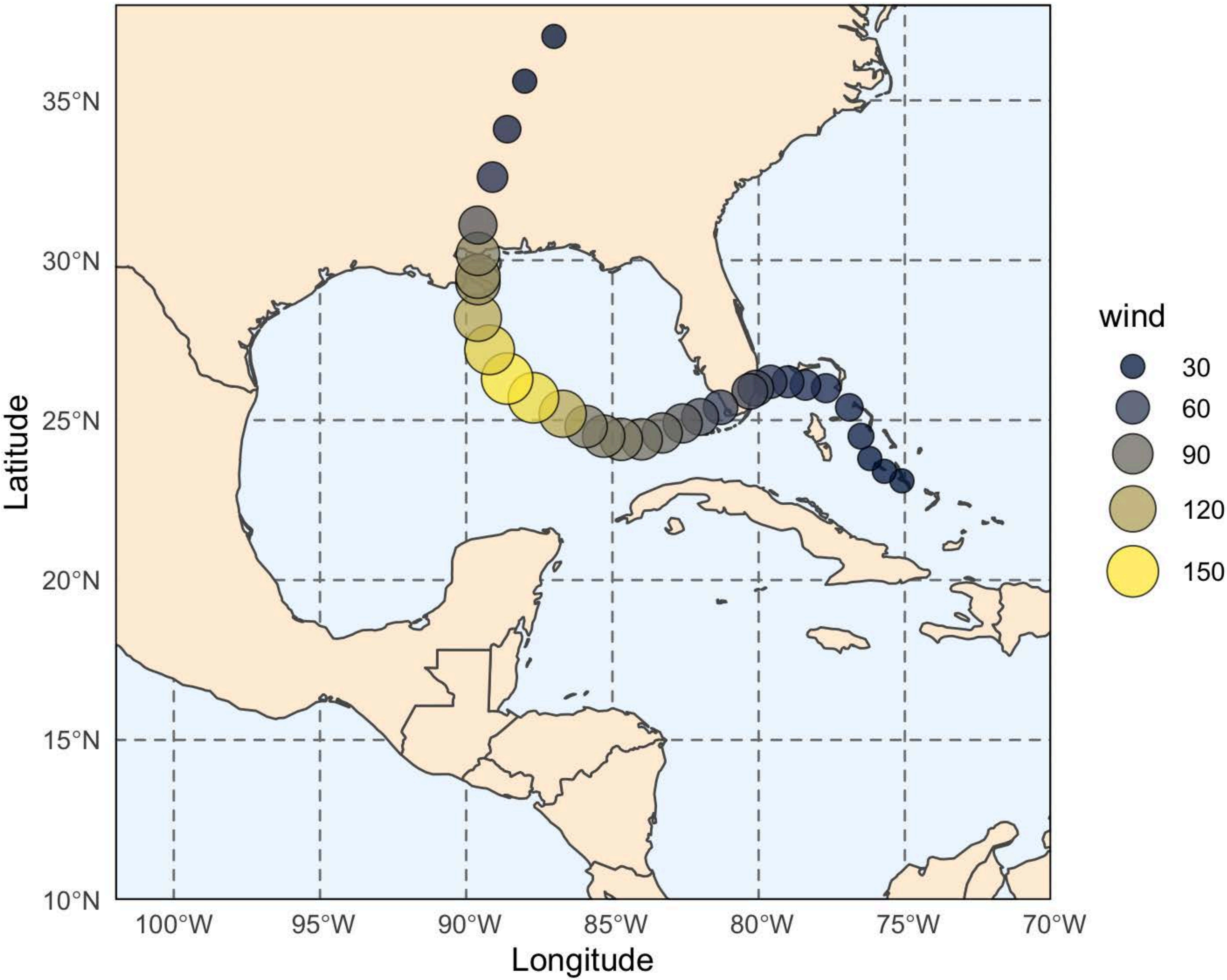
Trim the edges of the axes

Gray dashed graticules (lat/lon grid)

Light blue background for plot area (like water)

Rectangular border (no fill) around entire panel

```
ggplot() +
  geom_sf(data = world, fill = "antiquewhite1
  geom_point(data = katrina,
             aes                     ize = wi
             sha                     ) +
  scale_fill_vi                     dis", guide = "legend") +
  scale_size_area(max_size = 10) +
  coord_sf(xlim                     =                  FALSE) +
  labs(x = "Long                     "
  theme(panel.gr                     ne(color = gray(0.5),
                                        linetype = "dashed",
                                        size = 0.5),
        panel.background = element_rect(fill = "aliceblue"),
        panel.border = element_rect(fill = NA))
```



35°N
30°N
25°N
20°N
15°N
10°N
Latitude
100°W
95°W
90°W
85°W
80°W
75°W
70°W
Longitude
wind
30
60
90
120
150


wind
30
60
90
120
150

READING DATA AND WORKING WITH CRS

- Download the file texas_income.shp from the course website.
- Create a new R markdown file for this activity.
- Load the same packages, plus `here` and `rcartocolor`

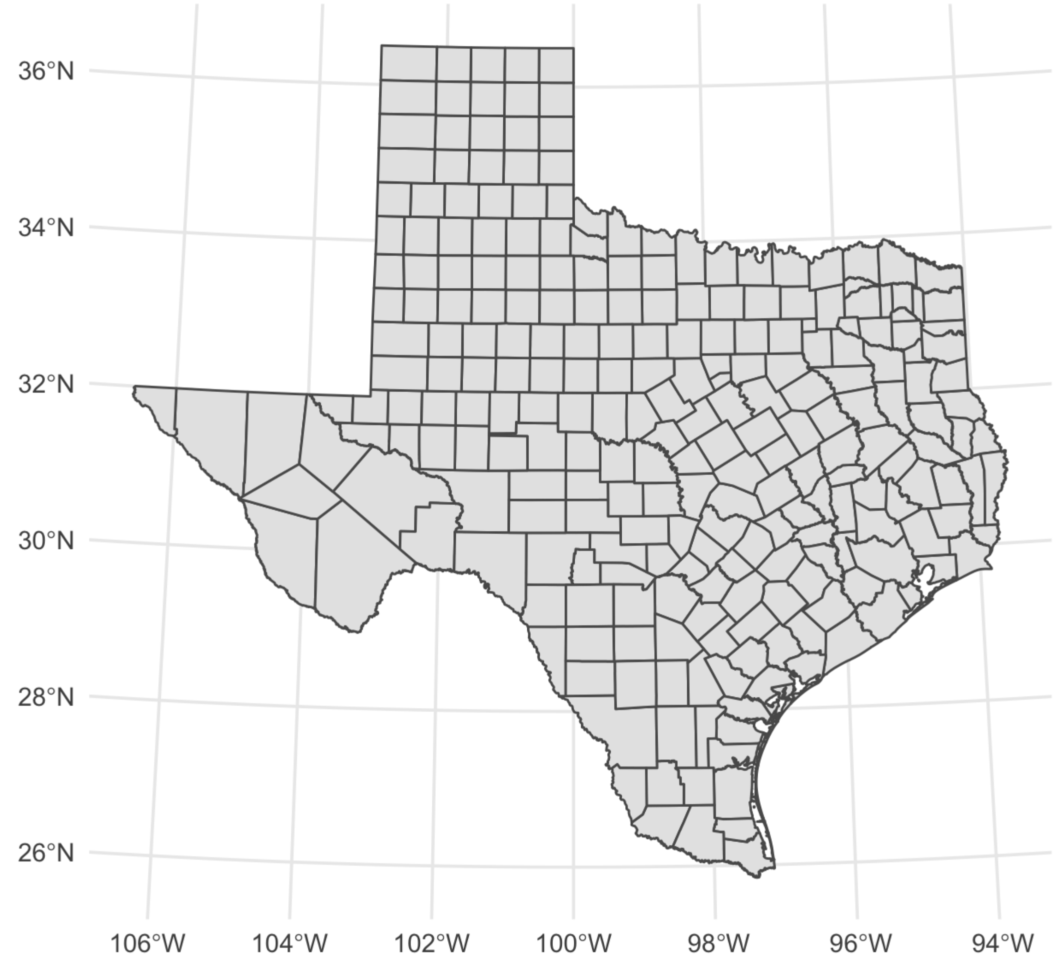
STEP 1: GET THE DATA

- Read in shapefiles and other common vector data formats using the `st_read()` function.

```
texas_income <- st_read(here("your_path/texas_income.shp"))
```

STEP 2: A SIMPLE PLOT

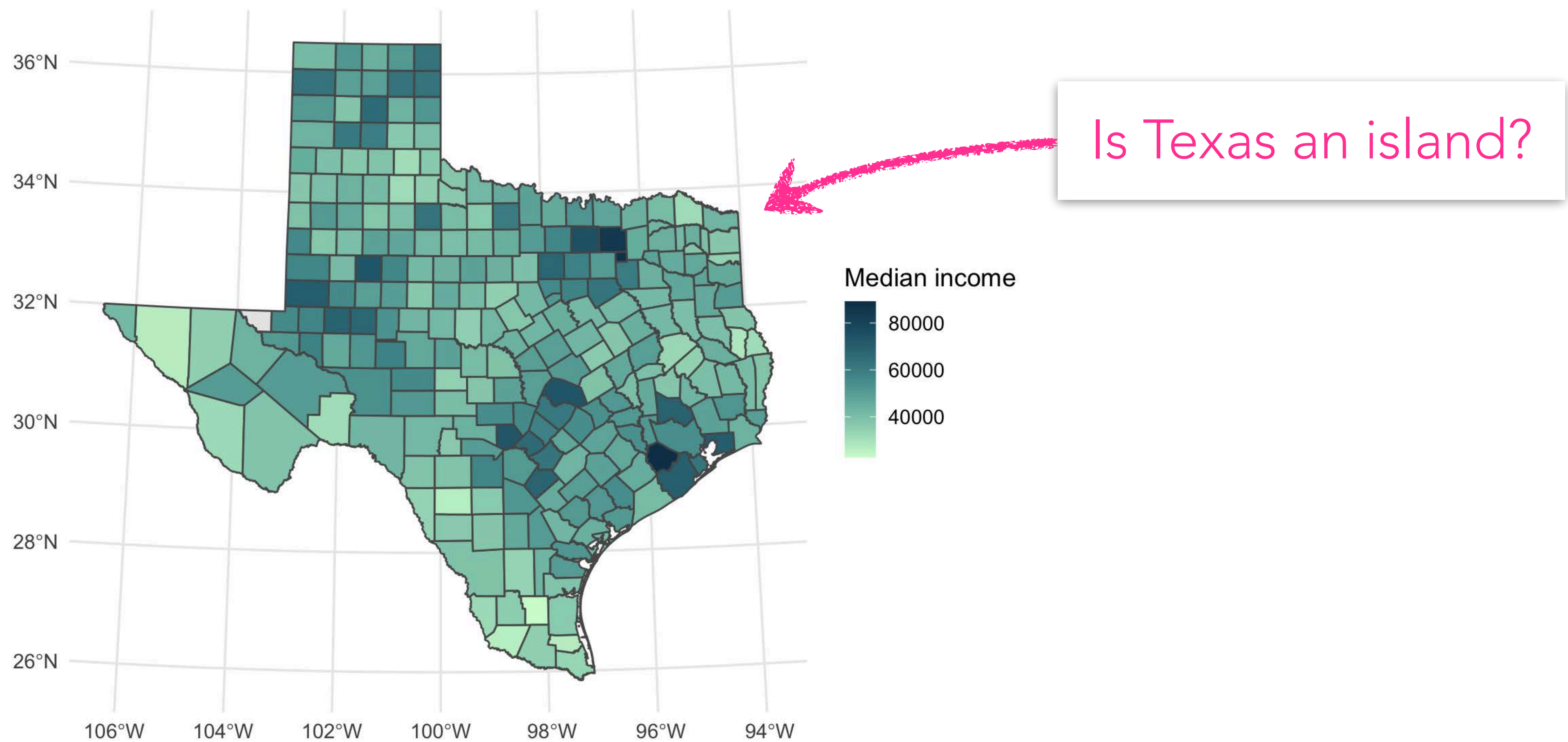
- Use the `geom_sf()` function to plot the geometry of the sf object.




```
ggplot(texas_income) +
  geom_sf()
```

STEP 3: SHOW SOME DATA

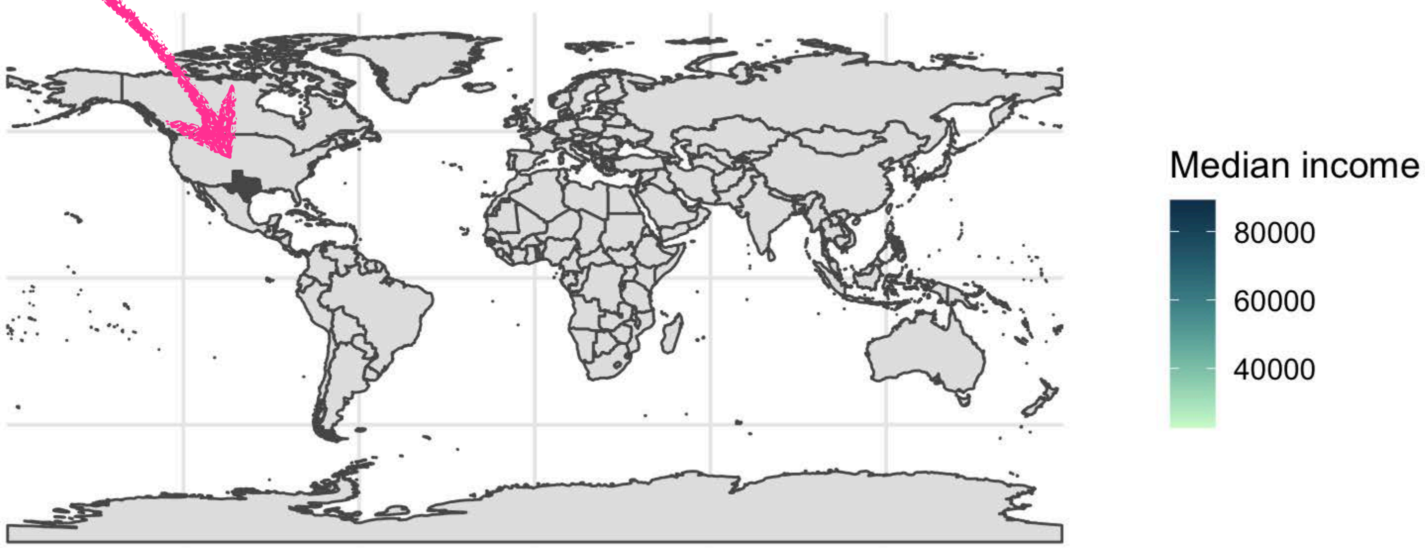
- Map fill to the column called `estimate` and provide a color palette.




```
ggplot(texas_income, aes(fill = estimate)) +
  geom_sf() +
  scale_fill_carto_c(palette = "DarkMint", name = "Median income")
```

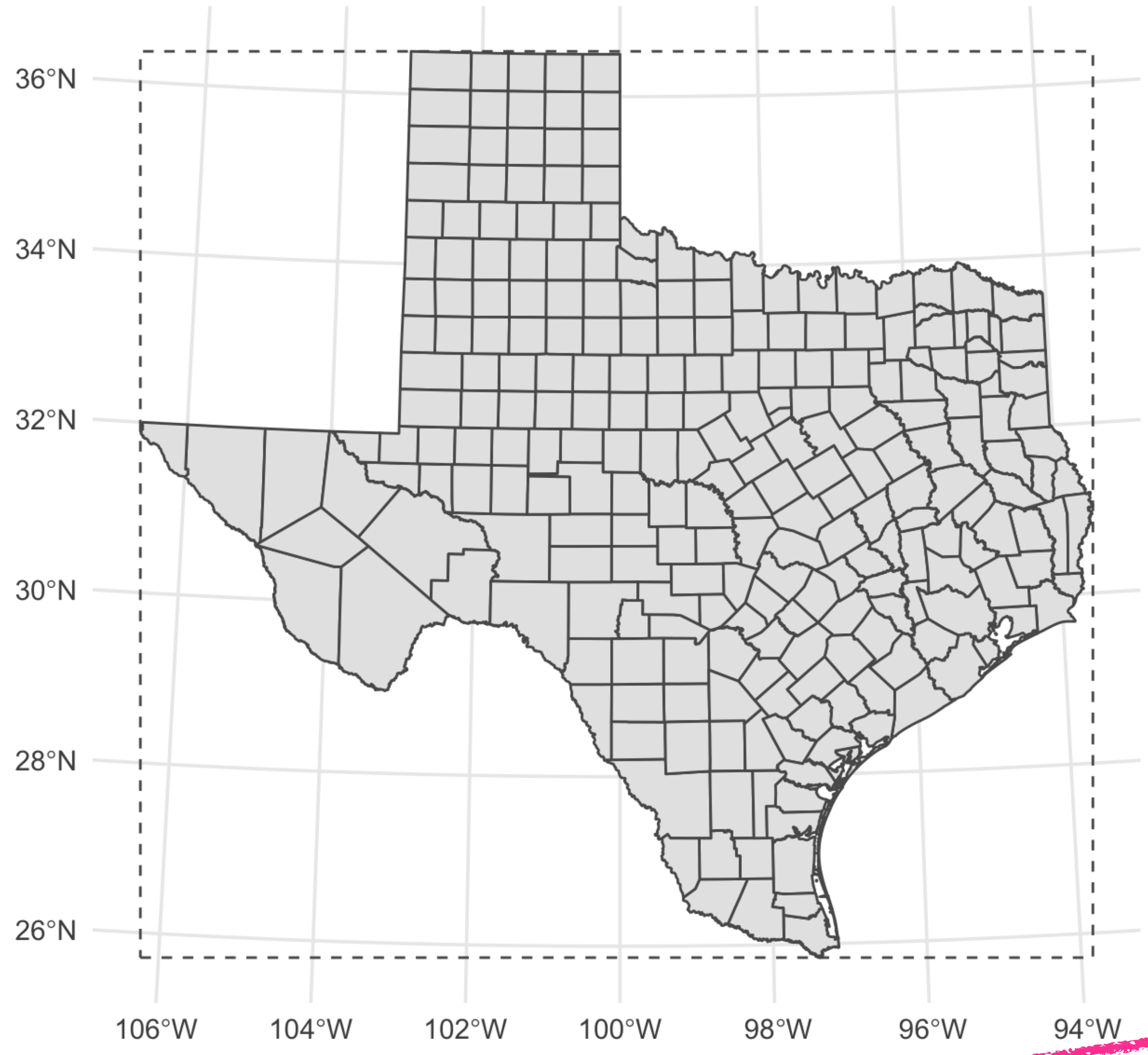
STEP 4: ADD WORLD?

We need to set limits for the plot




```
ggplot() +
  geom_sf(data = world, fill = "gray90") +
  geom_sf(data = texas_income, aes(fill = estimate)) +
  scale_fill_carto_c(palette = "DarkMint", name = "Median income")
```

STEP 5: FIND BBOX



`st_bbox()` finds the bounding box of a simple feature

`coord_sf()` needs xlim and ylim, which we can extract

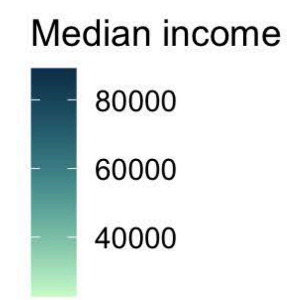
```
st_bbox(texas_income)

texas_xlim <- st_bbox(texas_income)[c("xmin", "xmax")]
texas_ylim <- st_bbox(texas_income)[c("ymin", "ymax")]
```

STEP 6: COMBINE?

Problem: `world` has a different CRS than `texas_income`

`coord_sf()` is smart enough to put everything in the same CRS, but by default, it uses the CRS of the **first** layer and transforms the others to that CRS if they differ



`world` uses lon/lat coordinates

`texas_xlim` and `texas_ylim` are not in lon/lat

```
ggplot() +
  geom_sf(data = world, fill = "gray90") +
  geom_sf(data = texas_income, aes(fill = estimate)) +
  coord_sf(xlim = texas_xlim, ylim = texas_ylim) +
  scale_fill_carto_c(palette = "DarkMint", name = "Median income")
```

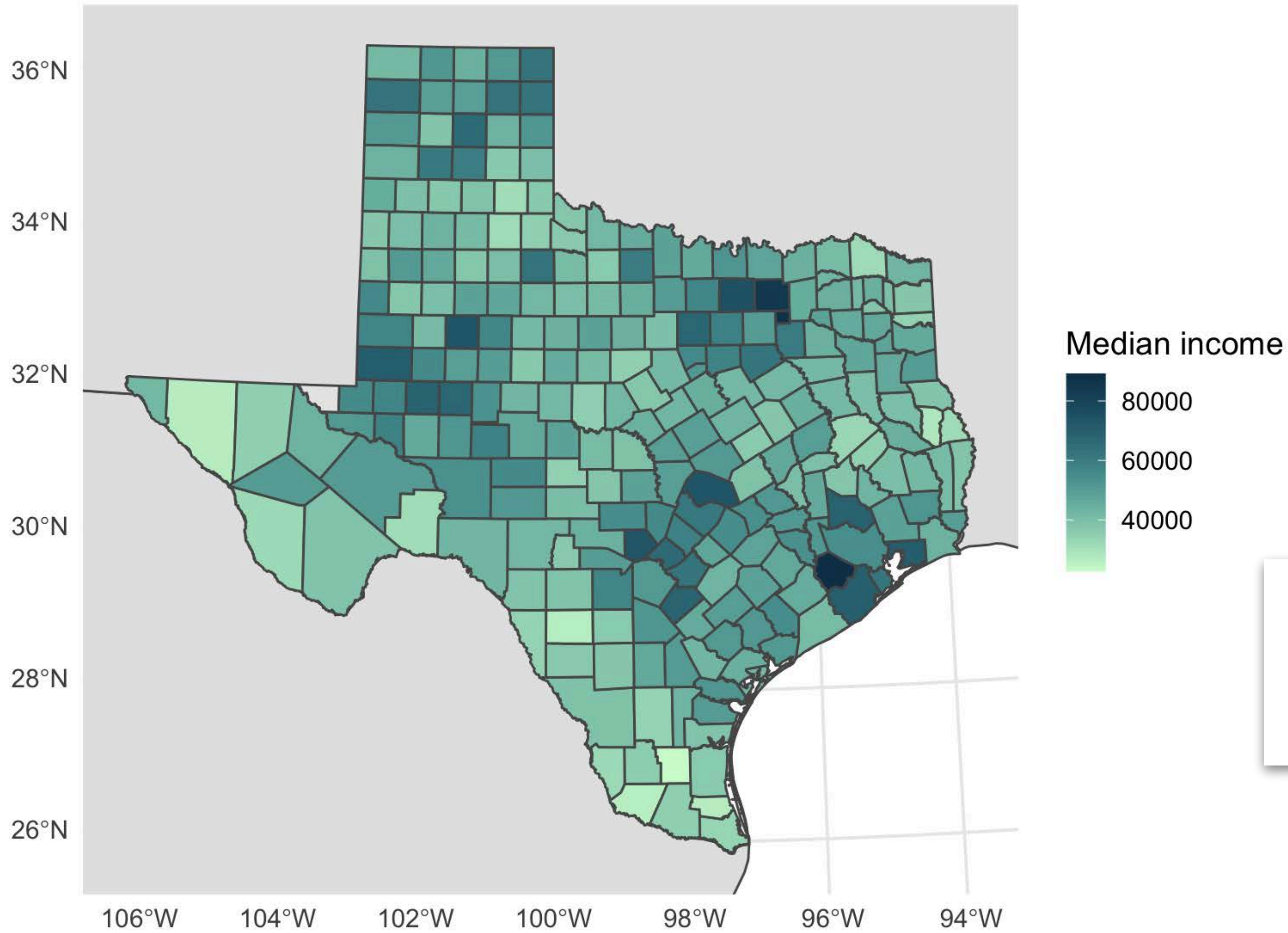
SOLUTION 1:
PROVIDE THE CRS

- Tell `coord_sf()` exactly which CRS to use:

```
texas_crs <- st_crs(texas_income)
```

`st_crs()` returns the CRS of a simple feature

SOLUTION 1: PROVIDE THE CRS



Provide the CRS to `coord_sf()`

```
ggplot() +
  geom_sf(data = world, fill = "gray90") +
  geom_sf(data = texas_income, aes(fill = estimate)) +
  coord_sf(xlim = texas_xlim, ylim = texas_ylim, crs = texas_crs) +
  scale_fill_carto_c(palette = "DarkMint", name = "Median income")
```

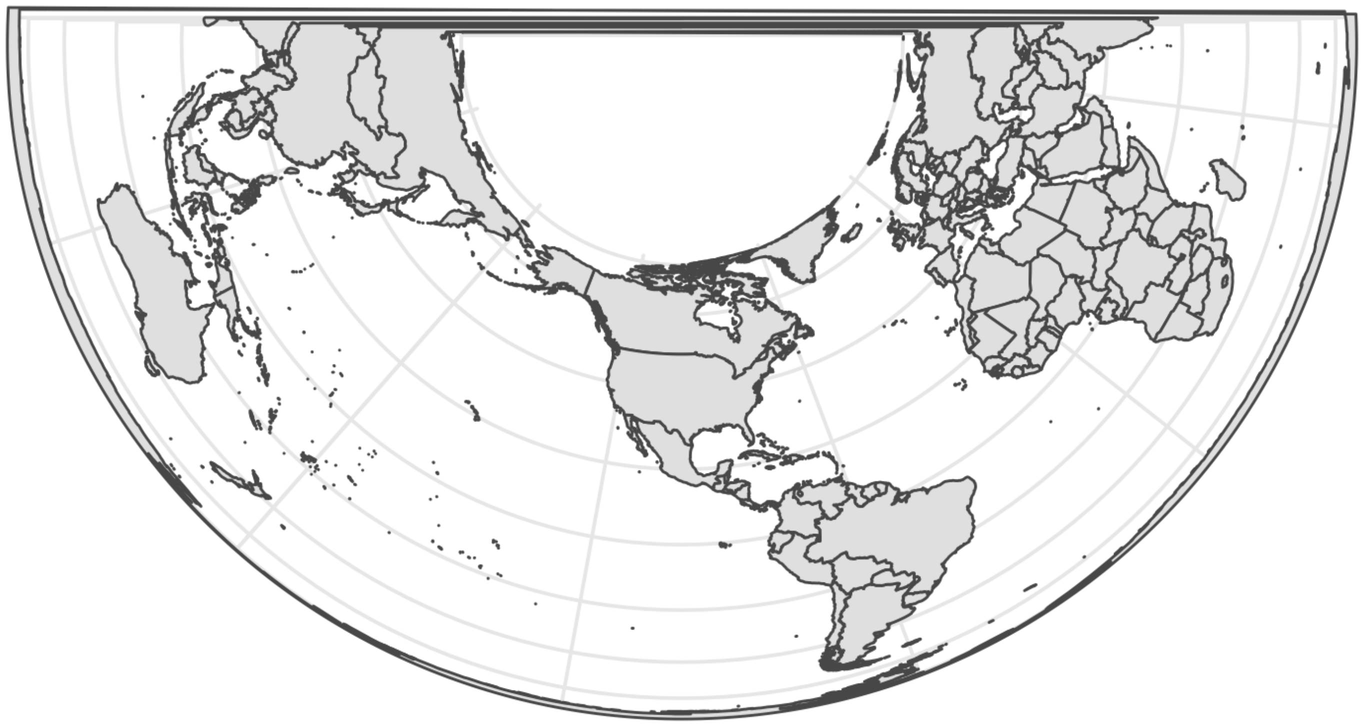
SOLUTION 2:
TRANSFORM LAYERS

- Sometimes, especially when you're doing geospatial analysis, you'll need your layers to all share the same CRS.

```
world_tx <- st_transform(world, crs = texas_crs)
```

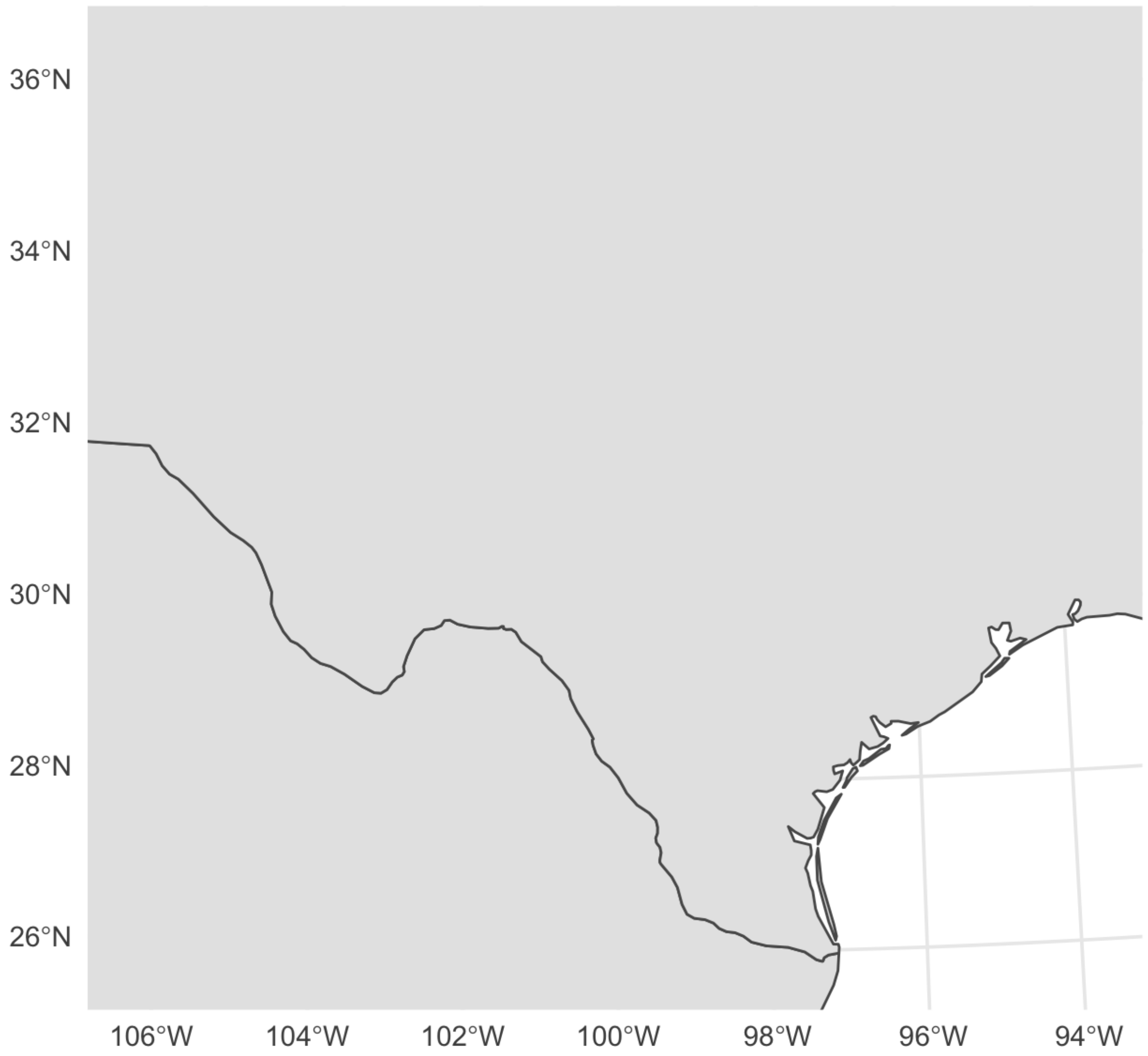
`st_transform()` converts a simple feature from one CRS to another

SOLUTION 2: TRANSFORM LAYERS



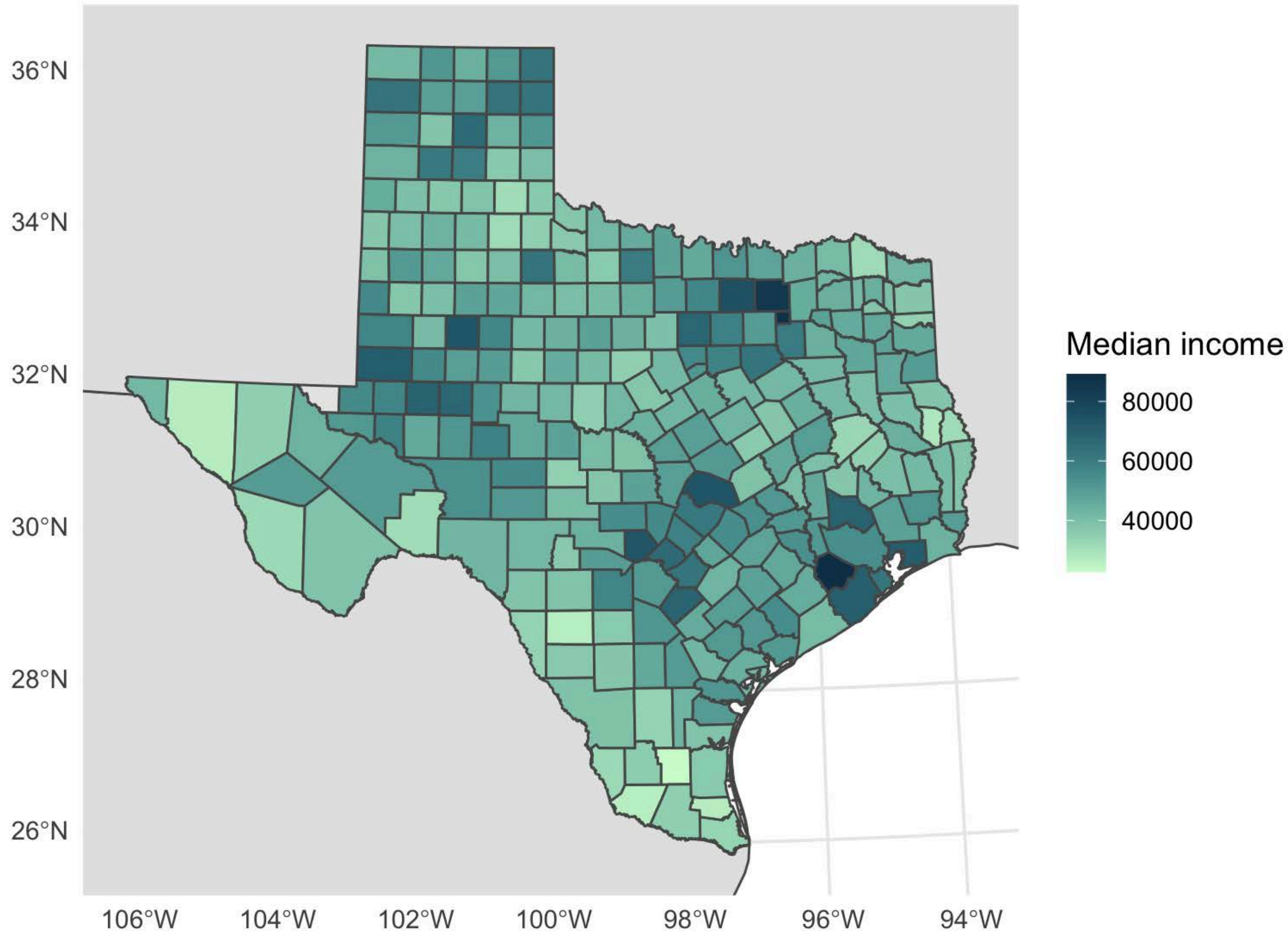
```
ggplot() +
  geom_sf(data = world_tx, fill = "gray90")
```

SOLUTION 2:
TRANSFORM LAYERS




```
ggplot() +
  geom_sf(data = world_tx, fill = "gray90") +
  coord_sf(xlim = texas_xlim, ylim = texas_ylim)
```

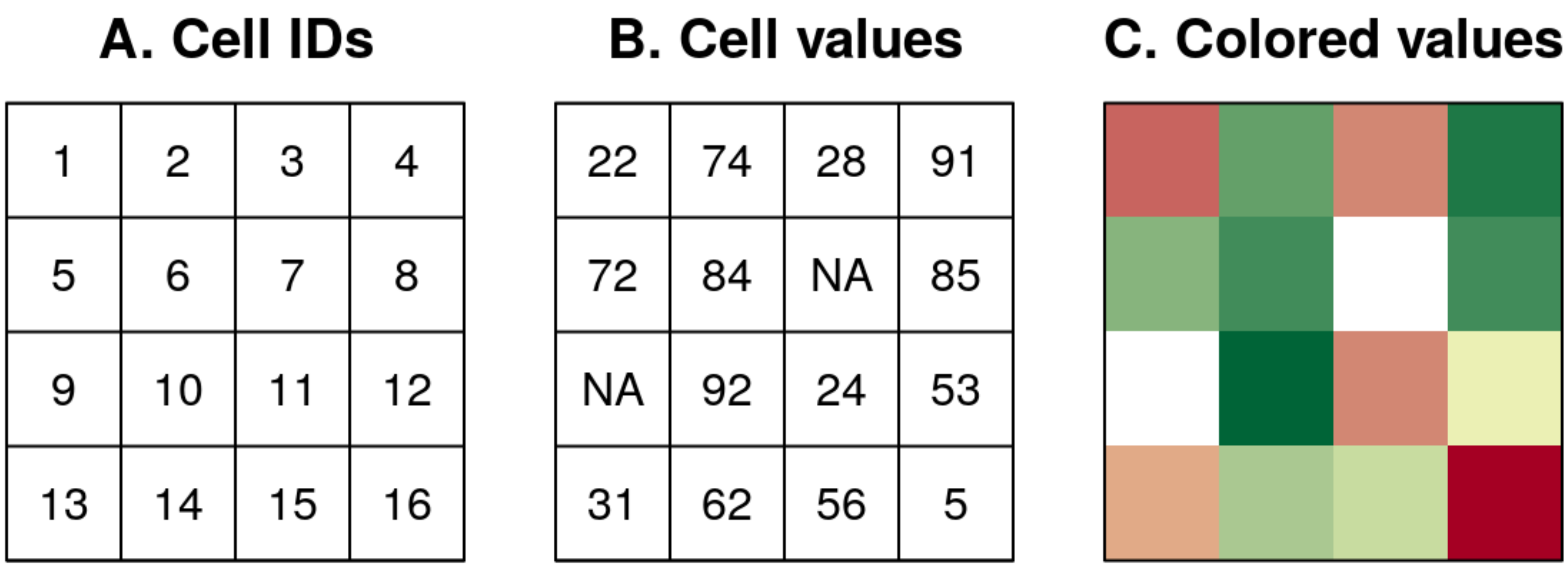
SOLUTION 2: TRANSFORM LAYERS



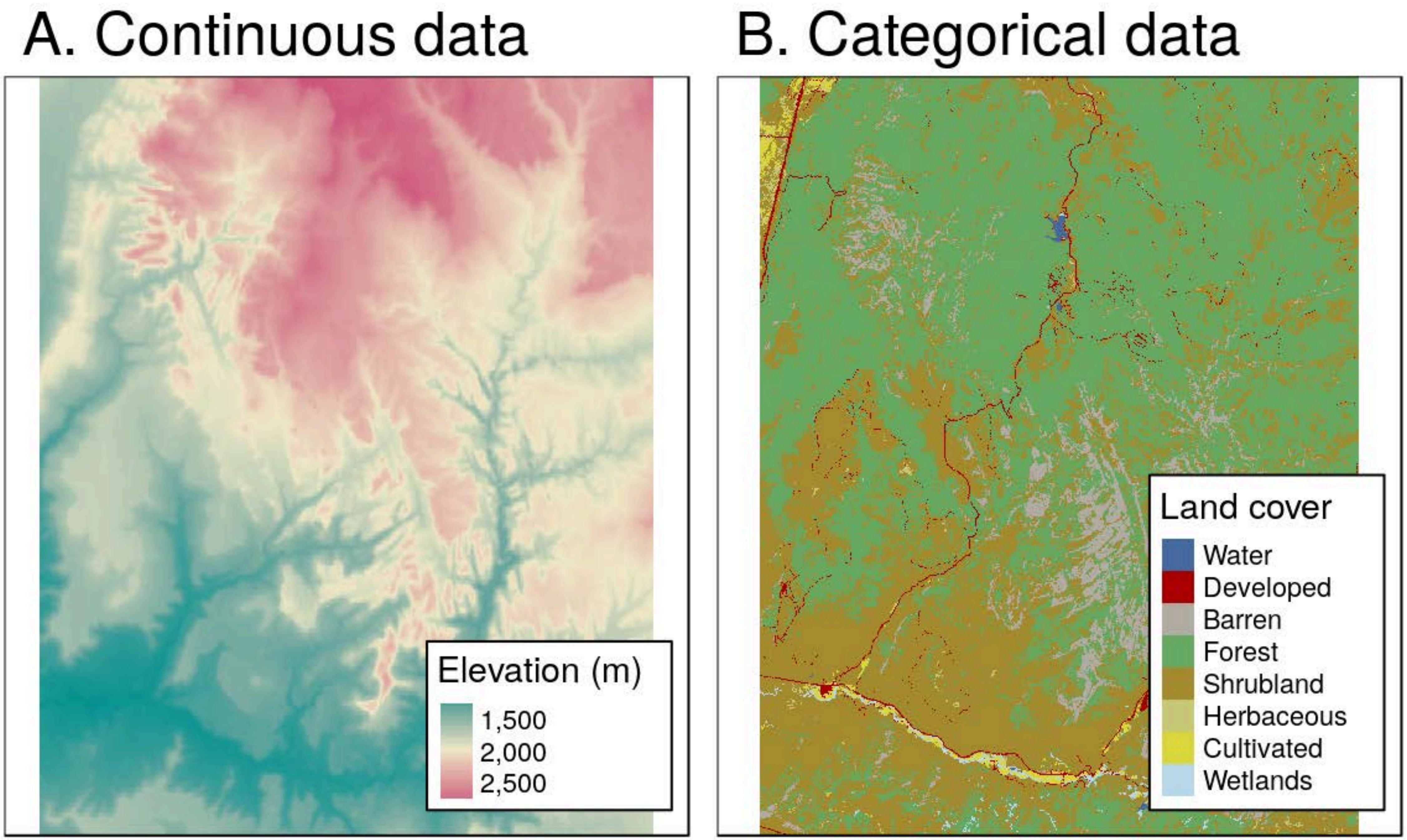

```
ggplot() +
  geom_sf(data = world_tx, fill = "gray90") +
  geom_sf(data = texas_income, aes(fill = estimate)) +
  coord_sf(xlim = texas_xlim, ylim = texas_ylim) +
  scale_fill_carto_c(palette = "DarkMint", name = "Median income")
```

RASTER DATA LAYERS

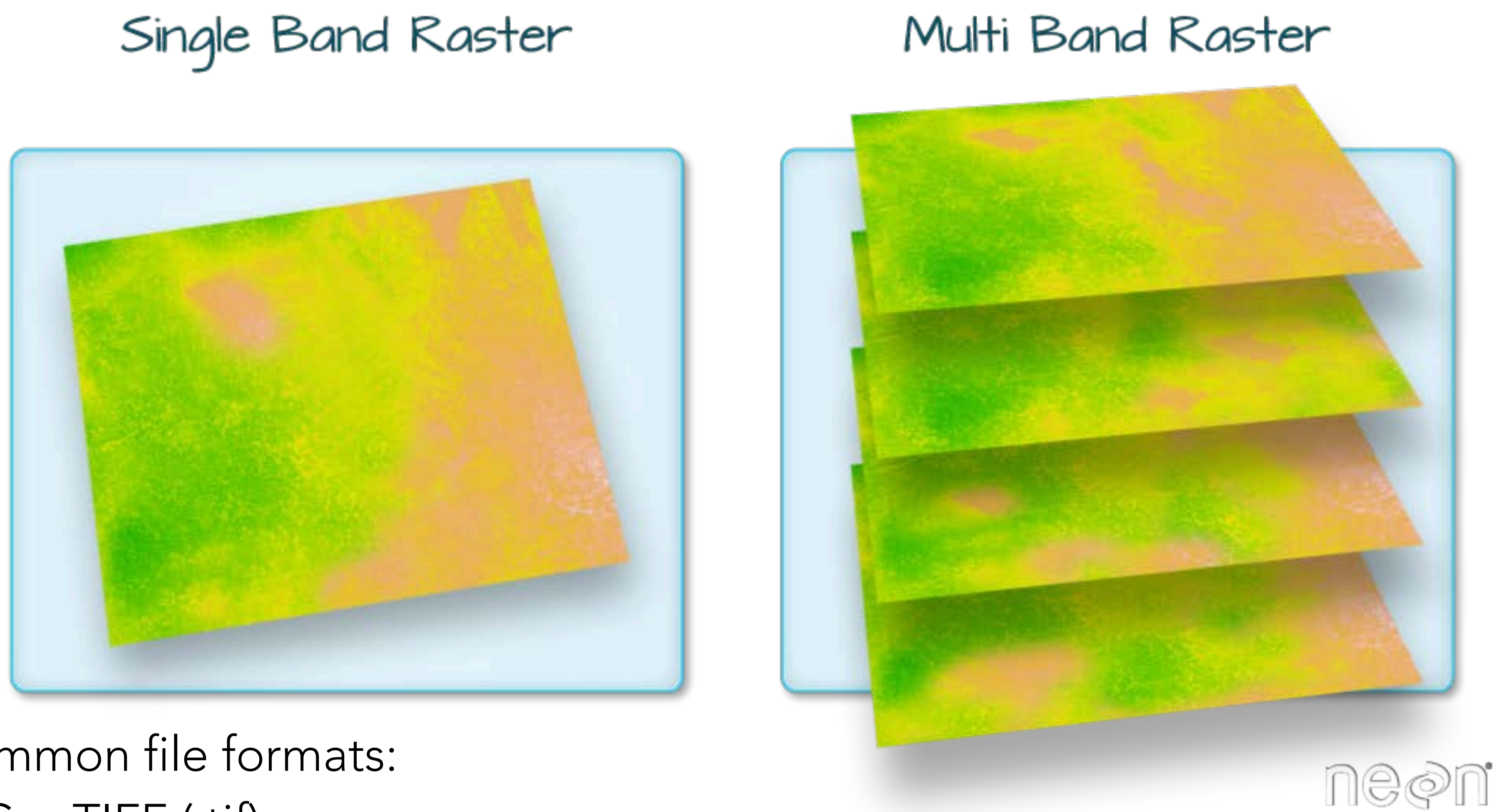
RASTER DATA LAYERS



RASTER DATA LAYERS



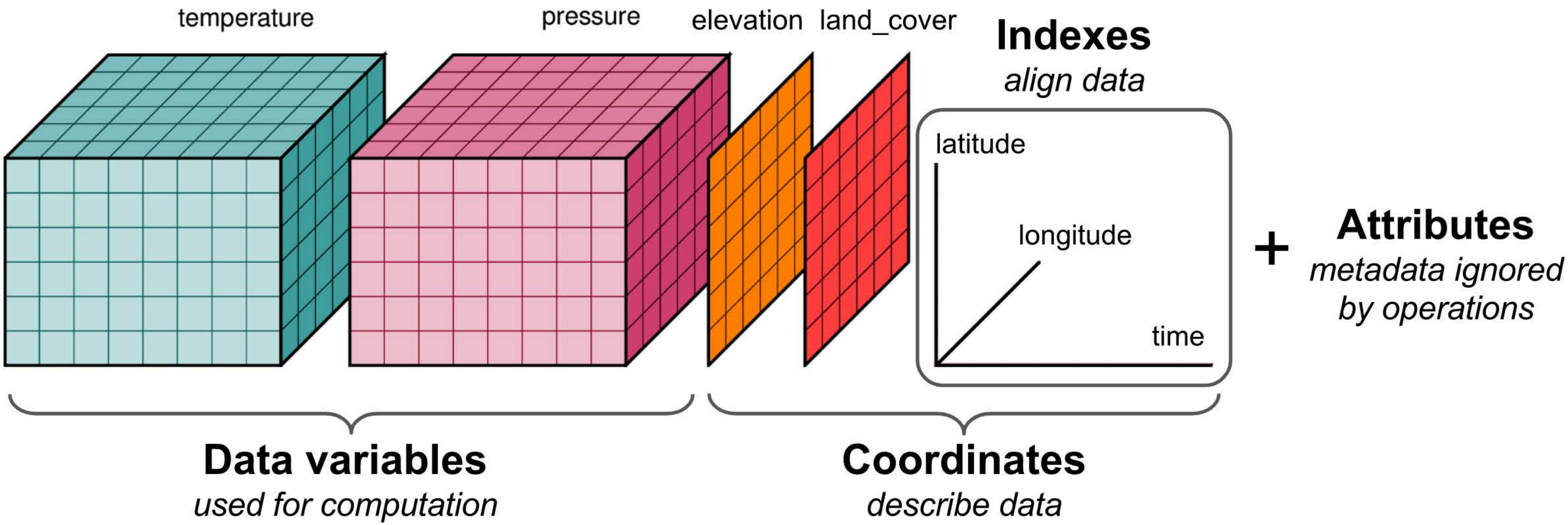
SIMPLE RASTER DATA FILE FORMATS



Common file formats:

- GeoTIFF (.tif)
- Erdas Imagine (.img)
- ASCII (.asc)

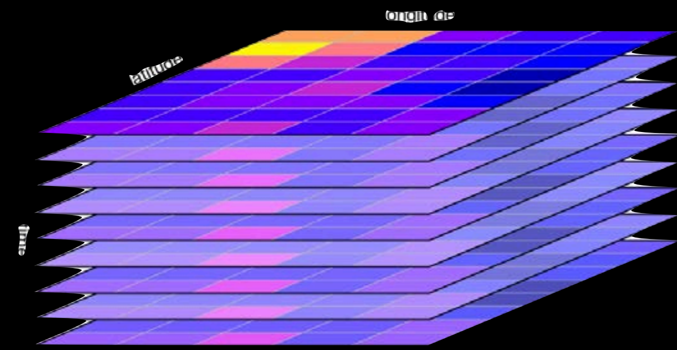
DATA CUBE FILE FORMATS



Common file formats:

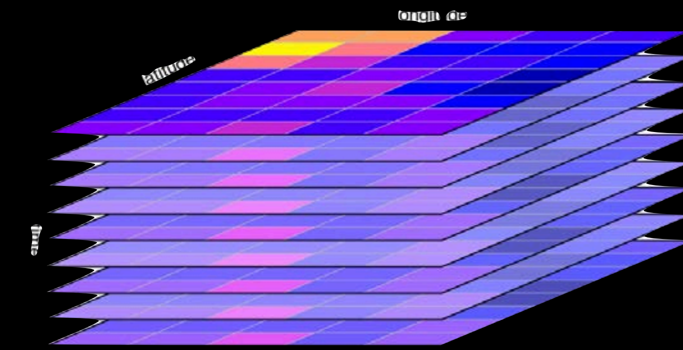
- HDF (.hdf)
- NetCDF (.nc)

THE STARS PACKAGE



- Package for working with "data cubes," including common raster data formats.
  - Array data with labeled dimensions, where some of the dimensions relate to space and/or time.
- Support for tidyverse methods, including ggplot2.
- Support for sf methods

THE STARS PACKAGE



Read in GeoTIFF file

```
sr <- read_stars(here("your_path/acg_elevation.tif"))
```

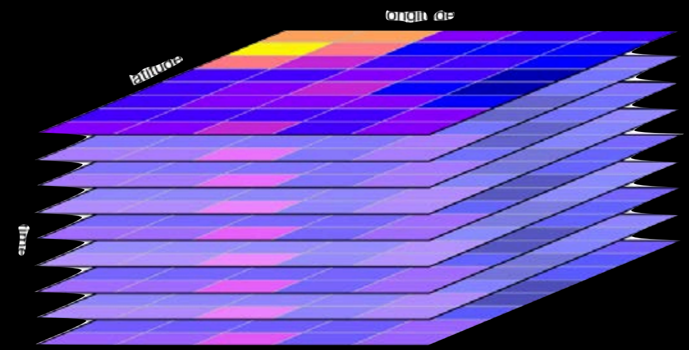
```
stars object with 2 dimensions and 1 attribute
attribute(s):
 acg_elevation.tif
 Min.   :   0.0
 1st Qu.: 146.0
 Median : 277.0
 Mean   : 342.8
 3rd Qu.: 441.0
 Max.   :1894.0
 NA's   :161687
dimension(s):
  from  to   offset        delta refsys point values x/y
x    1 904 -85.9688  0.000833333 WGS 84 FALSE   NULL [x]
y    1 563  11.0929 -0.000833333 WGS 84 FALSE   NULL [y]
```

Names of layers in raster (only 1 here)

Summary of cell values

Spatial attributes

THE STARS PACKAGE



- stars objects can be plotted in ggplot2 like normal `geom_` layers using the `geom_stars()` function.
- First attribute used as fill variable
    - If multiple attributes/layers, others can be plotted by faceting or by "slicing" out a single layer.

YOUR TURN

- Download the file acg_elevation.tif from the course website.
- Create a new R markdown file for this activity.
- Load the packages `stars`, `here`, `scico`, and `tidyverse`

STEP 1: GET THE DATA

- Read in GeoTIFF and other common raster data formats using the `read_stars()` function.

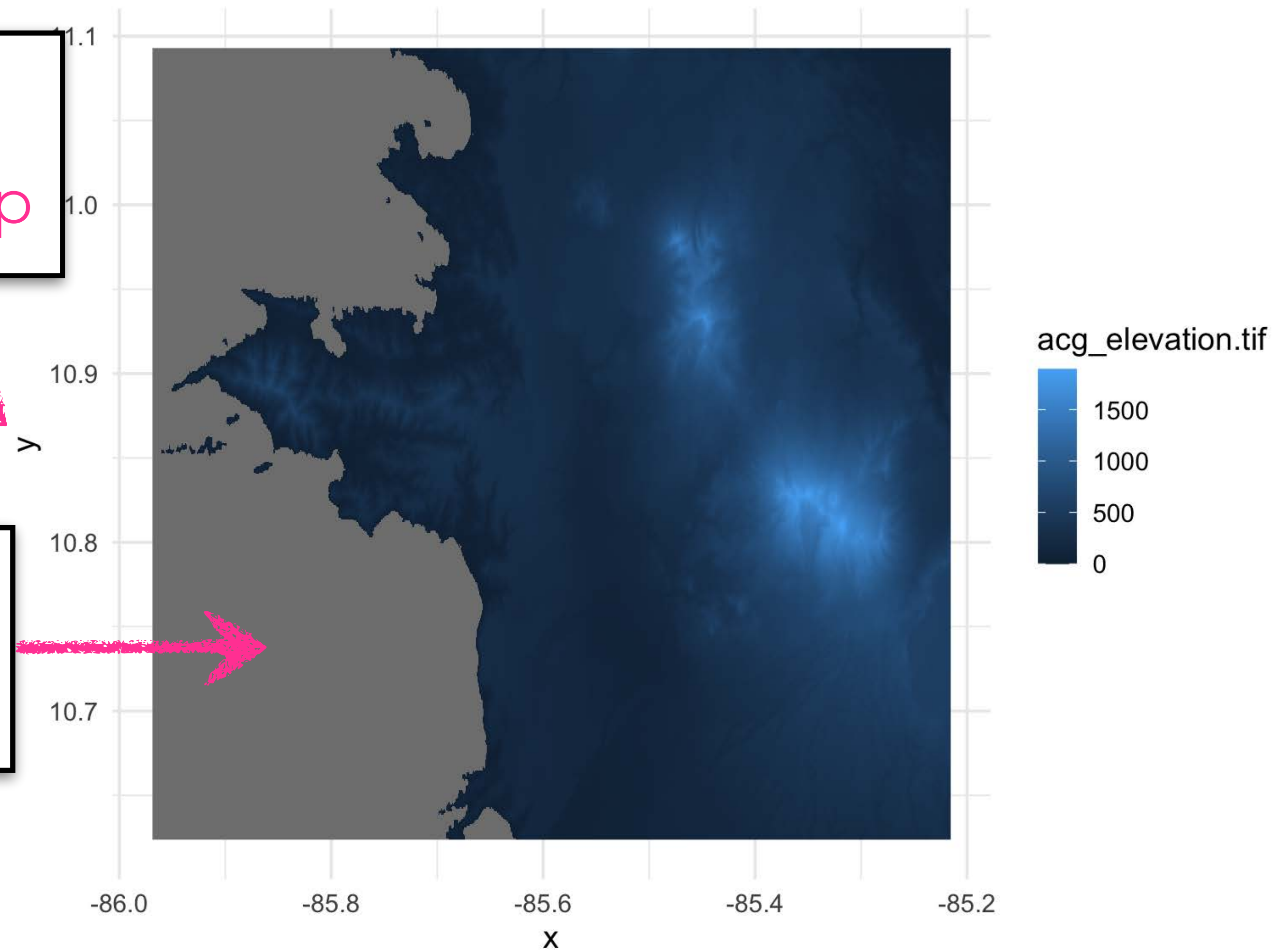
```
sr <- read_stars(here("your_path/acg_elevation.tif"))
```

STEP 2: A SIMPLE PLOT

- Use the `geom_stars()` function to plot the raster layer. By default, fill is set to the first attribute.

Unequal X and Y scales stretch the map

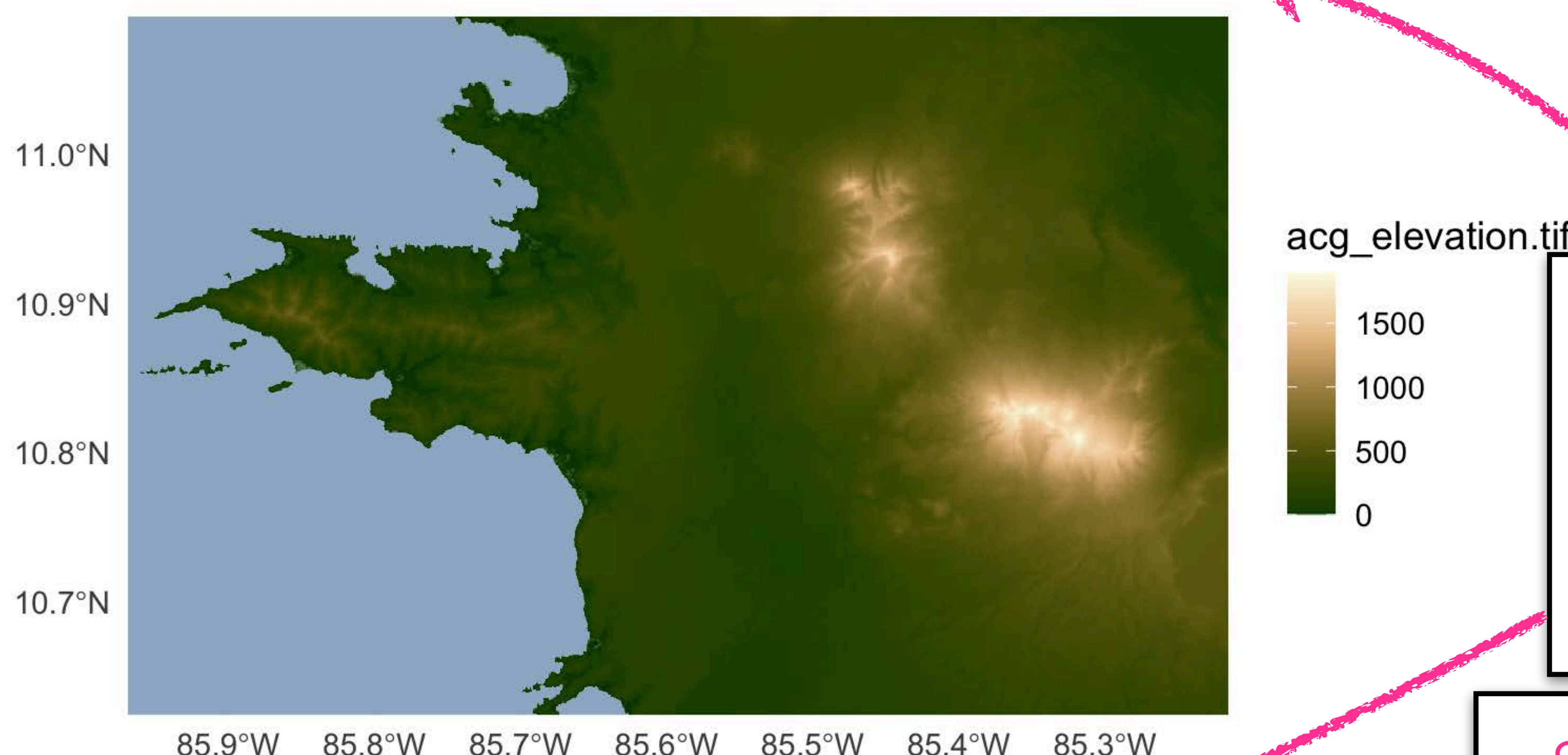
Missing values represent water



Default sequential palette is not good

```
ggplot() +
  geom_stars(data = sr)
```

STEP 3: THEME TWEAKS



oleron is a diverging scale for elevation and bathymetry

Set bluish color for missing values

Provide CRS for lon/lat formatting

```
ggplot() +
  geom_stars(data = sr) +
  scale_fill_scico(palette = "oleron", begin = 0.5,
                   na.value = "slategray3") +
  coord_sf(crs = st_crs(sr), expand = FALSE) +
  labs(x = NULL, y = NULL)
```

ACTIVITY: MAPS

- Go to this week's assignments on the course website.
- Follow the instructions in the maps.Rmd file to learn some map-making skills.