Visualizing spatial data in-class exercise

Using maps

```r
library(scico)
library(maps)
library(stars)
```

```
## Loading required package: abind
```

```
## Loading required package: sf
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```r
library(magrittr) # pipes
library(lintr) # code linting
library(raster) # raster handling (needed for relief)
```

```
## Loading required package: sp
```

```r
library(viridis) # viridis color scale
```

```
## Loading required package: viridisLite
```

```
##
## Attaching package: 'viridis'
```

```
## The following object is masked from 'package:maps':
##
##     unemp
```

```r
library(cowplot) # stack ggplots
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::extract()   masks raster::extract(), magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x purrr::map()       masks maps::map()
## x dplyr::select()    masks raster::select()
## x purrr::set_names() masks magrittr::set_names()
## x lubridate::stamp() masks cowplot::stamp()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

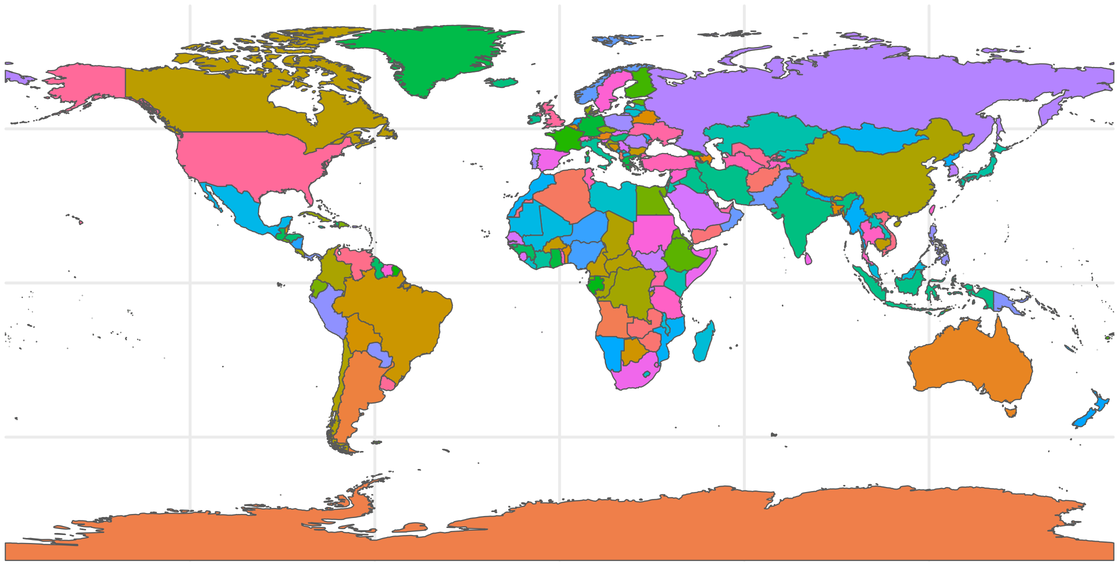
```r
library(sf)
library(rworldmap)
```

```
## ### Welcome to rworldmap ###
```

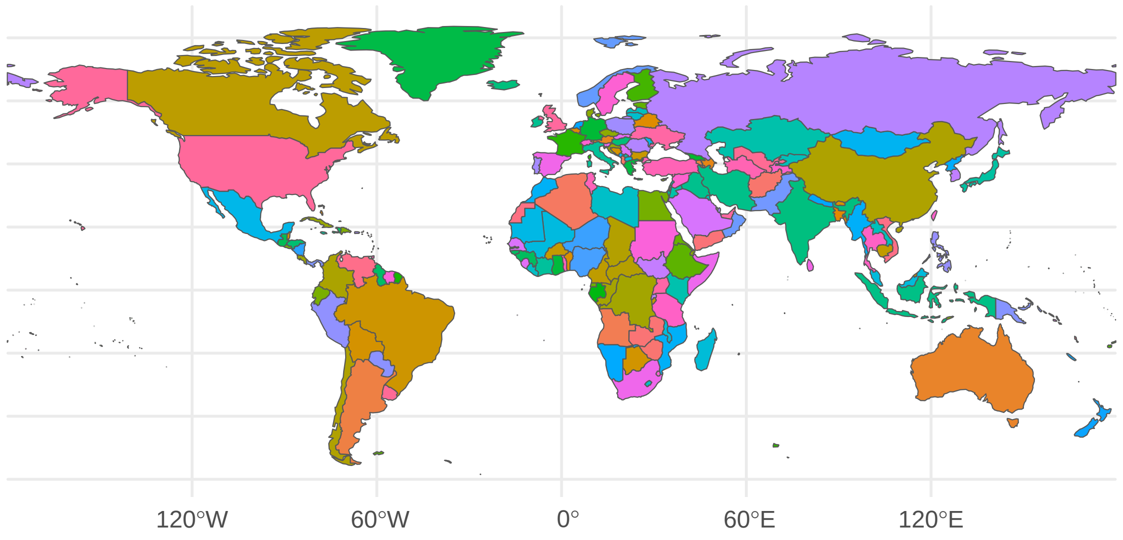
```
## For a short introduction type :   vignette('rworldmap')
```

```
wmap <- rworldmap::getMap(resolution = "low")  %>%
  st_as_sf()

ggplot(data = wmap) +
  geom_sf(aes(fill = NAME)) +
  theme_minimal() +
  guides(fill = FALSE) # don't show legend
```



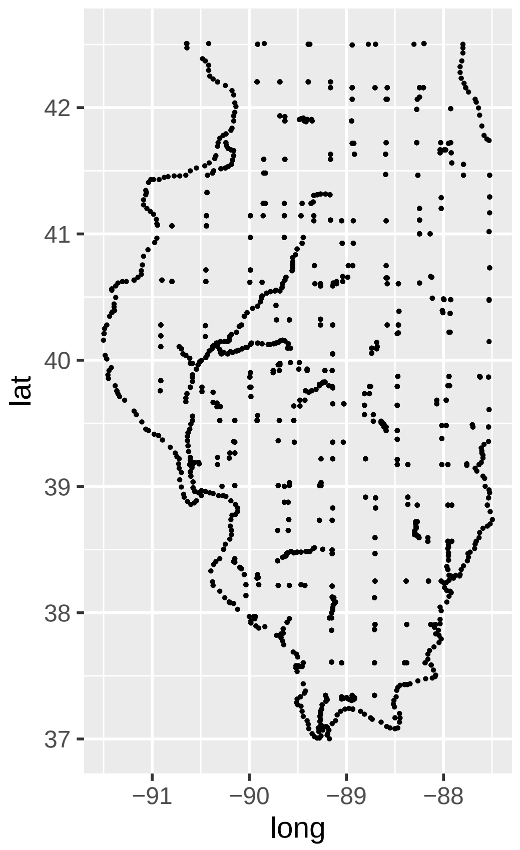
```
wmap_ant <- getMap()[-which(getMap()$ADMIN=='Antarctica'),]  %>%
  st_as_sf()
ggplot(data = wmap_ant) +
  geom_sf(aes(fill = NAME)) +
  theme_minimal() +
  guides(fill = FALSE) # don't show legend
```



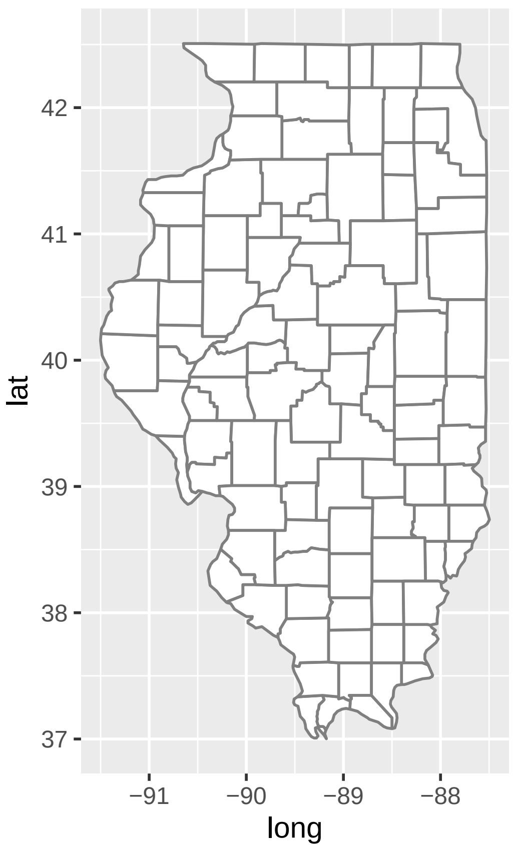
```
il_counties <- map_data("county", "illinois")
head(il_counties)
```

```
##        long      lat group order   region subregion
## 1 -91.49563 40.21018     1     1 illinois     adams
## 2 -90.91121 40.19299     1     2 illinois     adams
## 3 -90.91121 40.19299     1     3 illinois     adams
## 4 -90.91121 40.10704     1     4 illinois     adams
## 5 -90.91121 39.83775     1     5 illinois     adams
## 6 -90.91694 39.75754     1     6 illinois     adams
```

```
ggplot(il_counties, aes(long, lat)) +
  geom_point(size = .25, show.legend = FALSE) +
  coord_quickmap()
```



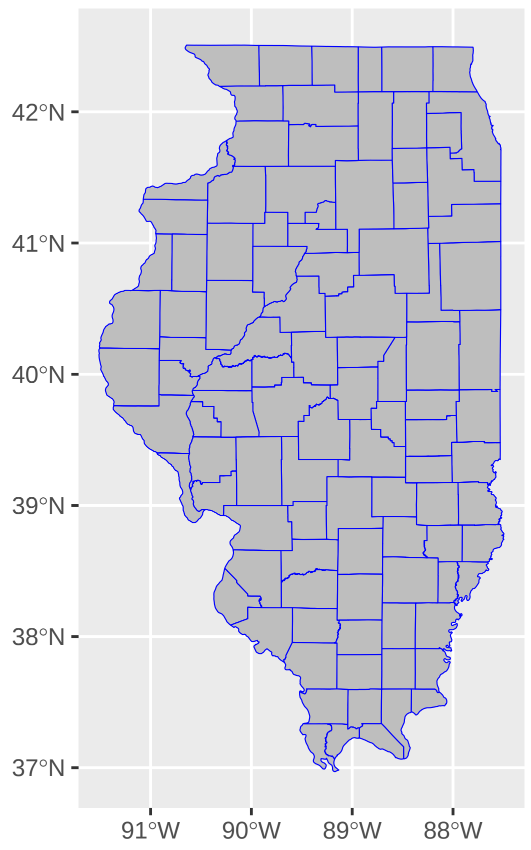
```
ggplot(il_counties, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "grey50") +
  coord_quickmap()
```



```r
il_county <- tigris::counties(state = "illinois", cb = TRUE) %>%
    st_as_sf()
```

```
##    |                                                                                                    |
```

```r
ggplot(il_county) +
  geom_sf(color = "blue", fill = "gray")
```



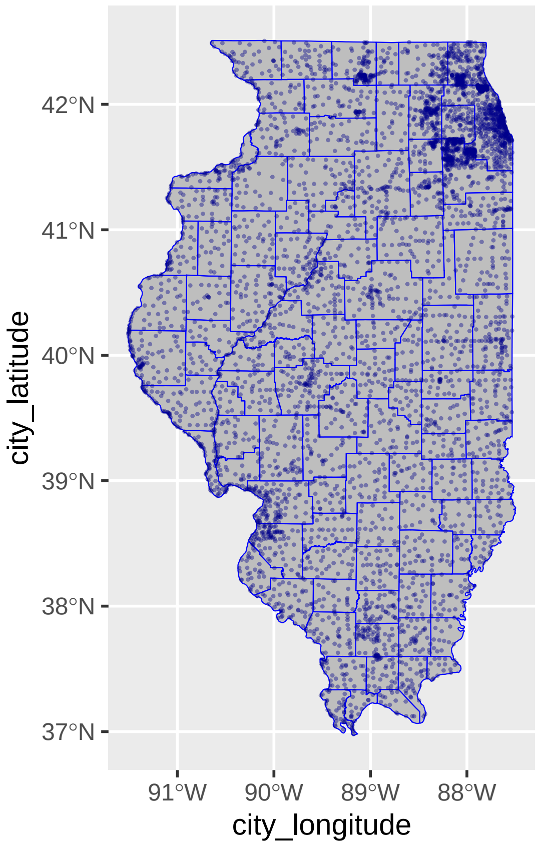
Add cities

Data source: https://geo-csv.com/illinois/

```
illinois <- read_csv("data/illinois.csv")
```

```
## Rows: 5272 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): city_name
## dbl (2): city_latitude, city_longitude
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

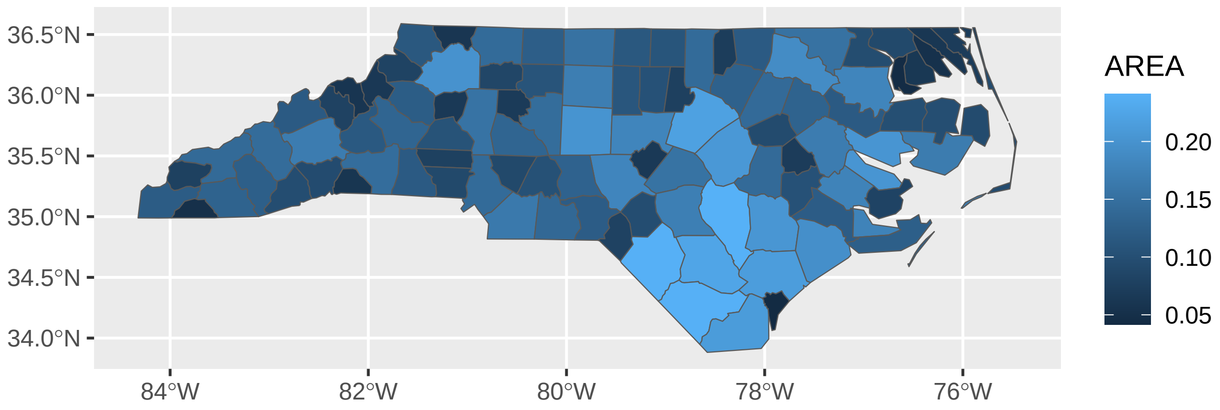
```
ggplot(il_county) +
  geom_sf(color = "blue", fill = "gray") +
 geom_point(data = illinois,
            aes(x = city_longitude, y = city_latitude),
            colour = "darkblue", size = 0.1, alpha = 0.4) +
  coord_sf()
```



Raster maps

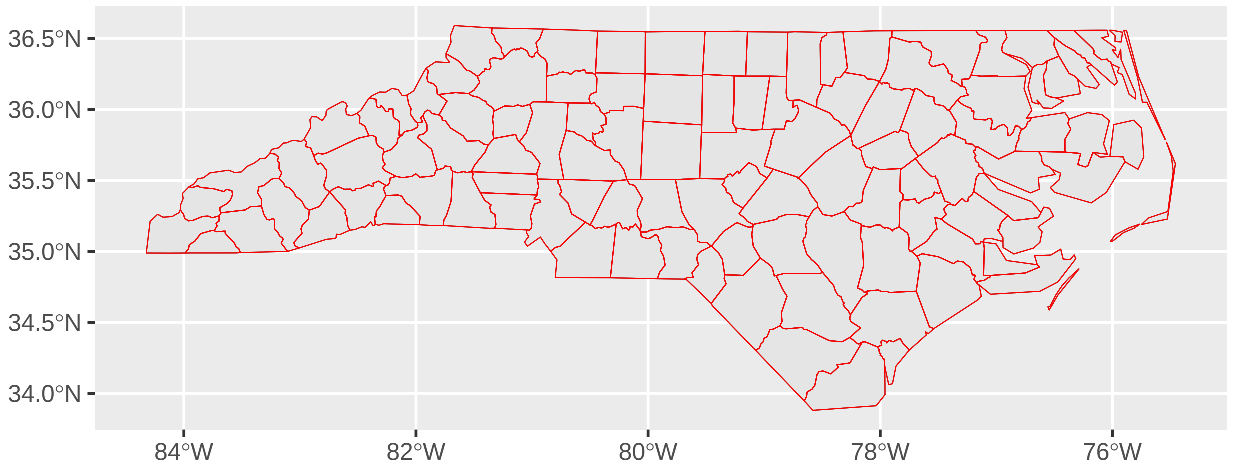
Simple example: https://ggplot2.tidyverse.org/reference/ggsf.html

```
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
ggplot(nc) +
  geom_sf(aes(fill = AREA))
```

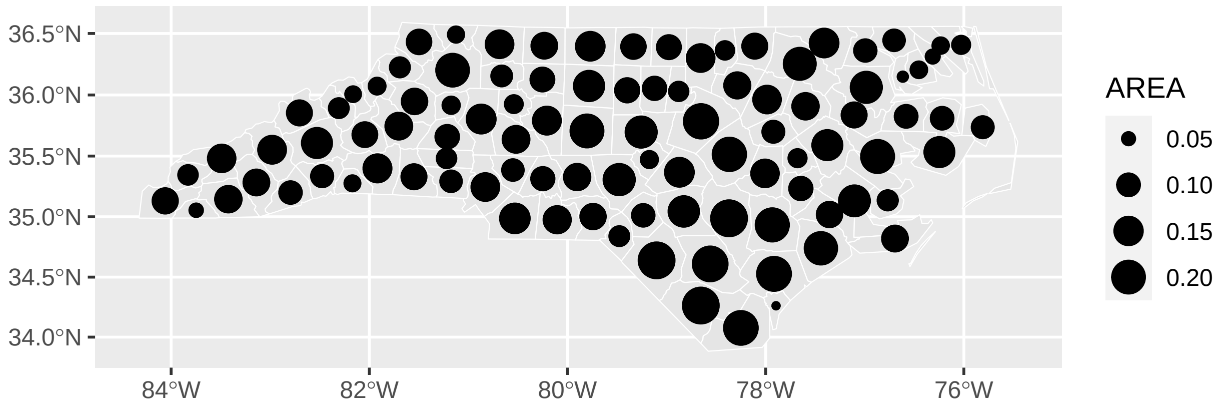


```
# If not supplied, coord_sf() will take the CRS from the first layer
# and automatically transform all other layers to use that CRS. This
# ensures that all data will correctly line up
nc_3857 <- sf::st_transform(nc, 3857)
ggplot() +
```

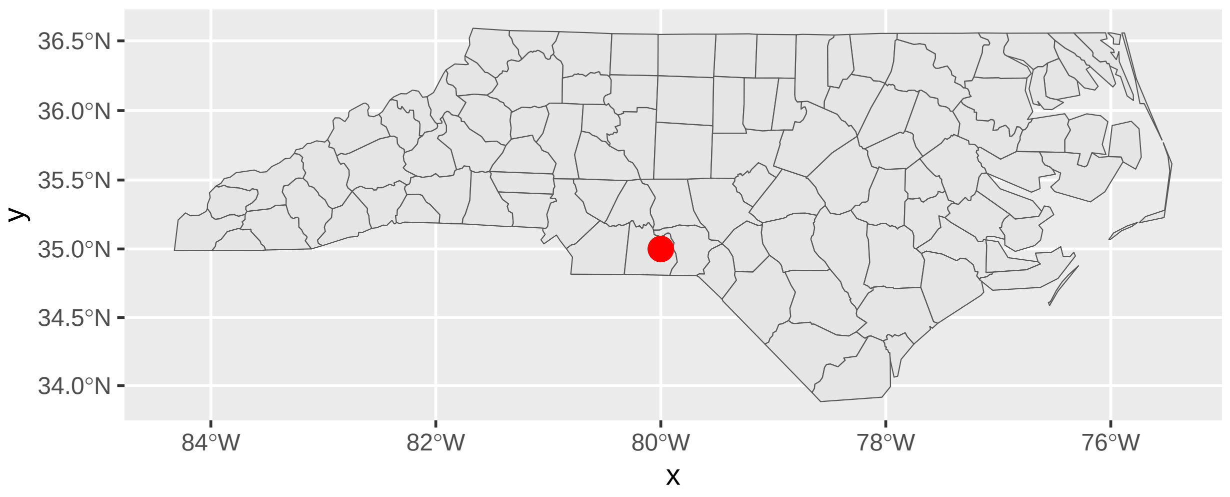
```r
  geom_sf(data = nc) +
  geom_sf(data = nc_3857, colour = "red", fill = NA)
```



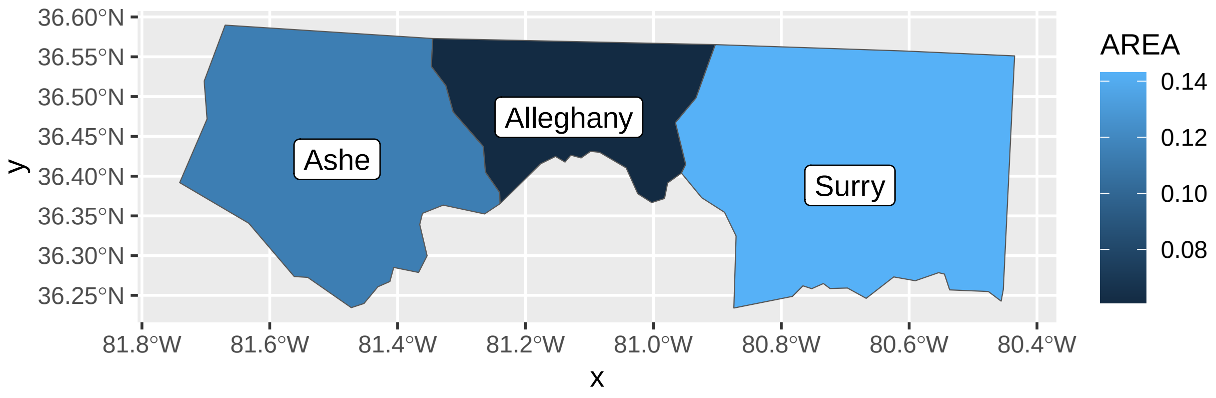
```r
# Unfortunately if you plot other types of feature you'll need to use
# show.legend to tell ggplot2 what type of legend to use
nc_3857$mid <- sf::st_centroid(nc_3857$geometry)
ggplot(nc_3857) +
  geom_sf(colour = "white") +
  geom_sf(aes(geometry = mid, size = AREA), show.legend = "point")
```



```r
# You can also use layers with x and y aesthetics. To have these interpreted
# as longitude/latitude you need to set the default CRS in coord_sf()
ggplot(nc_3857) +
  geom_sf() +
  annotate("point", x = -80, y = 35, colour = "red", size = 4) +
  coord_sf(default_crs = sf::st_crs(4326))
```



```r
# To add labels, use geom_sf_label().
ggplot(nc_3857[1:3, ]) +
  geom_sf(aes(fill = AREA)) +
  geom_sf_label(aes(label = NAME))
```



Spain

source: https://github.com/aaumaitre/maps_Spain