STAT 118: Notes K

Maps with `maps` and `sf`

Emily Malcolm-White

Before we get started, some context:

- R is ***fantastic*** for spacial analysis (not covered in this class… look for classes related to spacial statistics)
- R is *great* for interactive data visualization (via `leaflet` or `shiny`… more on this on Thursday)
- R is *okay* at spacial data visualization (creating maps).
  - There are many different packages in `R` for creating maps. I've found that different packages perform best for different maps. We will talk about a few different ones today.
  - If you have a highly map-centric project, there is nothing wrong with working in ArcGIS or QGIS if you find the mapping tools in R insufficient. There are many recent improvements with new packages (like `sp`, `rgdal` and `rgeos`) which profiles much of the functionality of GIS packages! Exciting! (not very beginner friendly - requires familiarity with GIS concepts)

Using the `maps` package

Perhaps the simplest approach to drawing maps is to use `geom_polygon()` to draw boundaries for different regions.

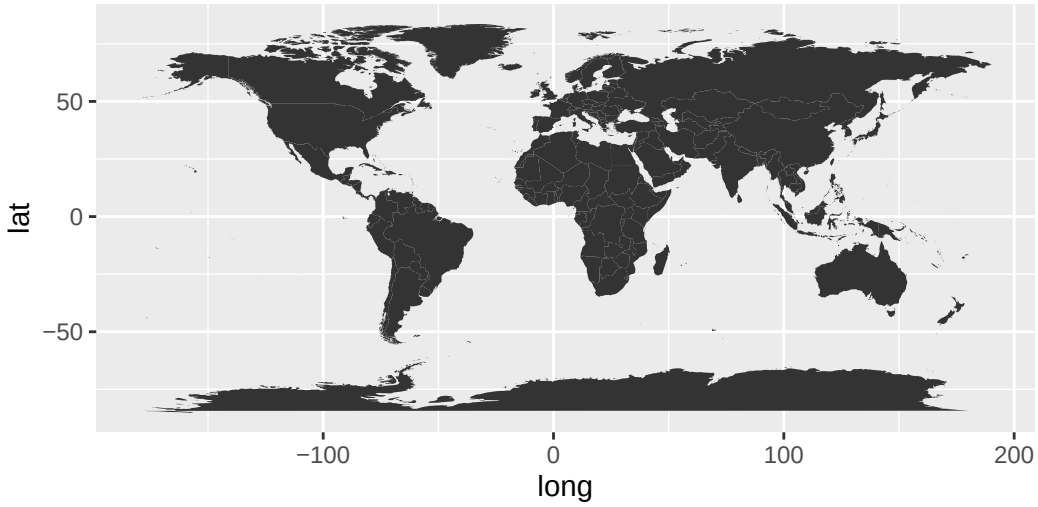
The `maps` package contains several built in maps: world (for all countries in the world), france, italy, nz, usa, state (usa state boundaries), and county (usa counties). The maps package isn't particularly accurate or up-to-date, but it's built into R so it's an easy place to start.

To reference each map you use `map_data("mapname")`.

```
#LOAD PACKAGES
library(tidyverse)
library(maps)
```

```r
#LOAD DATA
world_map <- map_data("world")
```

```r
#World Map
ggplot(world_map, aes(long, lat, group=group)) +
  geom_polygon() +
  coord_quickmap()
```



Note:

- `coord_quickmap()` adjusts the axes to ensure that longitude and latitude are rendered on the same scale. It is very important that this aspect ratio is maintained or a country may appear super stretched or super squished.
- the `aes(group=group)` option – This is SUPER IMPORTANT, so R knows which things to connect together

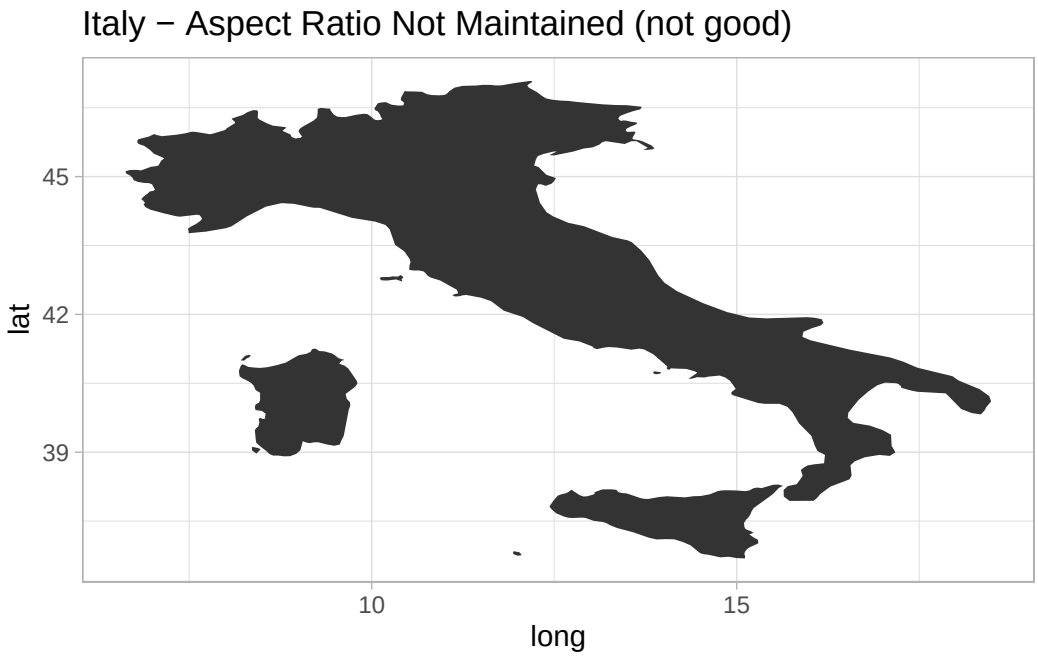
What about subsetting the data?

```
#Subset to get Italy
italy <- map_data("world", region ="Italy")

#Subset to get USA
usa <- map_data("world", region ="USA")
```

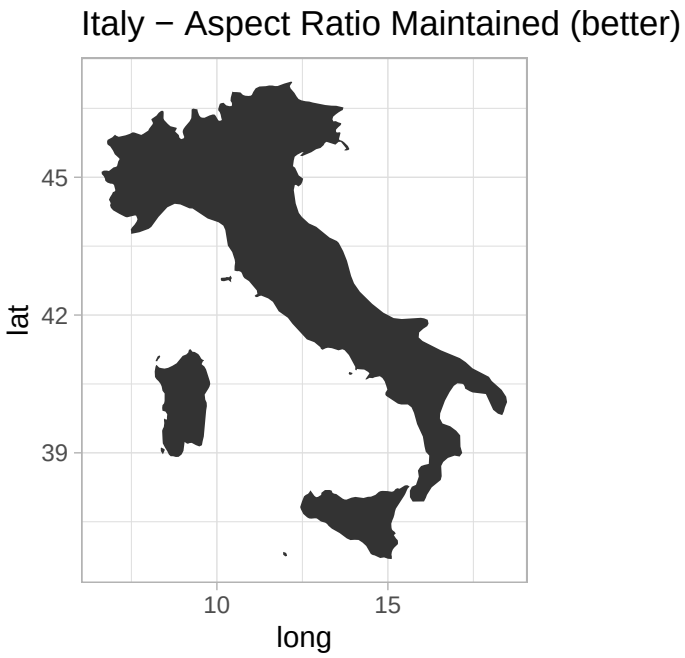
What if aspect ratio is not maintained?

```
# ASPECT RATIO NOT MAINTAINED
ggplot(italy, aes(long, lat)) +
  geom_polygon(aes(group=group)) +
  theme_light() +
  ggtitle("Italy - Aspect Ratio Not Maintained (not good)")
```




```
# ASPECT RATIO MAINTAINED
ggplot(italy, aes(long, lat)) +
  geom_polygon(aes(group=group)) +
```

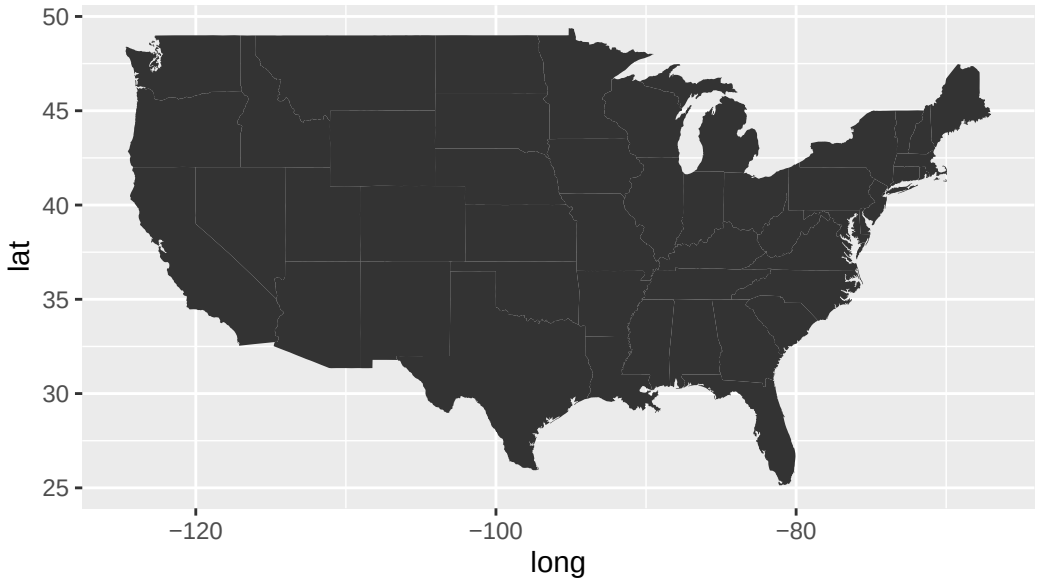
```
  coord_quickmap()  +
  theme_light() +
  ggtitle("Italy - Aspect Ratio Maintained (better)")
```



USA with states

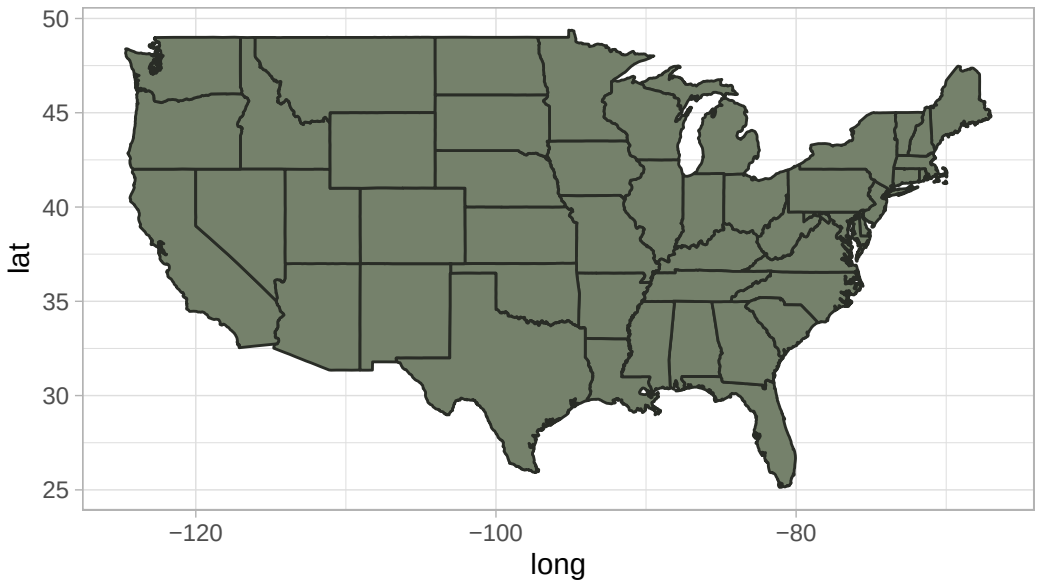
```
#Load Data from maps
usa_states <- map_data("state")

#Plot of USA with state borders
ggplot(usa_states, aes(long, lat)) +
geom_polygon(aes(group=group)) +
coord_quickmap()
```



How to customize colors?

```
ggplot(usa_states, aes(long, lat)) +
geom_polygon(aes(group=group), fill ="#75816b", color ="#292c26") +
coord_quickmap() +
theme_light()
```



Using the `sf` package

There are a few limitations to the approach outlined above, not least of which is the fact that the simple "longitude-latitude" data format is not typically used in real world mapping. Vector data for maps are typically encoded using the "simple features" standard produced by the Open Geospatial Consortium. The `sf` package developed by Edzer Pebesma provides an excellent toolset for working with such data, and the `geom_sf()` and `coord_sf()` functions in ggplot2 are designed to work together with the `sf` package.

```
#LOAD PACKAGES
#install.packages("sf") - note some students are getting a pop-up when they install the sf
library(sf)
```

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

```
#some students are needing into install the rgeos package seperately as well
#library(rgeos)
```

For our first example, we will be working with a dataset of North Carolina that is built in to the `sf` package.

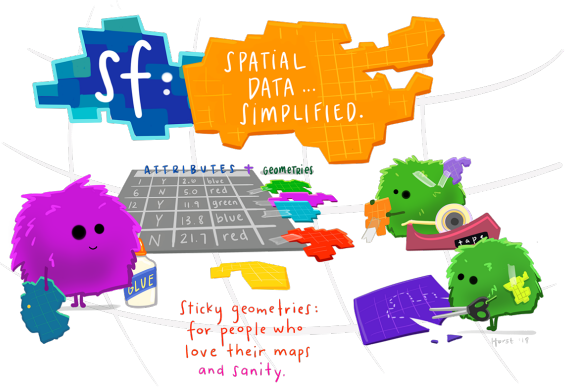


Figure 1: Artwork by @allisonhorst

```
demo(nc, ask = FALSE, echo = FALSE)
```

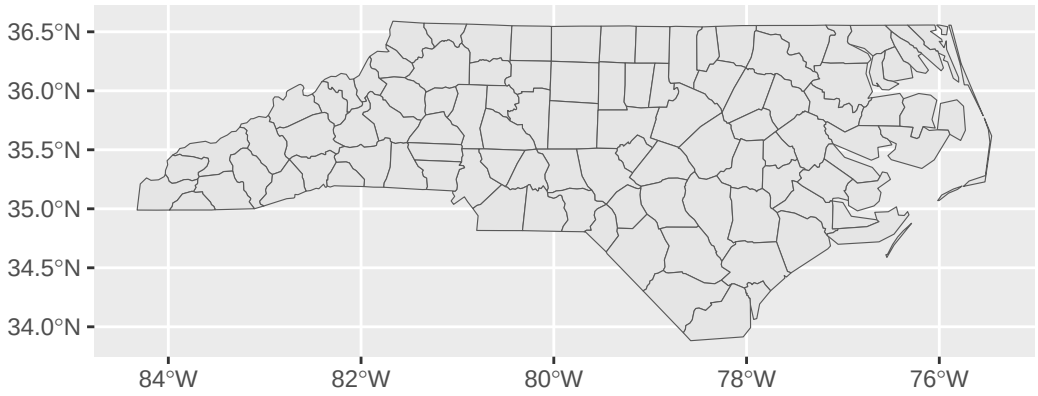
You should notice that the `nc` dataset is now saved in your R environment. This dataset contains information about Sudden Infant Death Syndrome (SIDS) for North Carolina counties, over two time periods (1974-78 and 1979-84). Let's take a look at that dataset.

Each row represents a county in North Carolina. This data frame contains the following columns:

- `AREA` County polygon areas in degree units
- `PERIMETER` County polygon perimeters in degree units
- `CNTY_` Internal county ID
- `NAME` County names
- `FIPS` County ID
- `FIPSNO` County ID
- `CRESS_ID` Cressie papers ID
- `BIR74` births, 1974-78
- `SID74` SID deaths, 1974-78
- `NWBIR74` non-white births, 1974-78
- `BIR79` births, 1979-84
- `SID79` SID deaths, 1979-84
- `NWBIR79` non-white births, 1979-84
- `geom` information needed to plot the map for each county

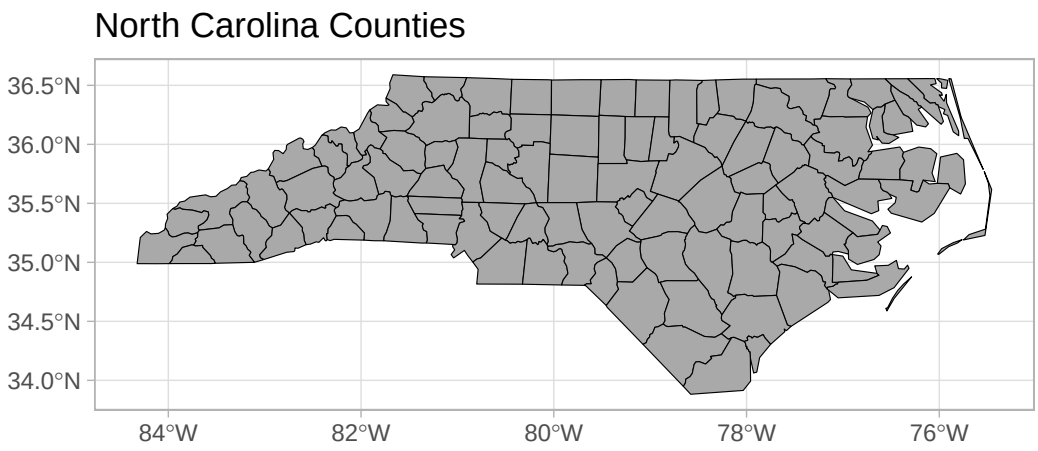
Let's begin by simply plotting the map:

```
nc %>%
  ggplot() +
  geom_sf()
```



Let's pretty it up:

```
nc %>%
  ggplot() +
  geom_sf(col="black", fill="darkgrey") +
  theme_light() +
  ggtitle("North Carolina Counties")
```

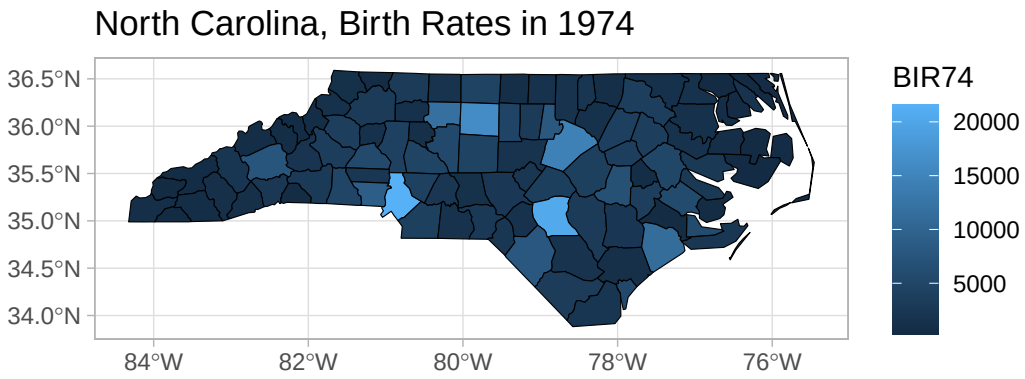


Cloropleth maps

Suppose we want to shade each of these counties, based on the number of births in 1974.

This is called a "cloropleth" map (a map that uses differences in shading, coloring, or the placing of symbols within predefined areas to indicate the average values of a property or quantity in those areas).

```
nc %>%
  ggplot() +
  geom_sf( aes(fill = BIR74), col ="black") +
  theme_light()+
  ggtitle("North Carolina, Birth Rates in 1974")
```

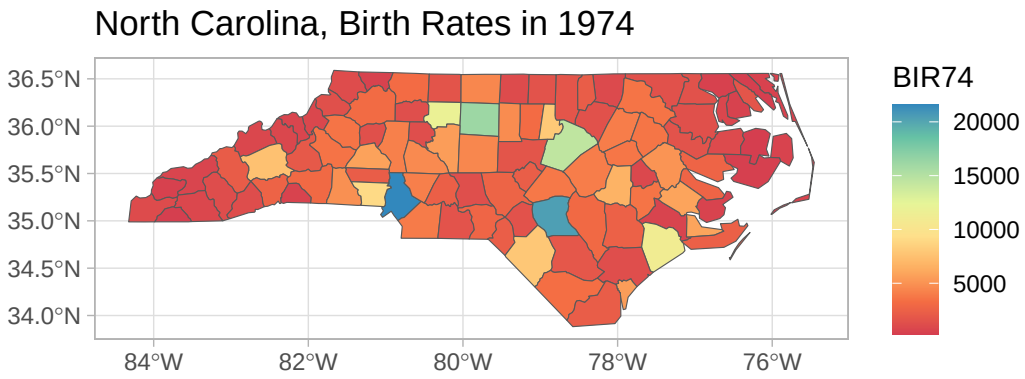


Here are some options to customize the plot that you might be interested in:

Using RColorBrewer palette

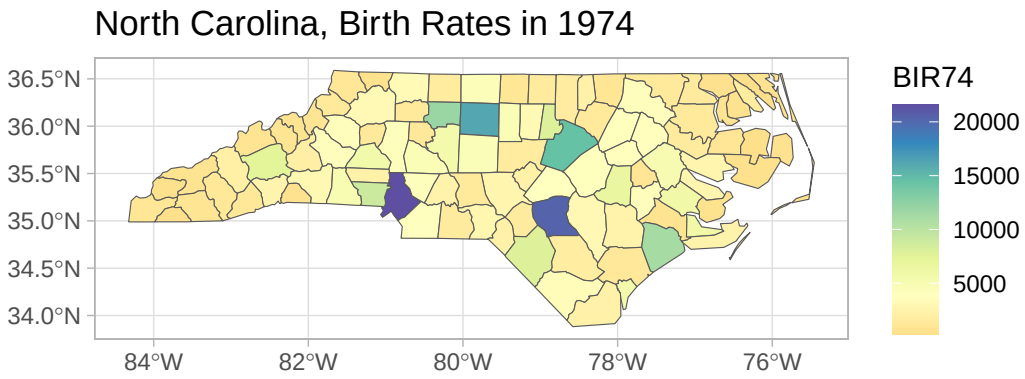
```
library(RColorBrewer)

nc %>%
  ggplot() +
  geom_sf(aes(fill = BIR74)) +
  ggtitle("North Carolina, Birth Rates in 1974") +
  scale_fill_gradientn(colors = brewer.pal(8,"Spectral") ) + #customize colors
  theme_light()
```



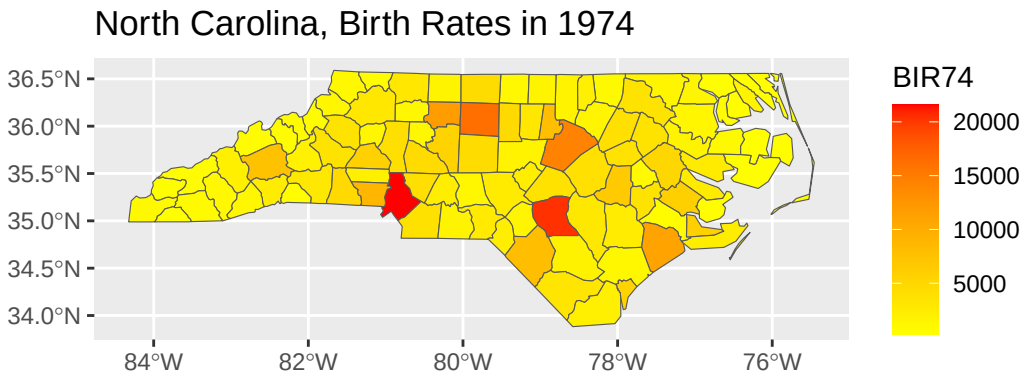
Using part of a RColorBrewer palette

```
nc %>%
  ggplot() +
  geom_sf(aes(fill = BIR74)) +
  ggtitle("North Carolina, Birth Rates in 1974") +
  scale_fill_gradientn(colors = brewer.pal(11,"Spectral")[5:11] ) + #customize colors
  theme_light()
```



Building your own color palette using `scale_fill_gradientn`

```
nc %>%
  ggplot() +
  geom_sf(aes(fill = BIR74)) +
  ggtitle("North Carolina, Birth Rates in 1974") +
  scale_fill_gradientn(colors = c("yellow","orange","red"))
```

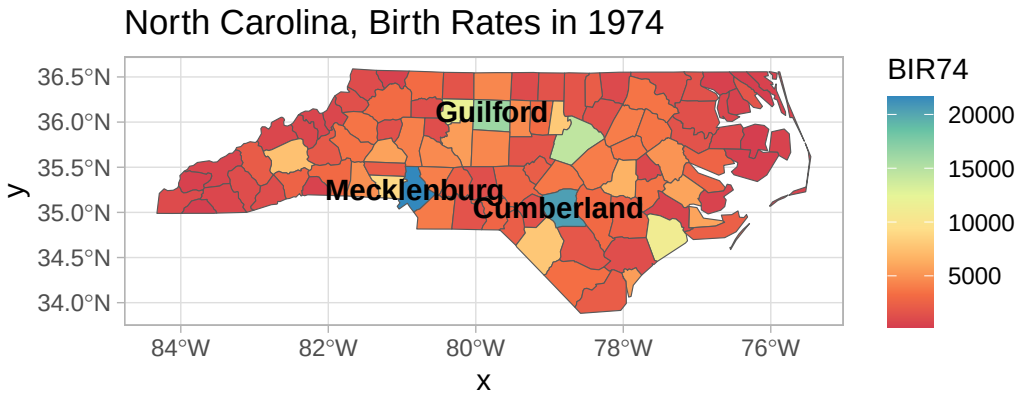


A note about customizing colors:

- you should use a color scheme that is sequential (has order to it), when you are displaying continuous data
- you should use a color scheme that is categorical, when your data is in categories and isn't ordered you should use a color scheme that is diverging, when want to put emphasis on two extremes and mid-range. For example, you might use a diverging palette from red to blue for political party affiliation in the US.
- pay attention to your map being color blind friendly (`RdYlGr` is the worst…)
- as a general rule, try not to use blue to represent a land mass (let's reserve that for bodies of water)

Adding labels

```
ggplot(nc) +
  geom_sf() +
  aes(fill = BIR74) +
  ggtitle("North Carolina, Birth Rates in 1974") +
  scale_fill_gradientn(colors = brewer.pal(8, "Spectral") ) +  #customize colors
  theme_light() +
  geom_sf_text(data = nc[nc$BIR74 >15000,], aes(label = NAME), fontface="bold")
```



`sf` cheatsheet

`sf` cheatsheet