Lab 8: Maps and Spatial Data

2024-05-23

Creating Maps in R

The maps package

```
# load packages
library(pacman)
p_load(maps)
```

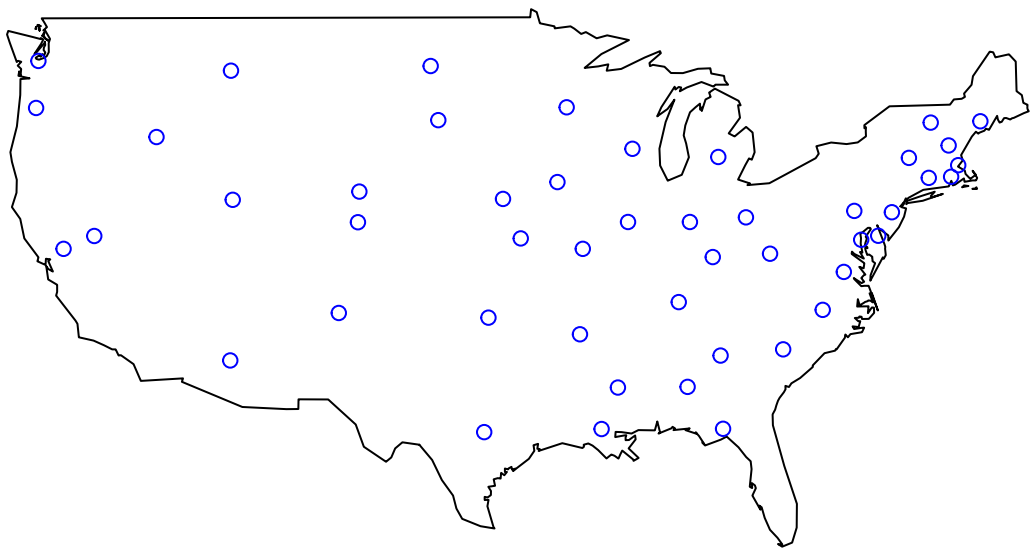
U.S. City Data

The `us.cities` dataset in the `maps` package contains information on the location of about 1000 cities in the U.S. along with information on population and whether the city is a state capital

```
head(us.cities)
```

```
##         name country.etc    pop   lat    long capital
## 1 Abilene TX          TX 113888 32.45  -99.74       0
## 2   Akron OH          OH 206634 41.08  -81.52       0
## 3 Alameda CA          CA  70069 37.77 -122.26       0
## 4  Albany GA          GA  75510 31.58  -84.18       0
## 5  Albany NY          NY  93576 42.67  -73.80       2
## 6  Albany OR          OR  45535 44.62 -123.09       0
```

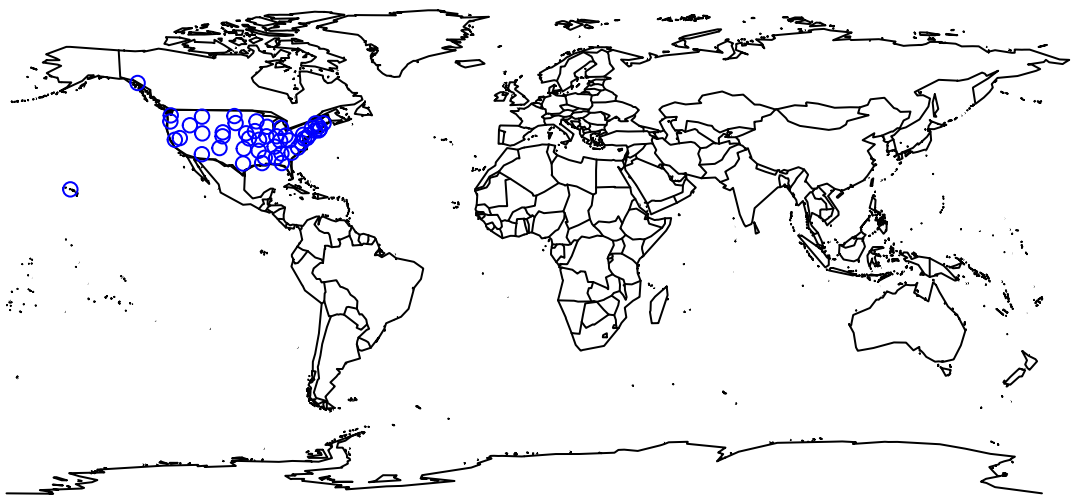
```
# Select the subset of cities that are state capitals
capitals = subset(us.cities, capital == 2)


# Start a map with the map() function
map(database = "usa")
  # to apply something to this map, start a function in the next line
points(x = capitals$long, y= capitals$lat, col = "blue")
  # and another
title("US state capitals")
```




```
  # it does this until there's a function that doesn't apply
```

We can *add* a map on top of another map using `add = T` in another `map()` function below the first one (note that `add = F` is the default). For example, let's overlap the U.S. capitals map on top of the world map:

```
map(database = "world")
map(database = "usa", add = T)
  # to apply something to this map, start a function in the next line
points(x = capitals$long, y= capitals$lat, col = "blue")
title("US state capitals in the World")
```



You may notice that this map includes Hawaii and Alaska, when the previous map did not.

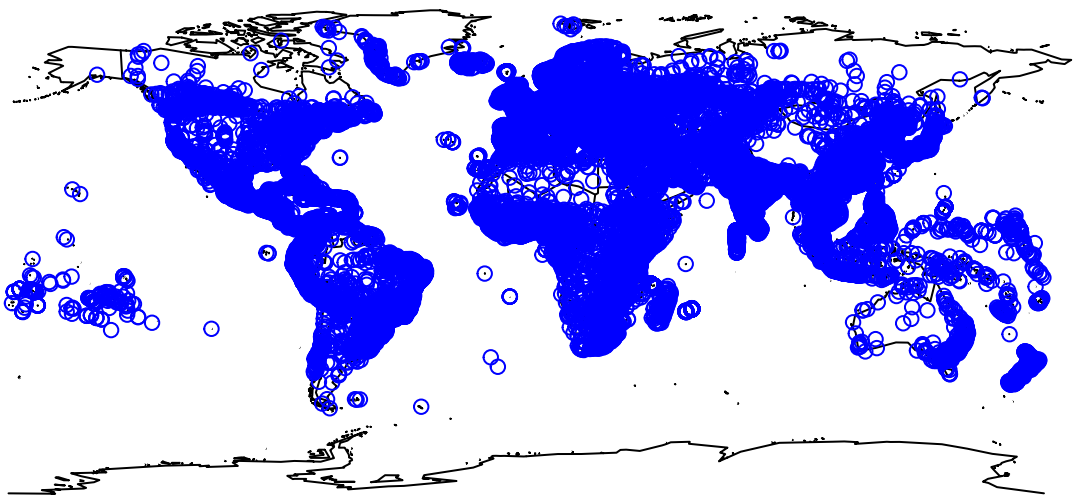
Other Datasets in the `maps` Package

The WORLD

```
# Data of cities around the world
head(world.cities)
```

```
##                 name country.etc   pop   lat  long capital
## 1 'Abasan al-Jadidah   Palestine  5629 31.31 34.34       0
## 2 'Abasan al-Kabirah   Palestine 18999 31.32 34.35       0
## 3       'Abdul Hakim    Pakistan 47788 30.55 72.11       0
## 4 'Abdullah-as-Salam      Kuwait 21817 29.36 47.98       0
## 5              'Abud   Palestine  2456 32.03 35.07       0
## 6            'Abwein   Palestine  3434 32.03 35.20       0
```

```
#
map(database = "world")
points(x = world.cities$long, y= world.cities$lat, col = "blue")
title("Map of Cities in maps Package")
```

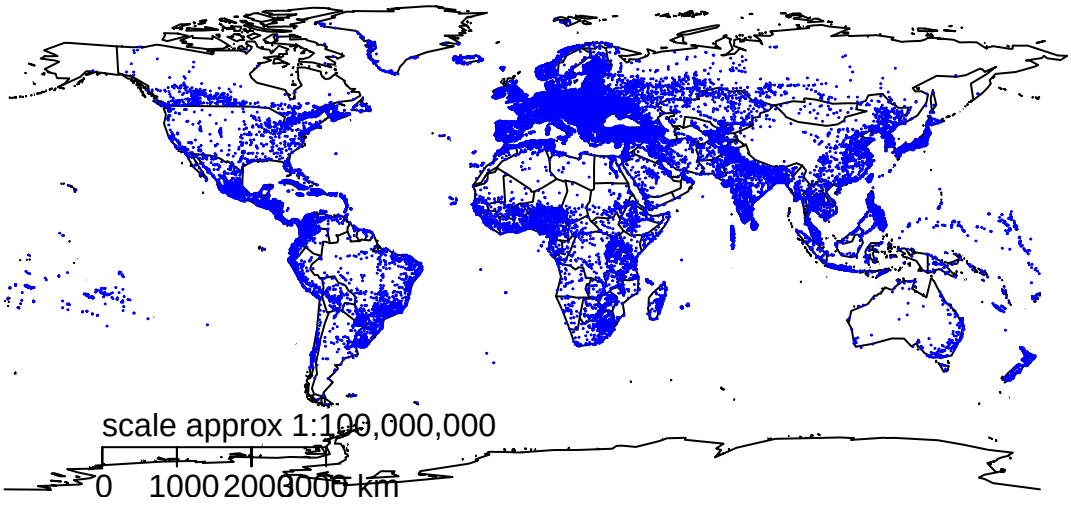


We can change the look of the dots within the `points()` function:

- `pch = 1, 2, ...` The difference numbers determine the shape. Lots of strange shapes that make you wonder "Who would ever use this and why?" You can find the options here
- `cex = 1` Continuous variable that determines the size of the city point on the map

Let's recreate that map from above but make the points filled in circles (`pch = 20`) and make the points much smaller (`cex = 0.1`)

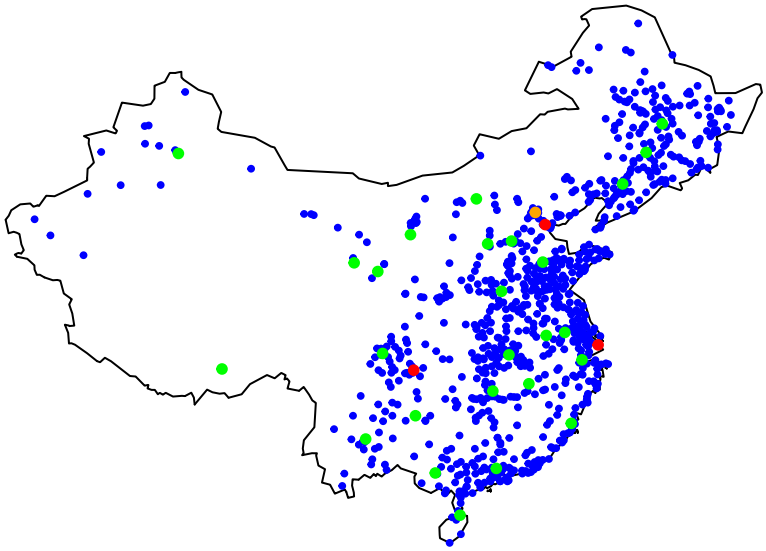
```
map(database = "world")
points(x = world.cities$long, y= world.cities$lat, col = "blue",
       pch = 20, cex = 0.1)
  # also try cex = 0.01
  # add scale
map.scale()
```



China

China has specific labels for cities where: 0 for non-capital, 1 for capital, 2 for China Municipalities, and 3 for China Provincial capitals

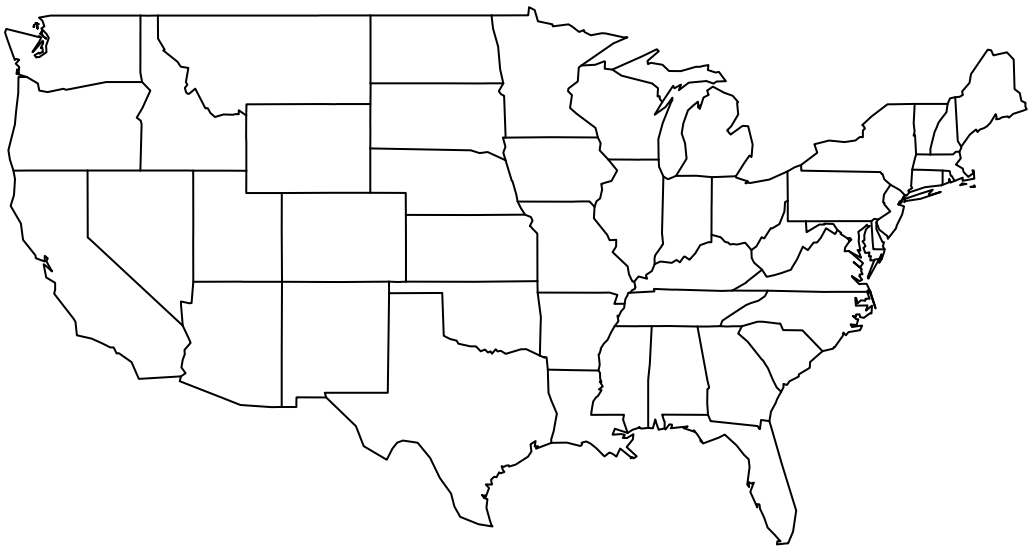
```
# Tell it the scope of your map
map("world", "china")
  # non capital cities
map.cities(country = "China", capitals = 0,
      pch=20, col="blue")
  # Provincial capitals
map.cities(country = "China", capitals = 3,
      pch=20, col="green")
  # Municipal Capitals
map.cities(country = "China", capitals = 2,
      pch=20, col="red")
  # Country capital
map.cities(country = "China", capitals = 1,
      pch=20, col="orange")
```



State Data

Back to the U.S.

```
# We can look at all of the (lower 48, sorry Hawaii and Alaska) states
map(database = "state")
```



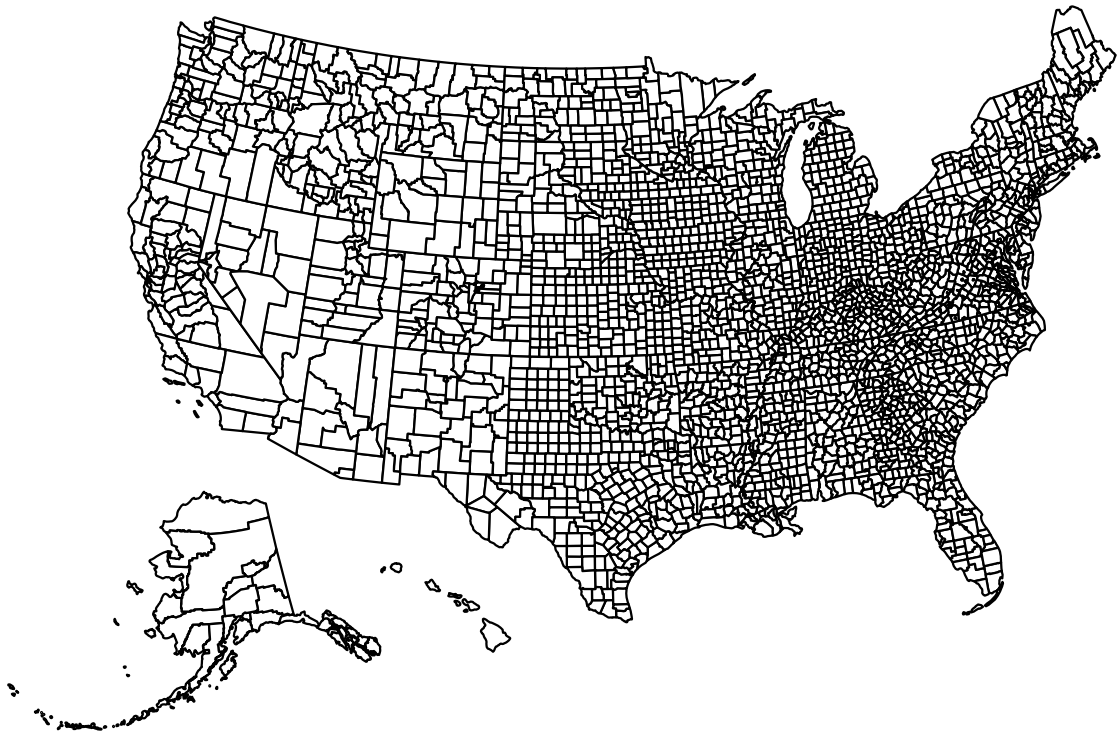
```r
# We can get specific regions:
map(database = "state", regions= "Oregon")
```

```
map(database = "state", regions= c("Oregon", "Washington", "California"))


# Let's plot the most populous city in each of these 3 states:
  # first:
p_load(dplyr)

  # second:
west_city = us.cities %>%
    # filter to the states we want
  filter(country.etc %in% c("OR", "WA", "CA")) %>%
    # group by each state to find the most populous
  group_by(country.etc) %>%
    # get the most populous city in each state
  slice_max(order_by = pop, n = 1) %>%
      # change n = 1 to 2 or 3 to get the top 2 or 3 most populous
  ungroup()

  # Now plot:
map(database = "state", regions= c("Oregon", "Washington", "California"))
points(x = west_city$long, y = west_city$lat,
       col = "blue", pch = 20)
```

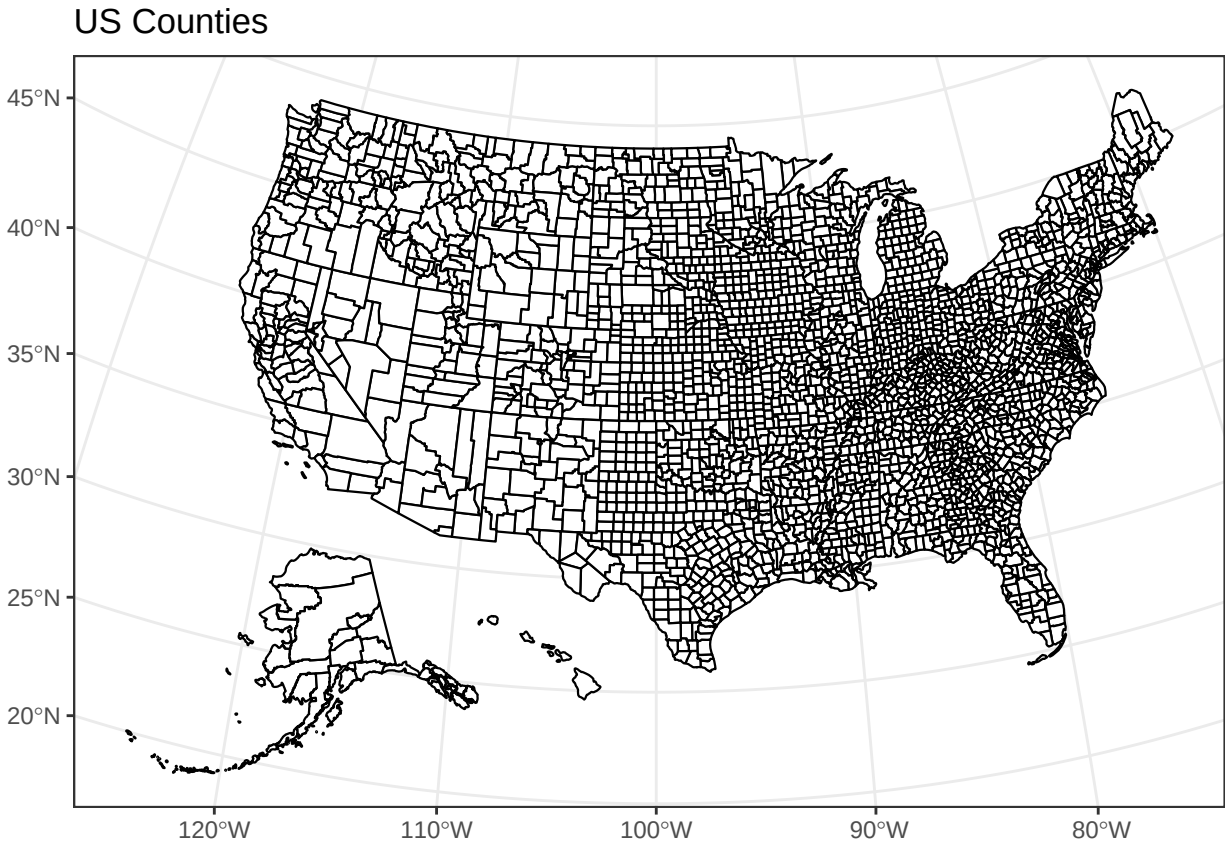
Also the `usmap` Package

The `usmap` package gives you more range of options (including county maps) and works in tandum with `ggplot2`, a package we're familiar with.

```
# Load packages
p_load(usmap, ggplot2)

# Plain map
plot_usmap(regions = "counties")
```



```
# Add some ggplot elements
plot_usmap(regions = "counties") +
  labs(title = "US Counties") +
  theme_bw()
```



Kind of misleading with Hawaii and Alaska in relation to the longitude and latitude labels, but it's cool.

Showing Data

Now let's incorporate economic data into these maps!

Go to the Github website and scroll down to Week 9 and click on "Spatial Data" and then "raw" and copy the URL. This is inflation data about all US states that comes from here

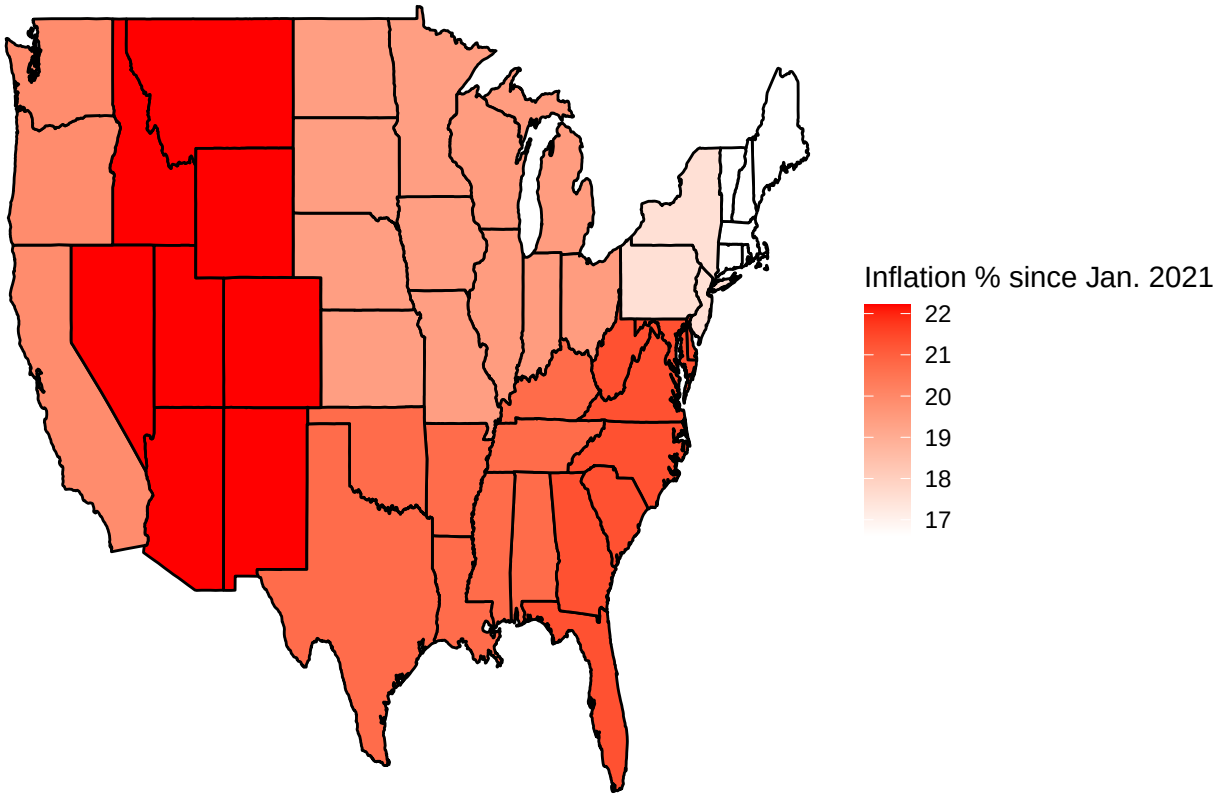
```
# recall from webscraping lab
p_load(readr)

# URL
urlfile = "https://raw.githubusercontent.com/cmulholland217/Metrics_Lab_Spring2025/refs/heads/main/stat

# the data
inflation = read_csv(url(urlfile))
```

```
## Rows: 52 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): State, State Code, Month
## dbl (8): Increase in Prices (%) Since January 2021, Total Monthly Inflation ...
##
## 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.
```

We are going to select the columns labeled “State Code” and “Increase in Prices (%) since January 2021,” and then merge with the “state” data from the packages we used above. Then we will make a *heat map* of the states where a darker red signifies higher inflation.

```
inflation = inflation %>%
    # select the 2nd and 4th columns
              select(1,4) %>%
    # rename columns
              rename_all(~c("state", "inflation")) %>%
    # make state names lowercase
              mutate(state = tolower(state))

# get spatial data about states
state_data = map_data("state")

# Merge datasets
  # note the different names for states in the datasets
state_map = merge(x = inflation, y = state_data, by.x = "state", by.y = "region")



# Plot in ggplot:
ggplot(state_map, aes(x = long, y = lat, group = group, fill = inflation)) +
  geom_polygon(color = "black") +
  scale_fill_continuous(low = "white", high = "red", name = "Inflation % since Jan. 2021") +
  theme_void()
```


Inflation % since Jan. 2021
22
21
20
19
18
17