Data Viz III: Geographic Mapping with ggplot2

Dr. Adrian Correndo

2025-02-07

Description

In this class, we will explore how to create geographic maps using `ggplot2`, `sf`, `maps`, `leaflet`, and `geojson`. We'll cover techniques for plotting data points on US and Canada maps, customizing map aesthetics, and working with spatial data.

1 Learning Objectives

By the end of this session, you will: 1. Load and visualize geographic data using `sf` and `maps`. 2. Create US and Canada maps using `ggplot2`. 3. Overlay data points on maps using geographic coordinates. 4. Customize map aesthetics for professional presentations. 5. Work with shapefiles and GeoJSON for more detailed geographic visualizations. 6. Implement a function for plotting. 7. Create an interactive map using `leaflet`.

Required Packages

Ensure the following packages are installed:

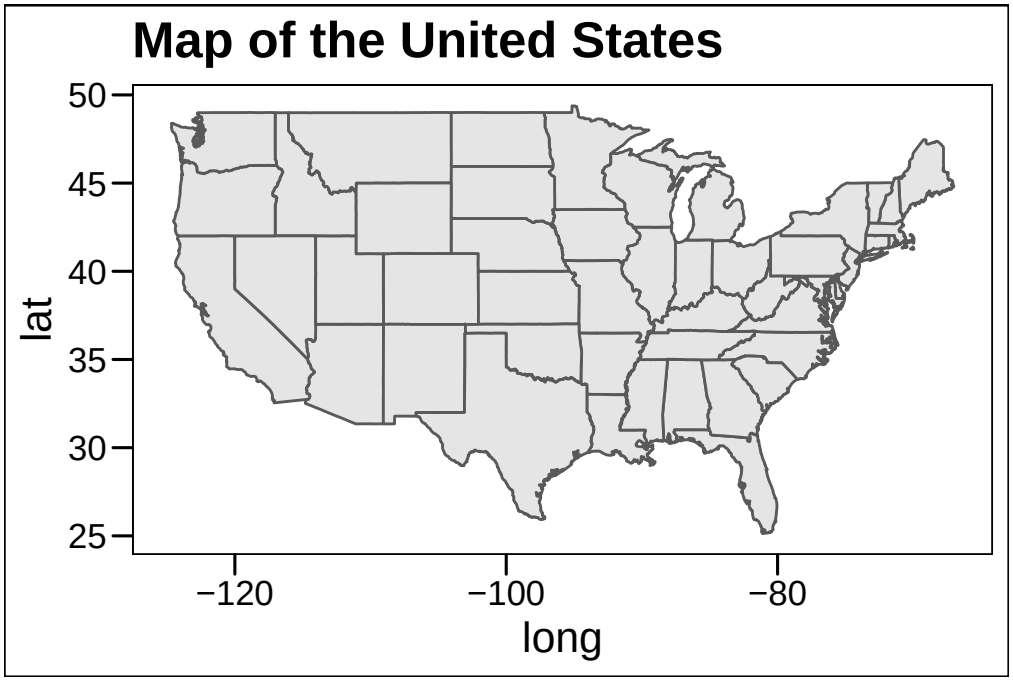
```
library(pacman)
p_load(ggplot2, sf, maps, dplyr,
       ggthemes, ggrepel, leaflet, geojsonio,
       ggspatial, ggpubr)
```

2 Introduction to Mapping with `ggplot2` and `maps`

We will start by creating basic maps of the US and Canada using the `maps` package.

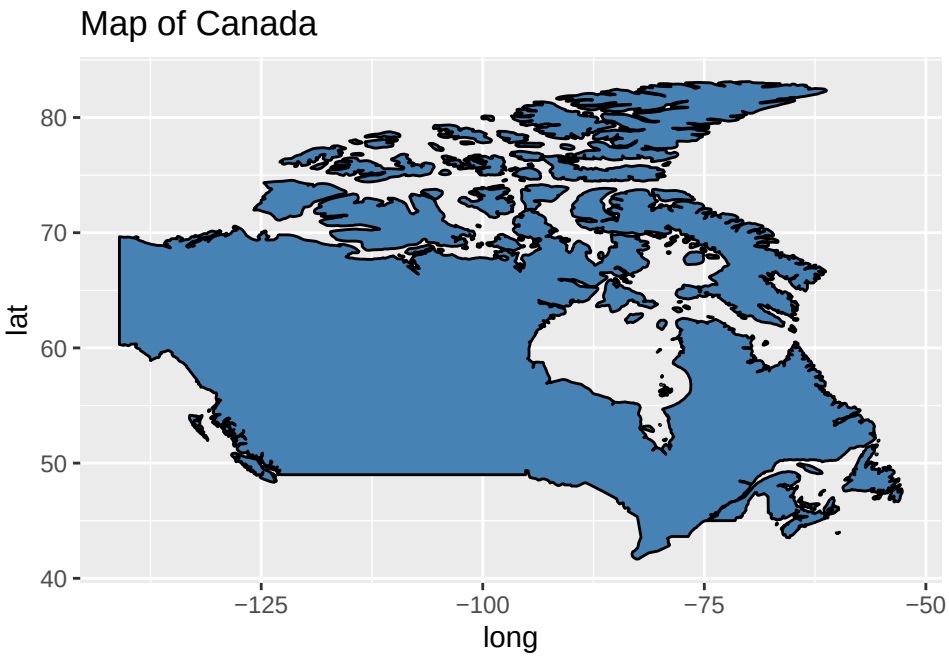
2.1 US Map with State Boundaries

```
# Load US map data
us_map <- map_data("state")

# Plot US map
us_plot <-
  ggplot() +
  geom_polygon(data = us_map, aes(x = long, y = lat, group = group),
               fill = "grey90", color = "grey35") +
  coord_fixed(1.3) +
  labs(title = "Map of the United States") +
  theme_base()

us_plot
```

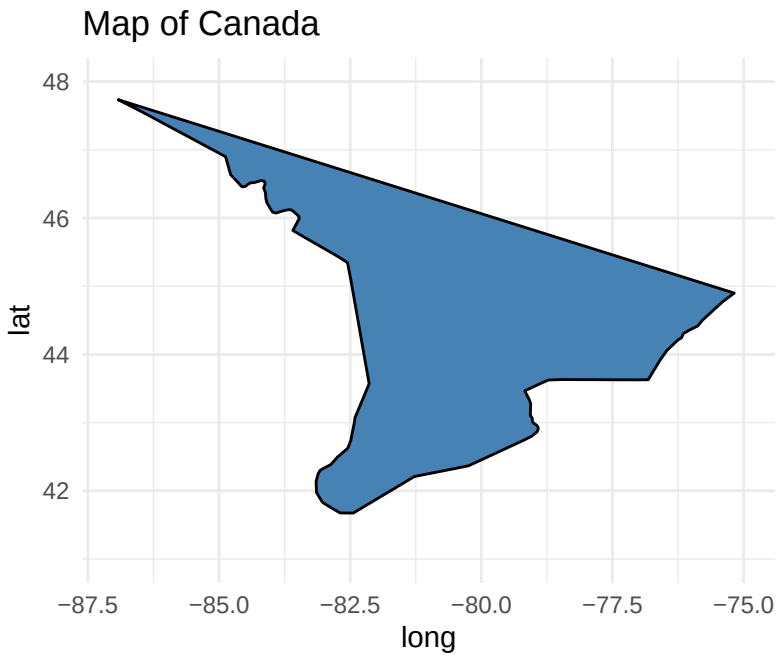


2.2 Canada Map with Provincial Boundaries

```
# Load Canada map data
canada_map <- map_data("world", region = "Canada")

# Plot Canada map
canada_plot <- ggplot() +
  geom_polygon(data = canada_map, aes(x = long, y = lat, group = group),
               fill = "steelblue", color = "black") +
  coord_fixed(1.3) +
  labs(title = "Map of Canada")

canada_plot
```




```
canada_cut <- canada_plot +
  # Cut limits of map
  scale_y_continuous(limits = c(41, 48))+
  scale_x_continuous(limits = c(-87, -75))+
  theme_minimal()

canada_cut
```



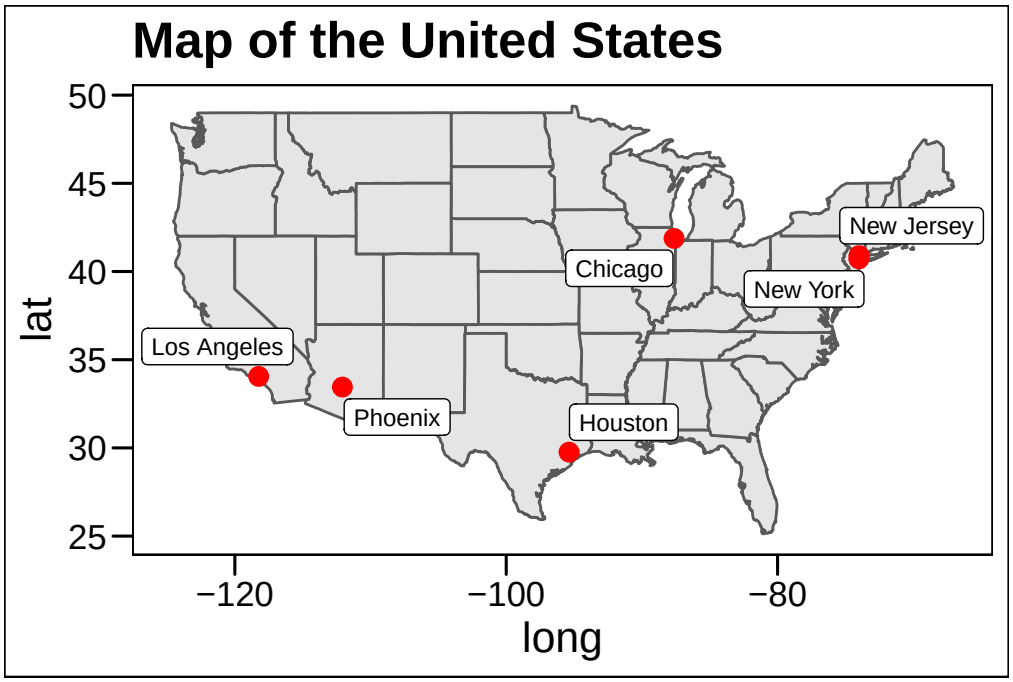
3 Plotting Data Points on Maps

Next, we'll plot specific data points (e.g., cities) on the US and Canada maps.

3.1 Plotting Cities on US Map

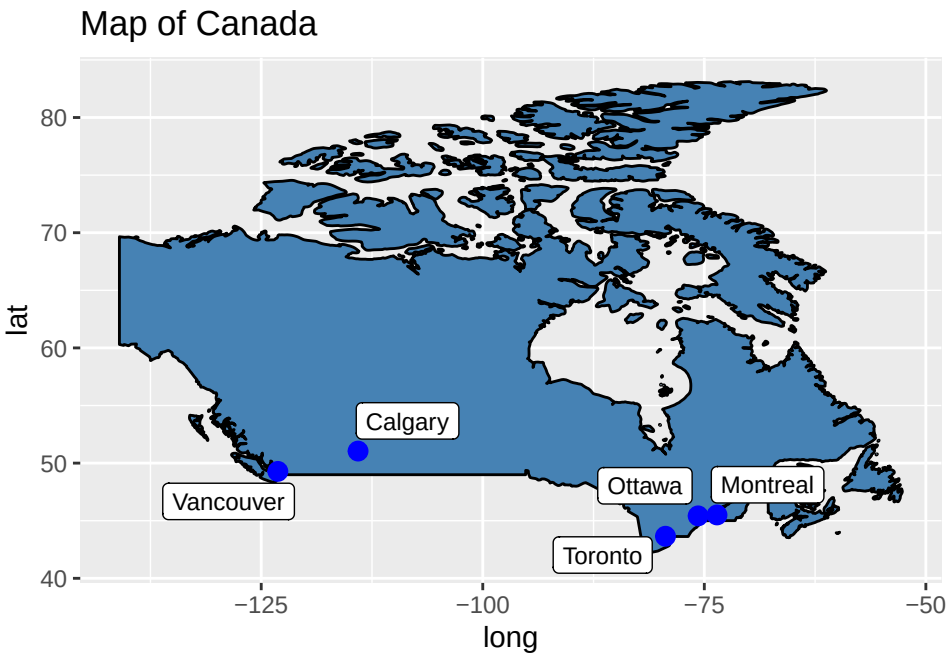
```
# Sample city data
cities_us <- data.frame(
  city = c("New Jersey", "New York", "Los Angeles", "Chicago", "Houston", "Phoenix"),
  lon = c(-74, -74.006, -118.2437, -87.6298, -95.3698, -112.074),
  lat = c(40.9, 40.7128, 34.0522, 41.8781, 29.7604, 33.4484)
)

# Plot US map with cities
us_plot +
  geom_point(data = cities_us, aes(x = lon, y = lat),
             color = "red", size = 3) +
  # geom_label(data = cities_us, aes(x = lon, y = lat, label = city),
  #                  size = 3, color = "black")
```

```
  geom_label_repel(data = cities_us, aes(x = lon, y = lat, label = city),
                   size = 3, color = "black")
```



3.2 Plotting Cities on Canada Map

```
# Sample city data for Canada
cities_canada <- data.frame(
  city = c("Toronto", "Vancouver", "Montreal", "Calgary", "Ottawa"),
  lon = c(-79.3832, -123.1216, -73.5673, -114.0719, -75.6972),
  lat = c(43.6511, 49.2827, 45.5017, 51.0447, 45.4215)
)

# Plot Canada map with cities
canada_plot +
  geom_point(data = cities_canada, aes(x = lon, y = lat),
             color = "blue", size = 3) +
  geom_label_repel(data = cities_canada, aes(x = lon, y = lat, label = city),
                   size = 3, color = "black")
```



4 Mapping with Shapefiles using `sf`

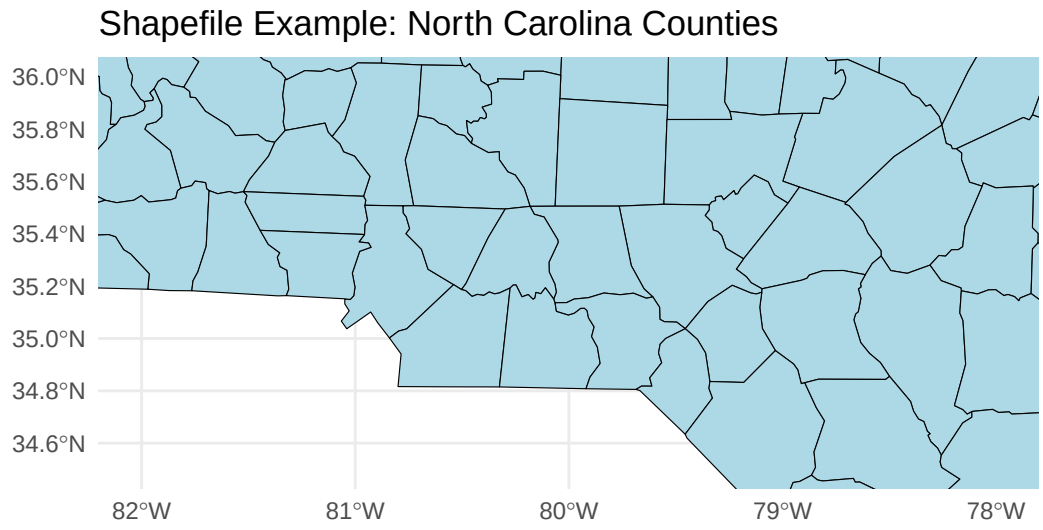
For more detailed geographic visualizations, we can use shapefiles with the `sf` package.

4.1 Loading and Plotting Shapefiles

```
# Load shapefile (replace 'path_to_shapefile' with your actual path)
# usa_shapefile <- st_read("path_to_shapefile/usa_shapefile.shp")

# Example using built-in dataset from `sf`
nc <- st_read(system.file("shape/nc.shp", package = "sf"))
```

```
Reading layer `nc' from data source
  `/Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library/sf/shape/nc.shp'
  using driver `ESRI Shapefile'
Simple feature collection with 100 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
Geodetic CRS:  NAD27
```

```
# Plot shapefile
ggplot(nc) +
  geom_sf(fill = "lightblue", color = "black") +
  labs(title = "Shapefile Example: North Carolina Counties") +
  # Adjust scales for limits
  scale_y_continuous(limits = c(34.5, 36))+
  scale_x_continuous(limits = c(-82, -78))+
  theme_minimal()
```



4.2 Read shp of Canada and US

```
## Load shp files
usa_shp <- st_read("shp_map/us/usa.shp") %>%
  # Remove non-contiguous and territories
  filter(!(NAME %in% c("Alaska", "District of Columbia", "Hawaii", "Puerto Rico")))
```

```
Reading layer `usa' from data source
  `/Users/acorrend/Documents/GitHub/plnt6800/coding/week_05/shp_map/us/usa.shp'
  using driver `ESRI Shapefile'
Simple feature collection with 52 features and 9 fields
```

```
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -179.1473 ymin: 17.88481 xmax: 179.7785 ymax: 71.35256
Geodetic CRS:  NAD83
```

```r
can_shp <- st_read("shp_map/can/canada.shp")
```

```
Reading layer `canada' from data source
  `/Users/acorrend/Documents/GitHub/plnt6800/coding/week_05/shp_map/can/canada.shp'
  using driver `ESRI Shapefile'
Simple feature collection with 13 features and 6 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -141.0181 ymin: 41.67695 xmax: -52.5823 ymax: 89.99943
Geodetic CRS:  NAD83
```

4.3 Create objects for maps

```r
# Create list of selected provinces
selected_provinces <- c("Ontario", "Manitoba", "Quebec")

# Define coordinates for Ontario cities
ontario_cities <- data.frame(
  city = c("Toronto", "Ottawa", "Hamilton", "London", "Kingston"),
  lon = c(-79.3832, -75.6972, -79.8711, -81.2497, -76.4880),
  lat = c(43.6511, 45.4215, 43.2557, 42.9834, 44.2312)
)
```

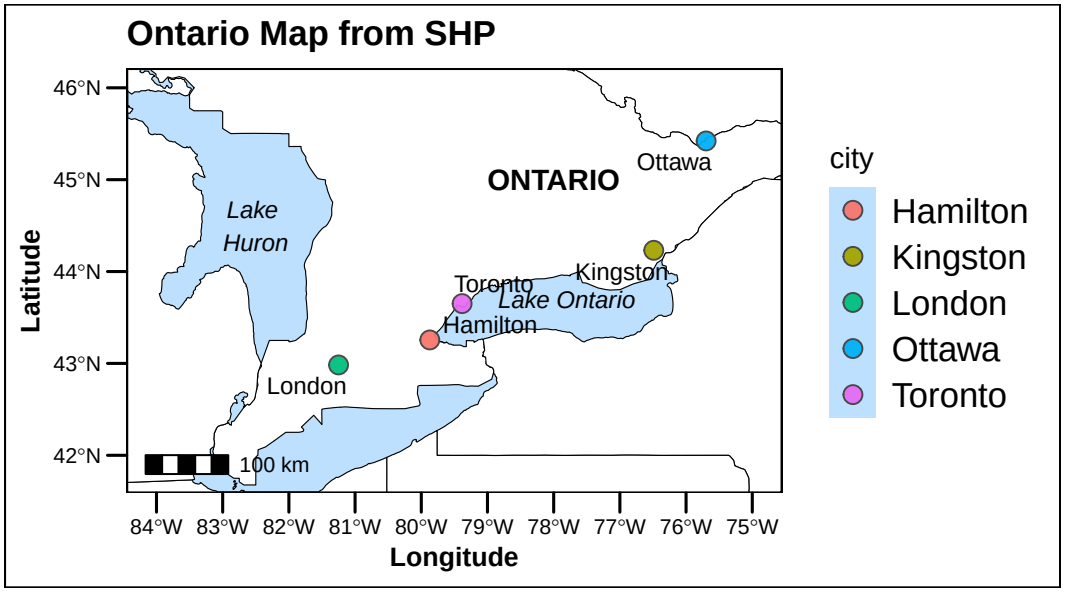
4.4 Define function to customize plot

```r
geo_plot <- function(x, y, z, title = NULL){
  ggplot()+
    geom_sf(data=x, fill = "white", color = "black") + # Provinces map
  geom_sf(data=y, fill = "white", color = "black")+ # US map
  # Adjust scales for lat and lon
  scale_y_continuous(limits = c(41.8, 46))+
  scale_x_continuous(limits = c(-84, -75), breaks = seq(-84,-74, by=1)) +
  # Add cities with points
```

```
  geom_point(data = z, aes(x = lon, y = lat, fill = city),
             color = "grey25", shape = 21, size = 3, alpha = 0.95) +
  # Scalebar
  annotation_scale(tick_height = 0.3)+
  # Text Notes for names of cities
  geom_text_repel(data = z,
                  aes(x=lon, y=lat, label = city), size = 3)+
  # Name of PROVINCE
  annotate("text", x = -78, y = 45, label = "ONTARIO",
           size = 4, fontface = "bold")+
  # Name of Lakes
  ## Ontario
  annotate("text", x = -77.8, y = 43.7, label = "Lake Ontario",
           size = 3, fontface = "italic")+
  ## Huron
  annotate("text", x = -82.5, y = 44.5, label = "Lake \nHuron",
           size = 3, fontface = "italic")+
  # Add labels
  labs(title = title,
       x = "Longitude", y = "Latitude")+
  # Adjust theme
  theme_base()+
  # reduce axis text size
  theme(
    panel.background = element_rect(fill = "#bde0fe"),
    title = element_text(size = rel(.7)),
    axis.title =  element_text(size = rel(.9), face = "bold"),
    axis.text = element_text(size = rel(.5))
    )
}
```

4.5 Plot

```
# Plot from function
geo_plot(x = can_shp, y = usa_shp, z = ontario_cities,
         title = "Ontario Map from SHP")
```



5 Working with GeoJSON Data

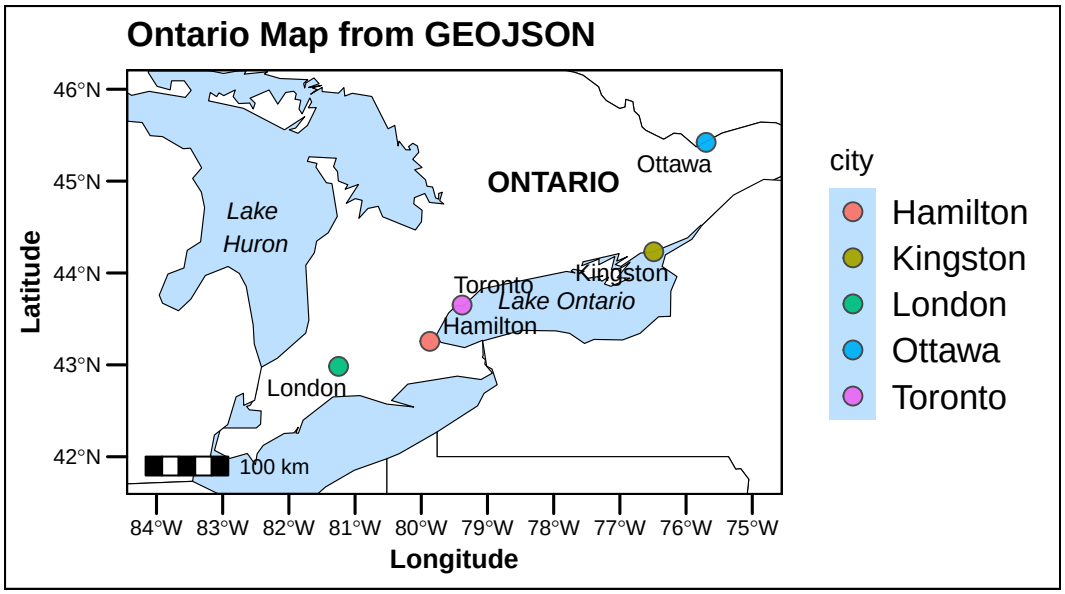
We can use GeoJSON files for geographic data. Here's how to load and visualize a GeoJSON file for Ontario, Canada.

5.1 Loading GeoJSON Data

```
# Load Canada GeoJSON (replace with actual file path if available)
# Read geojson for Canada
can_geojson <- read_sf("geojson_maps/canada_provinces.geojson")  %>%
  # dplyr::filter(name == "Ontario")  %>%
  dplyr::filter(name %in% selected_provinces)  %>%
  dplyr::mutate(Province = name,
                GEOID = cartodb_id)  %>%
  dplyr::select(GEOID, Province, geometry)

# Read geojson for US
usa_geojson <- read_sf("geojson_maps/us-states.json")
```

5.2 Plot GEOJSON

```
# Plot from function
geo_plot(x = can_geojson, y = usa_geojson, z = ontario_cities,
         title = "Ontario Map from GEOJSON")
```



5.3 Explanation:

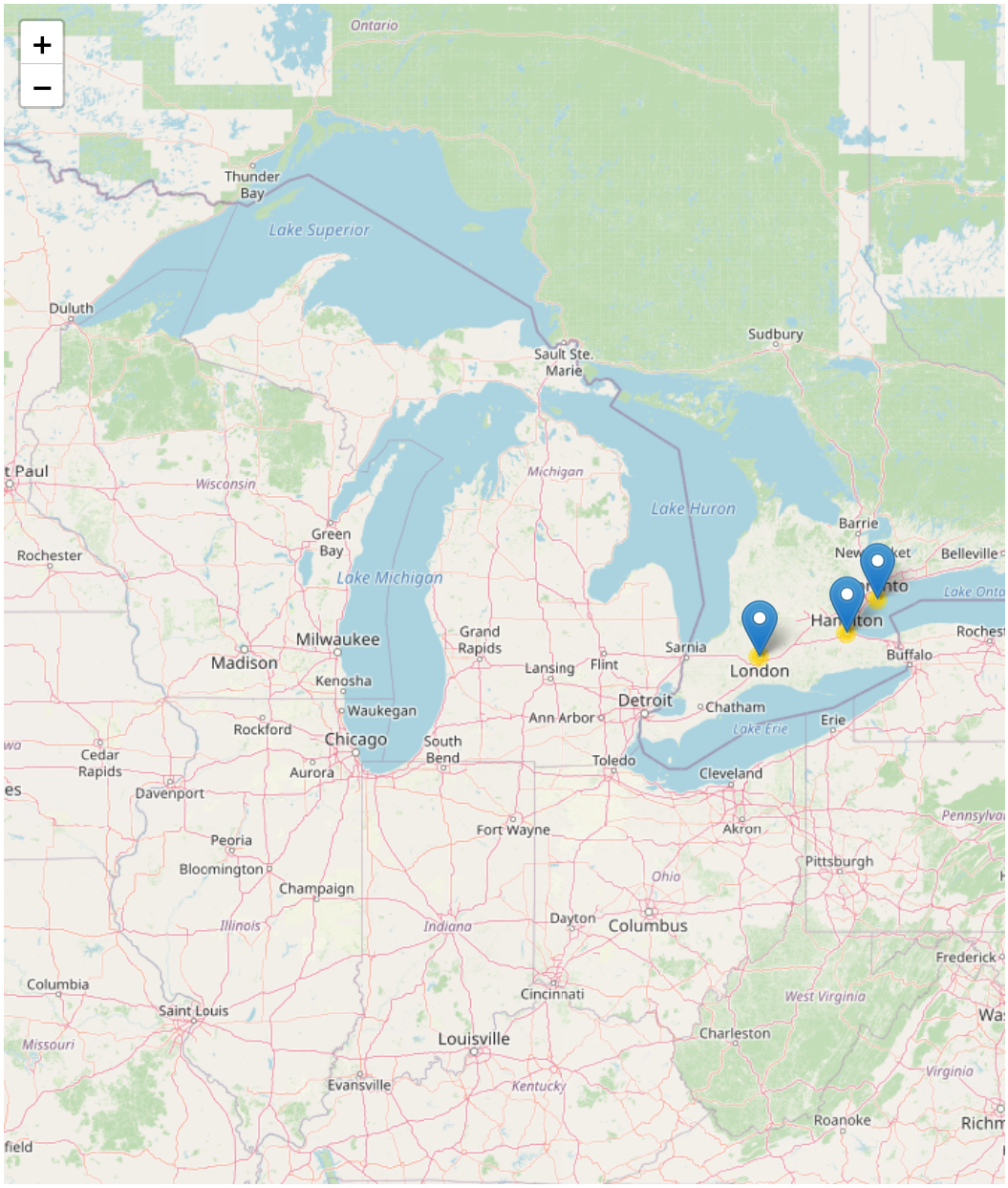
- `geojson_read()`: Reads the GeoJSON file.
- `st_as_sf()`: Converts the data into an `sf` object for plotting.
- `geom_sf()`: Plots the GeoJSON data.

6 Creating Interactive Maps with `leaflet`

`leaflet` allows us to create interactive maps. Let's create a map of Ontario, Canada.

6.1 Ontario Map with `leaflet`

```
# Create interactive map
ontario_cities %>%
leaflet() %>%
  addTiles() %>%
# Add PINS
addMarkers(~lon, ~lat, popup = ~city) %>%
addCircleMarkers(~lon, ~lat, popup = ~city, radius = 5, color = "gold",
                 fillOpacity = 0.7) %>%
# Configure initial view of the map
setView(lng = -80, lat = 44, zoom = 6)
```



6.2 Explanation:

- `addTiles()`: Adds the base map layer.
- `addMarkers()`: Plots city locations with popups displaying city names.
- `setView()`: Centers the map on Toronto with a specified zoom level.

7 Conclusion

In this lesson, you learned how to: - Create geographic maps of the US and Canada using `ggplot2` and `maps`. - Plot data points on maps with latitude and longitude coordinates. - Use shapefiles and GeoJSON for more detailed geographic visualizations with `sf`. - Customize map aesthetics for clearer and more professional presentations. - Create interactive maps using `leaflet`.

These techniques are valuable for visualizing spatial data and can be adapted for a wide range of research applications.

8 Additional Resources

1. **ggplot2 Documentation** – Official documentation for creating maps and more.
2. **sf Package Documentation** – Guide to handling spatial data in R.
3. **R Maps Package** – Reference for basic map data.
4. **Leaflet for R** – Documentation for interactive mapping in R.
5. **GeoJSON Data and Tools** – Create and explore GeoJSON data.
6. **R Spatial Data Science** – Tutorials and resources for spatial data analysis.
7. **R Graph Gallery - Maps** – Examples of different map types using R.

Experiment with different datasets and explore additional map customizations to further enhance your visualizations!