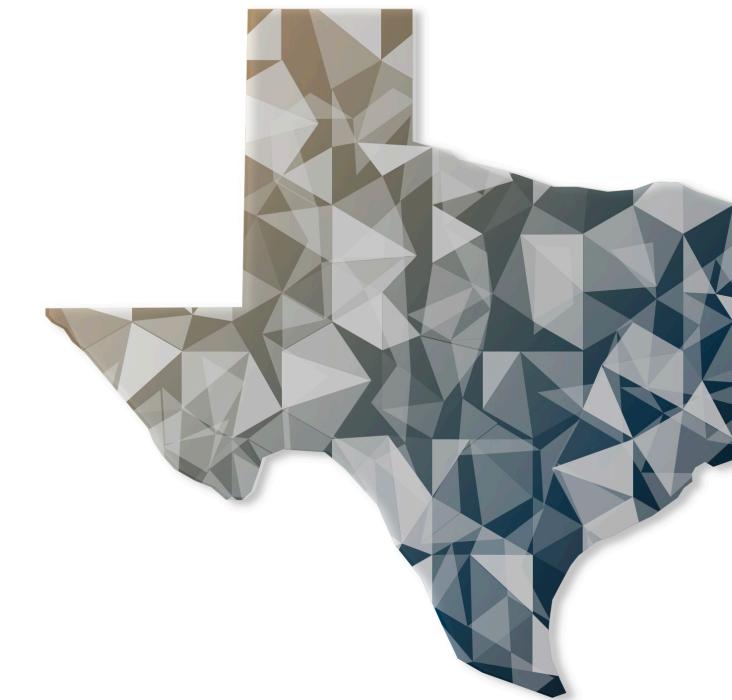


# **WORKSHOP | Spatial Analysis in R**

## **TxGIS Day 2020**

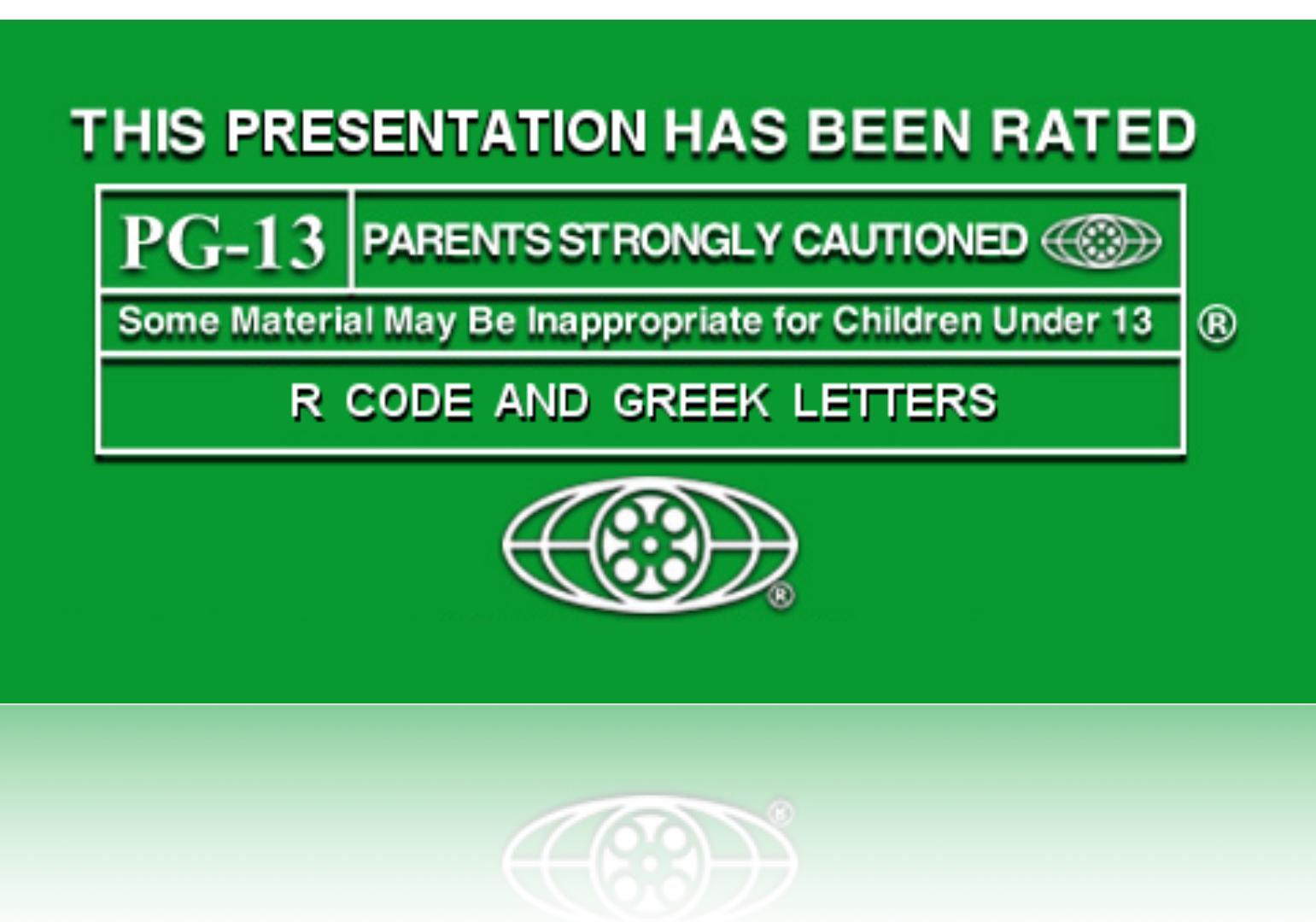


**Alexander Abuabara — Nov 17, 2020, 2:00 PM - 3:30 PM**

# WORKSHOP | Spatial Analysis in R

## Contents of the Workshop

1. Introduction
2. Spatial *raster* data
3. Spatial *vector* data
4. Mapping
5. Hands on “Walking, Cycling, and Driving Distances”
6. (Hands on “Elevation Analysis”)



# About myself

## Alexander Abuabara

- Urban scientist and hazard researcher at Texas A&M University
- PhD candidate in Urban and Regional Science
- Research on hurricane risk on coastal communities using causal probabilistic analysis through influence diagrams (Bayesian networks), which includes mapping risk for a possible better risk communication
- Hurricane Evacuation Zones



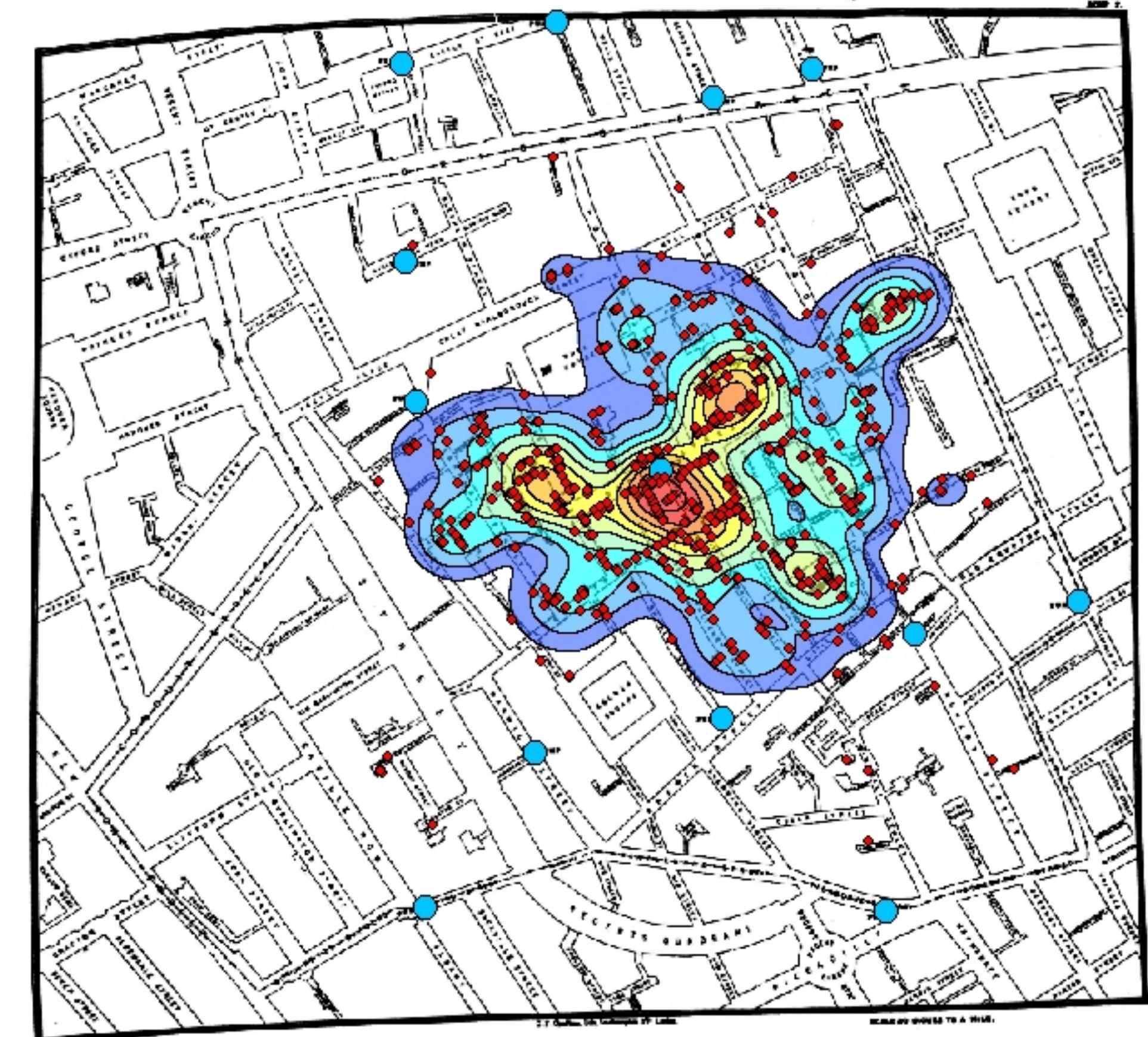
# Some assumptions

- You have [R 4.03](#) and possibly [RStudio v1.4 preview](#) installed
- Basic GIS and spatial language
- Some understanding of how R works, setup a working folder, execute a code, save files  
(check manuals <https://cran.r-project.org>)
- Feel free to [contact me](#) for further informations



# Uses of spatial data in science

- spatial analysis:  
using location or spatial relationships as  
an **explanatory** or **predictive** variable
- examining the effects of **scale**
- backdrops to show **context**

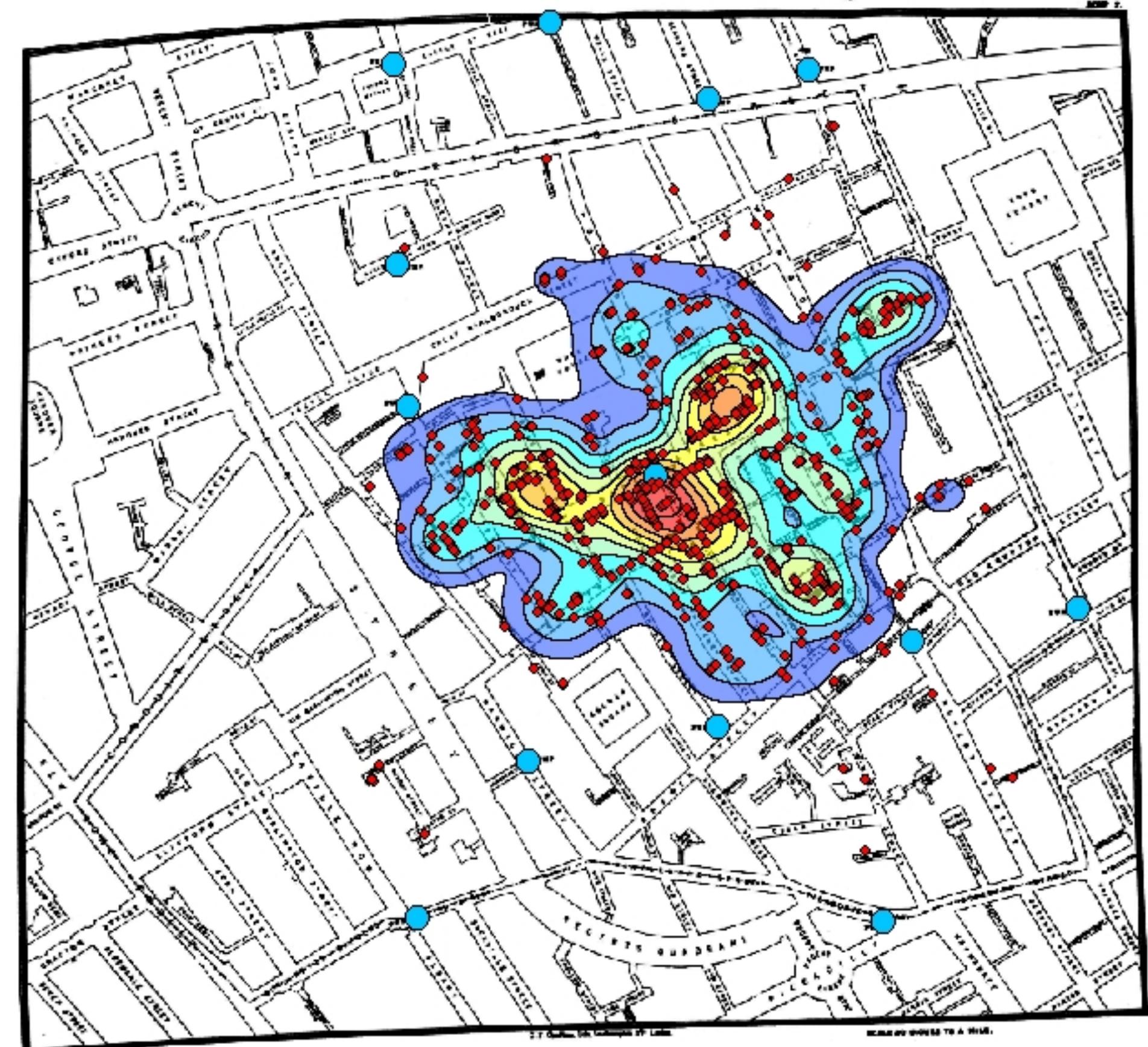


Picture from Andy Lyons  
The Berkeley R Language Beginner Study Group

# Challenges with spatial data in R

- Many types and **formats** of spatial data
- R is not the easiest program to learn
- There are a ton of **packages** from a very diverse user community

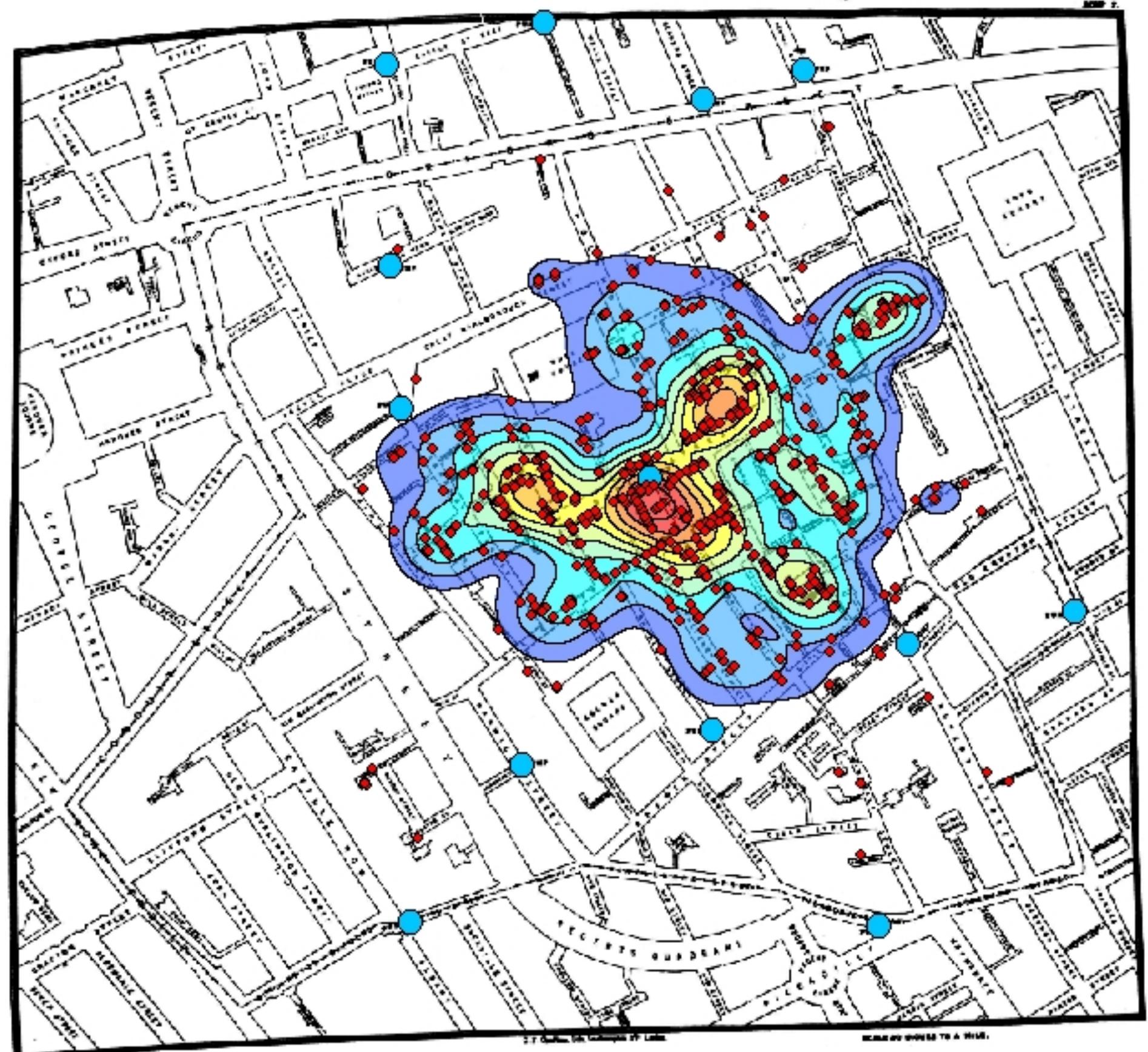
Therefore, understanding and manipulating spatial data in R can be a real challenge!



Picture from Andy Lyons  
The Berkeley R Language Beginner Study Group

# Presentation goals

- Point out some of the commonly-used packages for spatial data manipulation and analysis
- Help you get to the point where you can do an analysis or visualization
- Be a resource you can come back to



Picture from Andy Lyons  
The Berkeley R Language Beginner Study Group

# Getting started

## Why R?



Attribute	Desktop GIS (Graphical User Interface)	Geocomputation with R
Home disciplines	Geography	Computing, Statistics
Software focus	Graphical User Interface	Command line
Reproducibility	Minimal	Maximal

- Community
- open source modeling tools
- R code run as fast as C (but a “higher” level language)
- Why R? 2020 Keynotes

# Getting started

## Why R?

- R is a free software environment for statistical computing and graphics.
- It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.
- Some basic R operators (R as a calculator)

*Arithmetic operators*

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
<sup>^ or </sup> **	Exponentiation

*Logical operators*

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Exactly equal to
!=	Not equal to
!x	Not x
x   y	x OR y
x & y	x AND y

# Brief history of R-spatial

- Spatial packages already available in the **S** language since the 1990s (Bivand and Gebhardt, 2000)
- By 2000, modifications of these became R packages for point pattern analysis, geostatistics, exploratory spatial data analysis and spatial econometrics
- R-GIS bridges (Bivand, 2000)
- Bivand (2003) proposed a spatial data class system for R which eventually led to the packages **rgdal** (first released in 2003; Bivand, Keitt, and Rowlingson, 2018) and **sp** (first released in 2005; Bivand, Pebesma, and Gomez-Rubio, 2013)
- 2008: Applied Spatial Data Analysis with R (Bivand, Pebesma, and Gomez-Rubio, 2013)
- 2010: **raster** package (Hijmans, 2017)
- 2011: **rgeos** package (Bivand and Rundel, 2017)
- 2016-17: **sf** - simple features for R (Pebesma, 2018a)
- 2017-18: **stars** - spatiotemporal tidy arrays for R (Pebesma, 2018b)

Further reading: <https://geocompr.robinlovelace.net/intro.html#the-history-of-r-spatial>

# R-spatial

- **sf** and **sp** are the most important R packages to handle vector data;  
**sf** is a successor of **sp** but its still evolving.  
Moreover, many other R packages depend on the functions and classes  
for the **sp** package
- **raster** is an extension of spatial data classes to work with rasters
- It is also easy to connect R with a GIS software - GRASS GIS (**rgrass7**),  
SAGA GIS (**RSAGA**), QGIS (**RQGIS** and **qgisremote**), and ArcGIS  
(**arcgisbinding**)

Further reading: <https://geocompr.robinlovelace.net/intro.html#the-history-of-r-spatial>

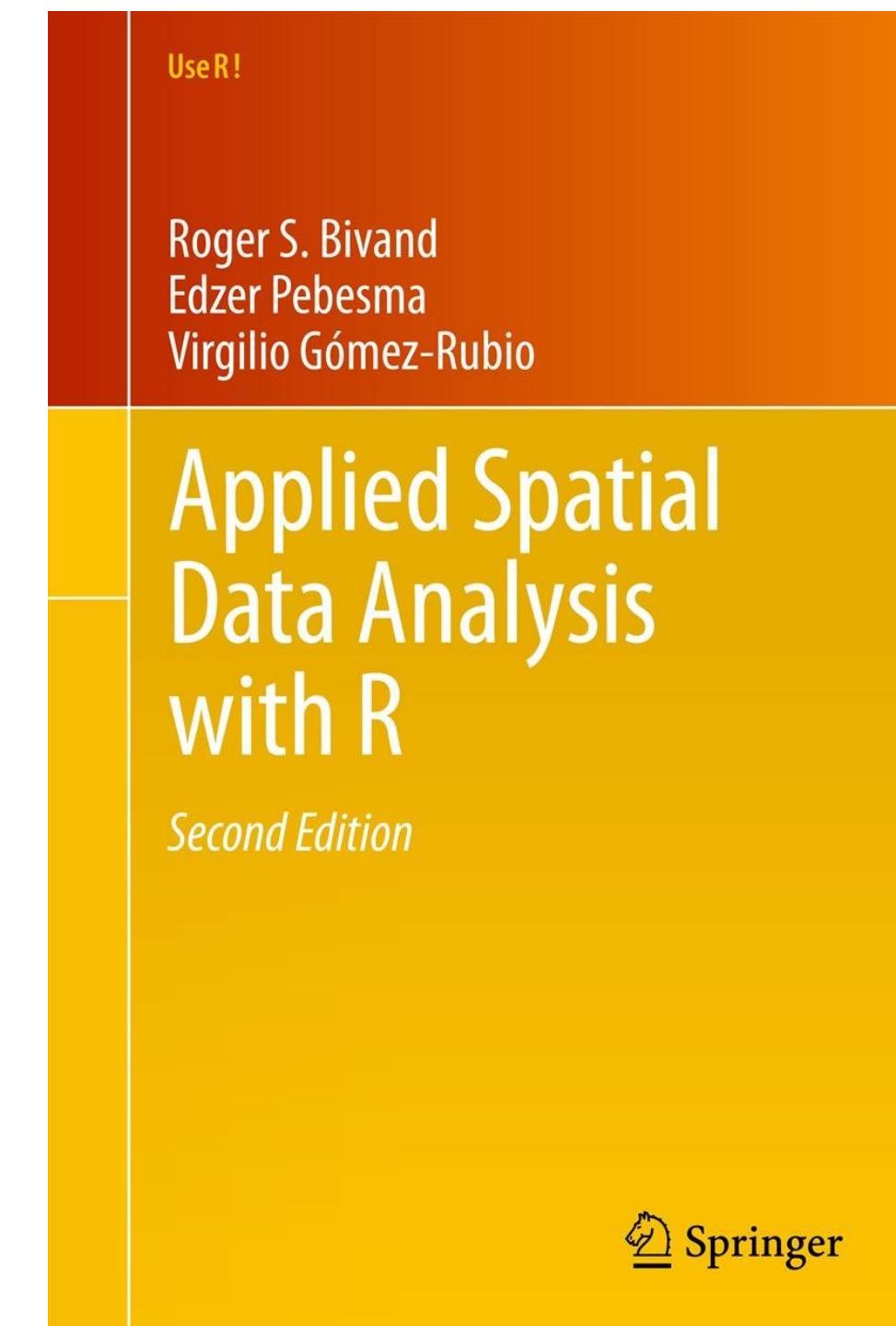
# Resources

## Online

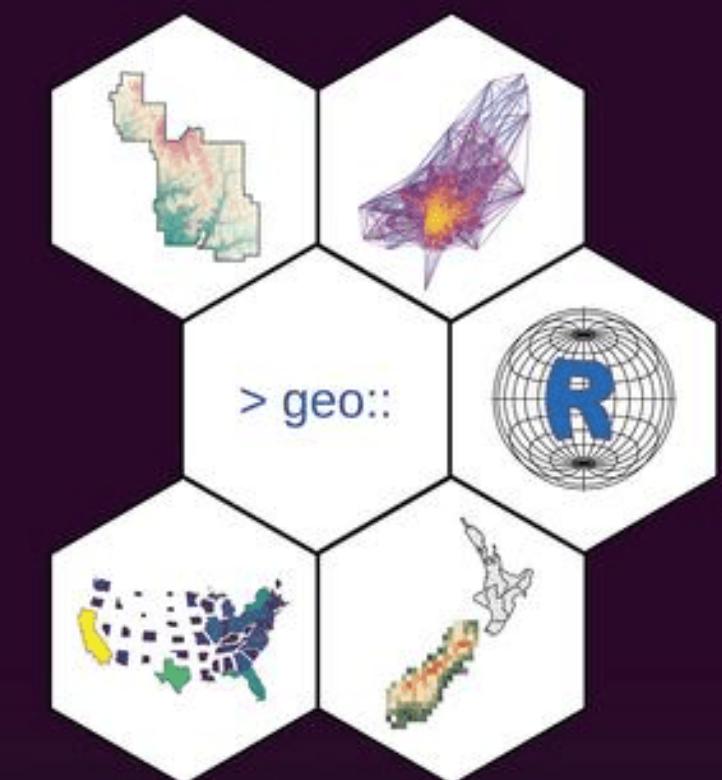
- Geocomputation with R
- Applied Spatial Statistics with R
- Spatial Data Science with R
- Using Spatial Data with R
- Spatial Data Science
- r-spatial

## Paper

- Applied Spatial Data Analysis with R
- An Introduction to R for Spatial Analysis and Mapping



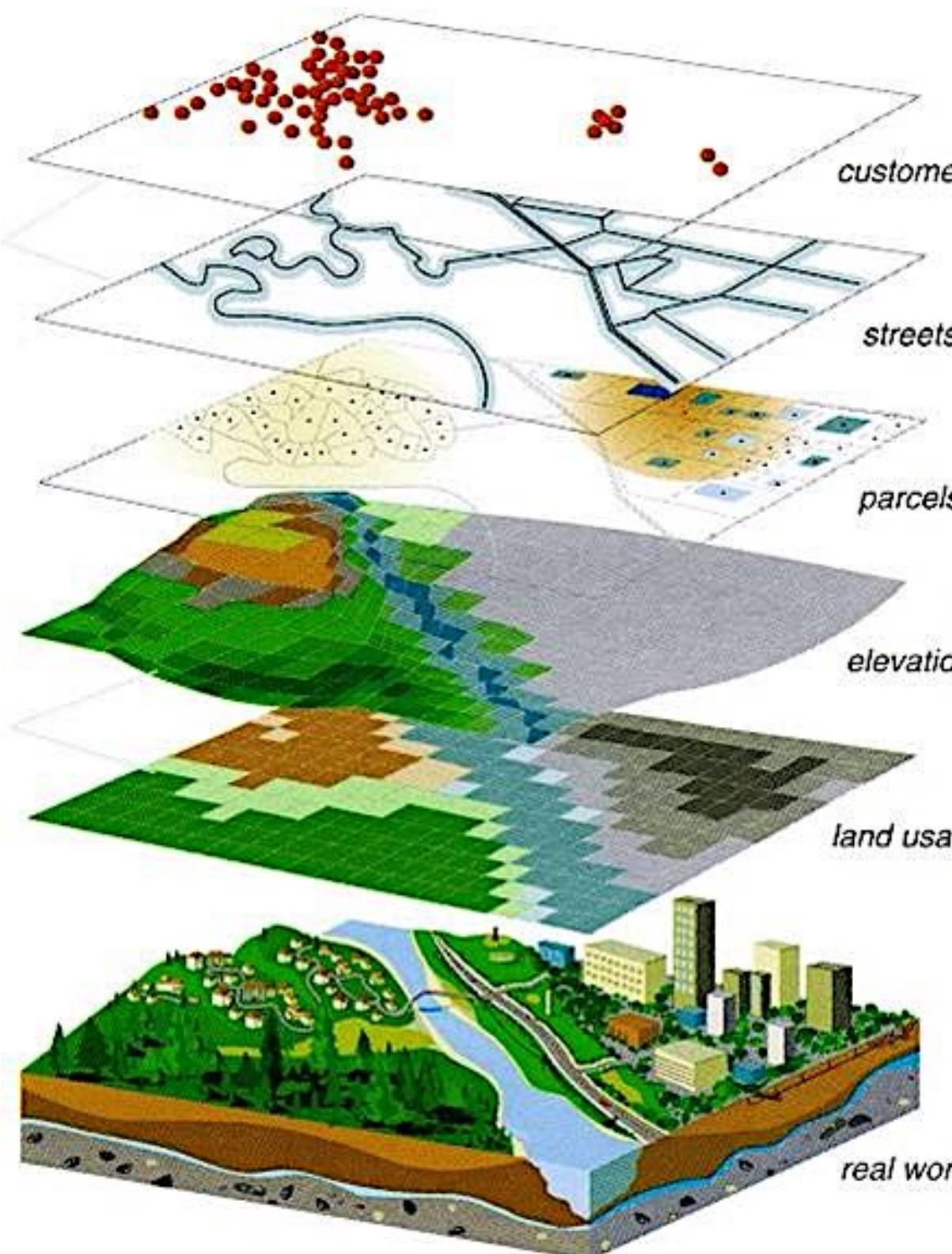
## Geocomputation with R



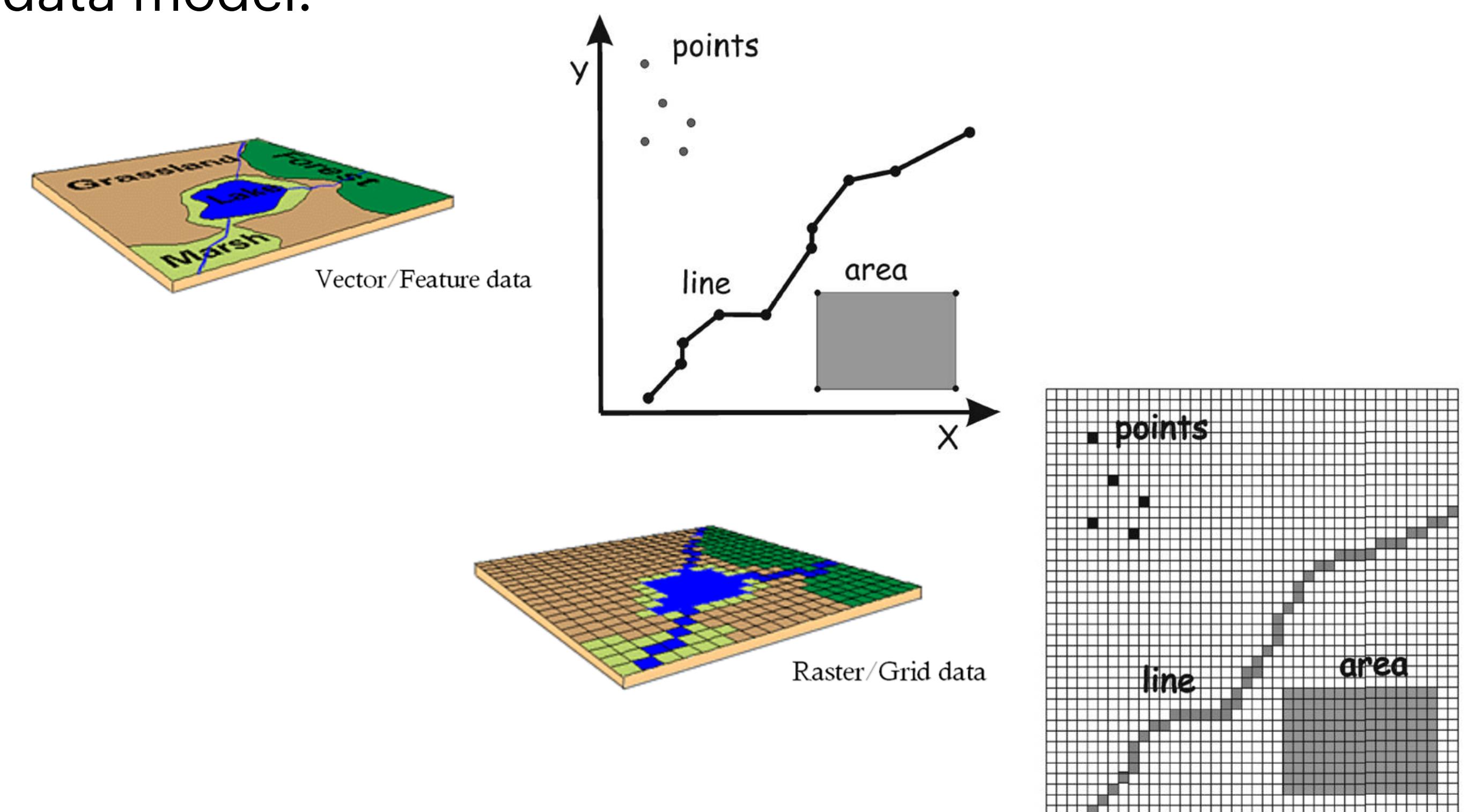
# Getting started

## Representing physical features

Two data models for representing digitally geographic data:  
**the vector<sup>1</sup>** and **the raster<sup>2</sup>** data model.



Images from Andy Lyons  
The Berkeley R Language Beginner Study Group



<sup>1</sup>Hijmans, Robert J. (2019). *Raster: Geographic Data Analysis and Modeling*. R package version 2.8-19.  
URL: <https://CRAN.R-project.org/package=raster>.

<sup>2</sup>Pebesma, Edzer (2018). "Simple Features for R: Standardized Support for Spatial Vector Data". In: *The R Journal* 10.1, pp. 439-446. URL: <https://journal.r-project.org/archive/2018/RJ-2018-009/index.html>.

# Getting started

## Creating spatial objects

Three options:

- from scratch (e.g. digitizing<sup>1</sup>, geocoding<sup>2</sup>)
- promote a data frame including a geometry column (e.g. Census)
- import a GIS file (e.g. shapefiles, rasters, geopackages)

<sup>1</sup>**Digitizing** is the process by which coordinates from a map, image, or other sources of data are converted into a digital format in a GIS.

<sup>2</sup>**Geocoding** is the process of taking a text-based description of a location, such as an address or the name of a place, and returning geographic coordinates, frequently latitude/longitude pair, to identify a location on the Earth's surface.

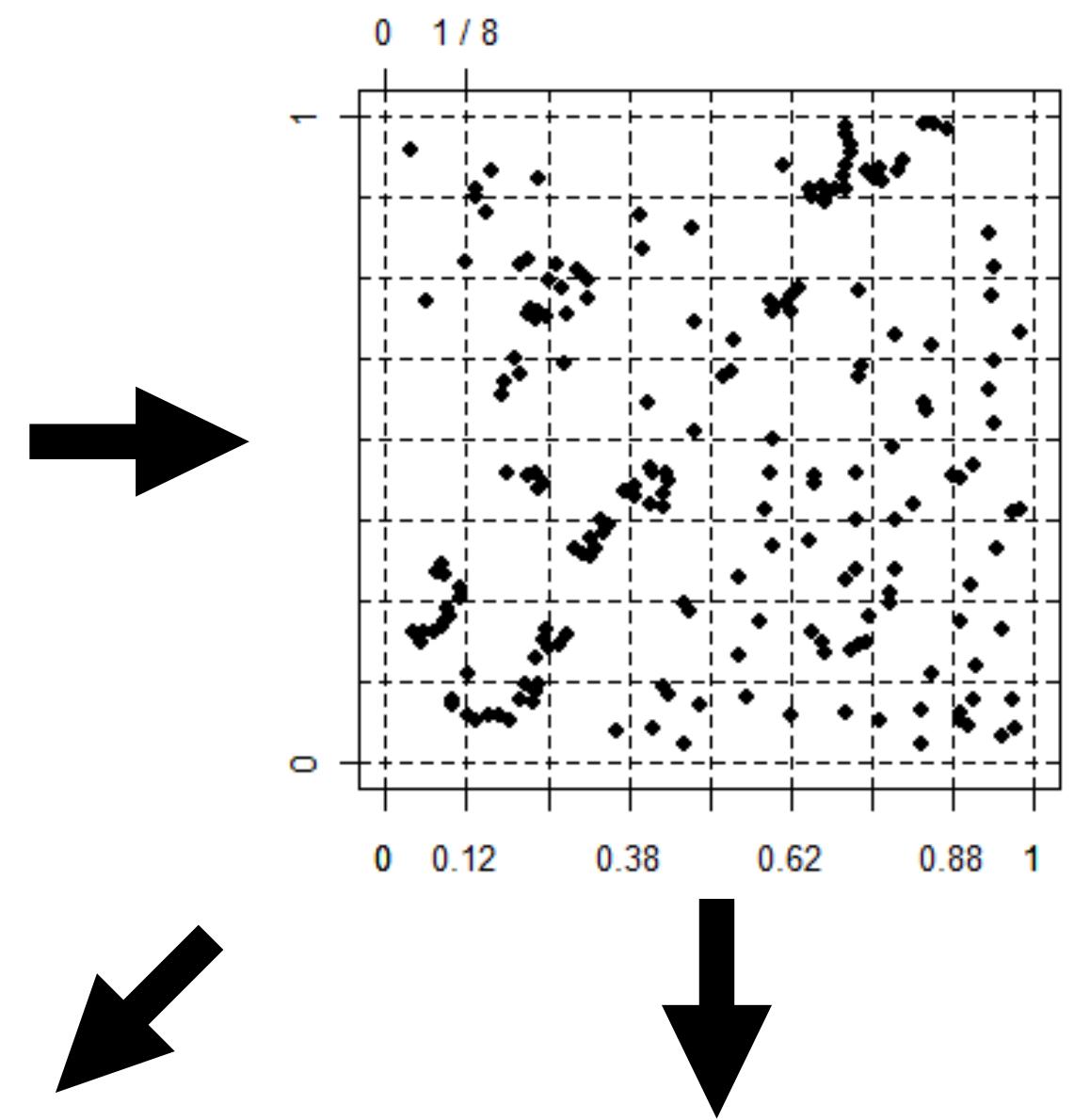
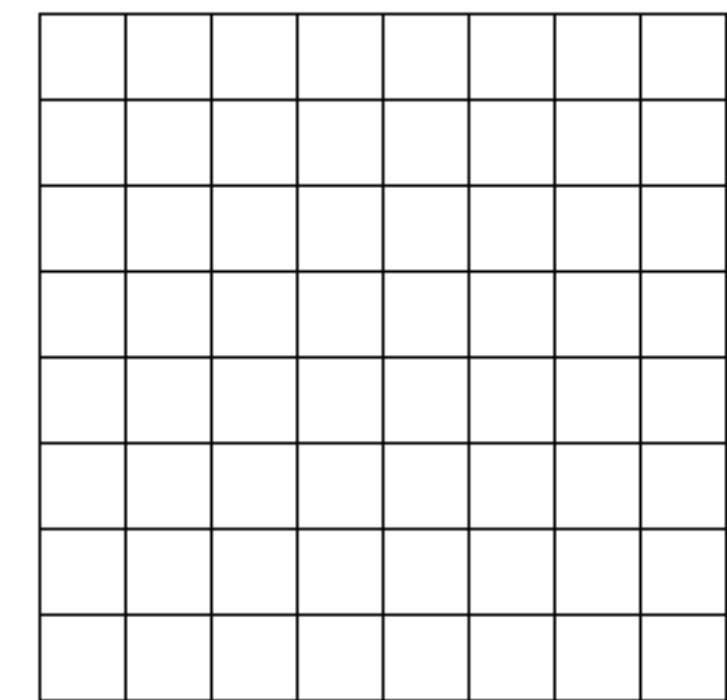
Find more at <http://wiki.gis.com/wiki/index.php/Digitizing>

# Spatial raster data

# Spatial raster data

## Raster package

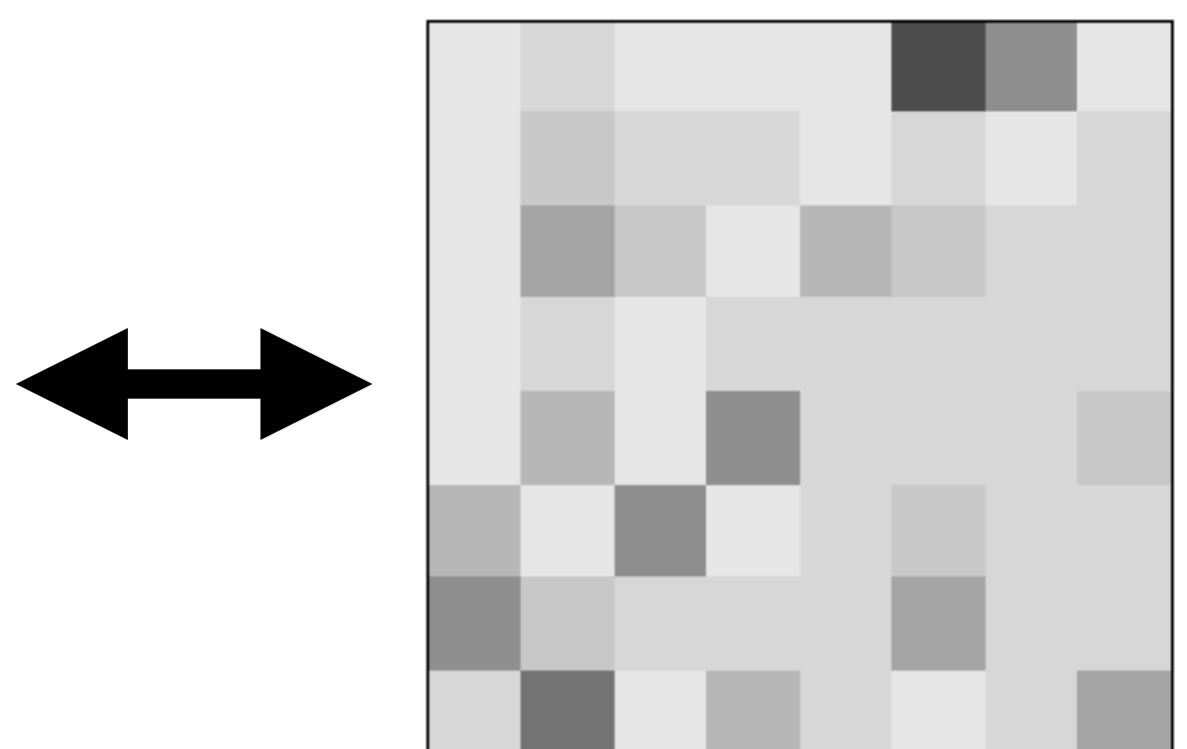
- **read, write** and **manipulate** gridded spatial data
- basic and high-level analysis **functions**
- **processing** of large files



1	3	0	0	1	12	8	0
1	4	3	3	0	2	0	2
1	7	4	1	5	4	2	2
0	3	1	2	2	2	2	3
0	5	1	9	3	3	3	4
5	0	8	0	2	4	3	2
8	4	3	2	2	7	2	3
2	10	1	5	2	1	3	7

Using a raster to summarize a point pattern

Image adapted from [https://en.wikipedia.org/wiki/Raster\\_graphics#/media/File:The\\_use\\_of\\_a\\_raster\\_data\\_structure\\_to\\_summarize\\_a\\_point\\_pattern.gif](https://en.wikipedia.org/wiki/Raster_graphics#/media/File:The_use_of_a_raster_data_structure_to_summarize_a_point_pattern.gif)



# Spatial raster data

## Lots of real world applications

- Raster or "gridded" data are stored as a grid of values which are rendered on a map as pixels.
- **Each pixel value represents an area on the Earth's surface.**

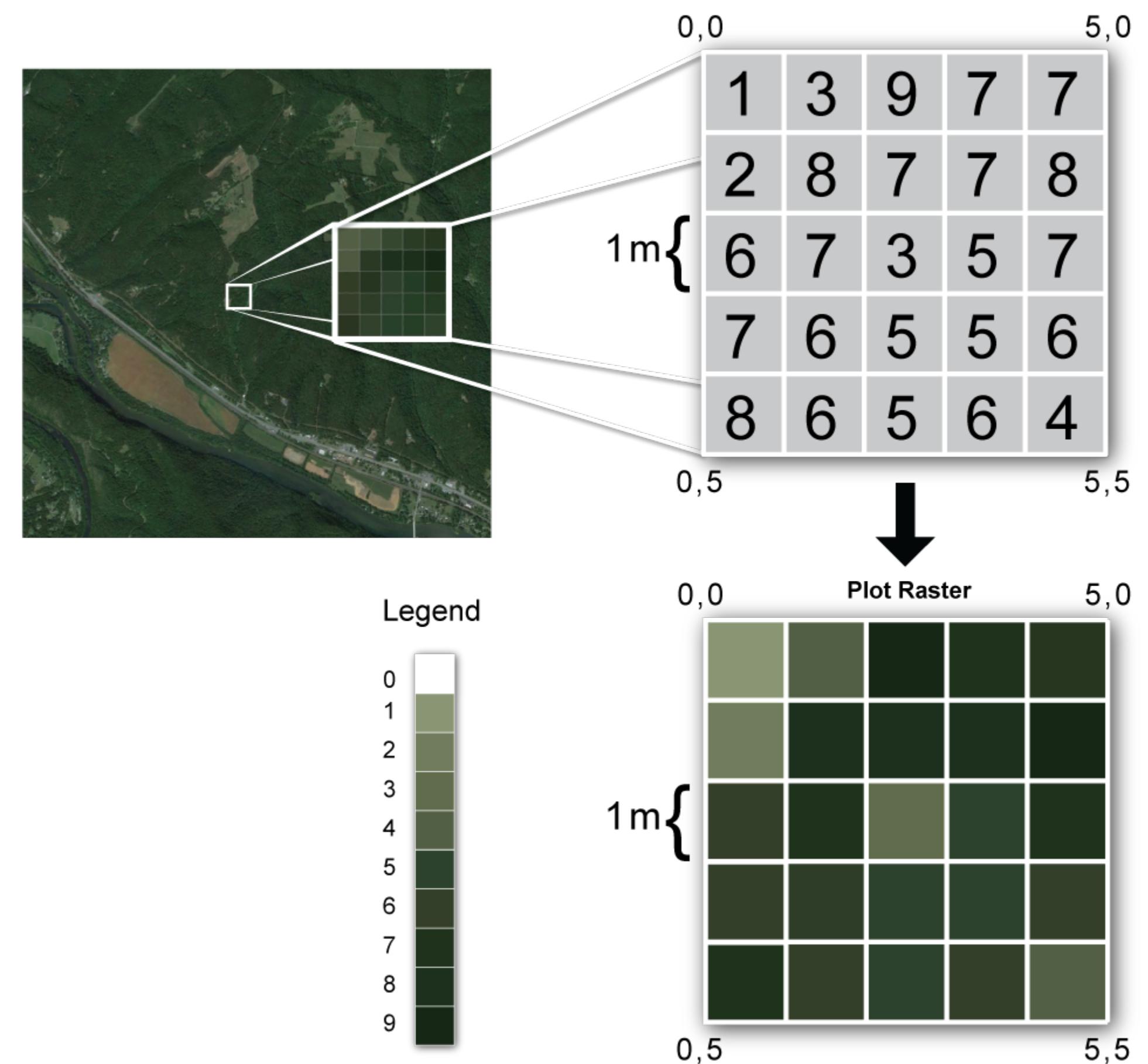


Image: National Ecological Observatory Network  
<https://www.neonscience.org/resources/learning-hub/tutorials/dc-raster-data-r>

# Spatial raster data

## Resolution

- Each pixel value represents an area on the Earth's surface
- Resolution represents the area on the ground that each pixel covers
- The best way to view resolution units is to look at the coordinate reference system string

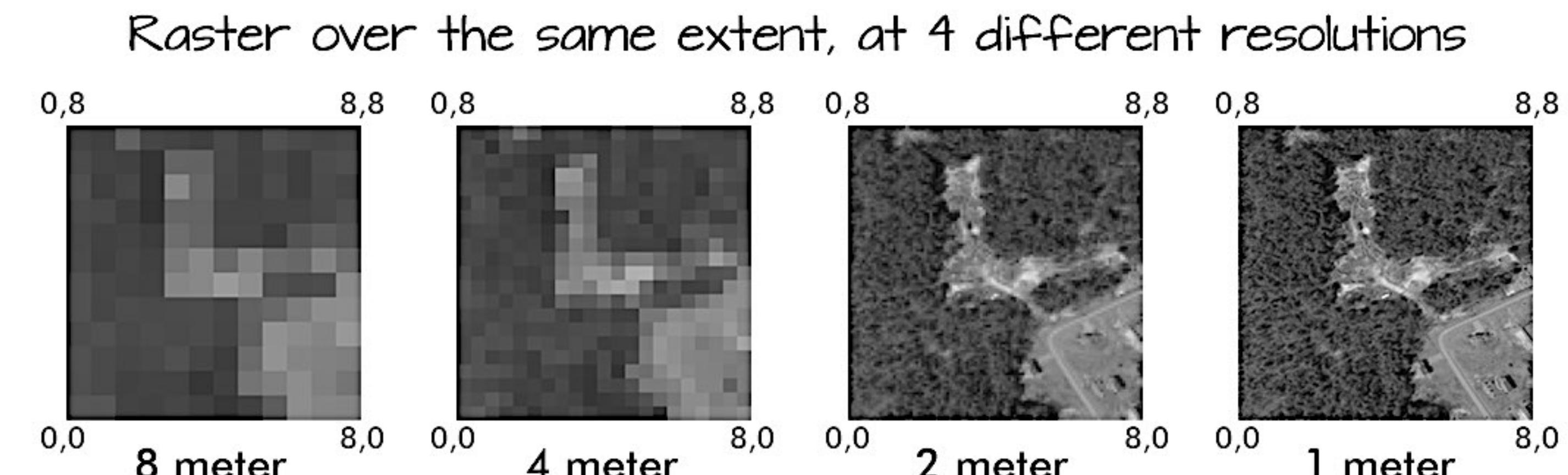


Image: National Ecological Observatory Network  
<https://www.neonscience.org/resources/learning-hub/tutorials/dc-raster-data-r>

# Spatial raster data

## Creating Raster objects from scratch

```
library(raster)

r <- raster(ncol=10, nrow=10)

ncell(r)

hasValues(r)

values(r) <- 1:ncell(r)

set.seed(0)

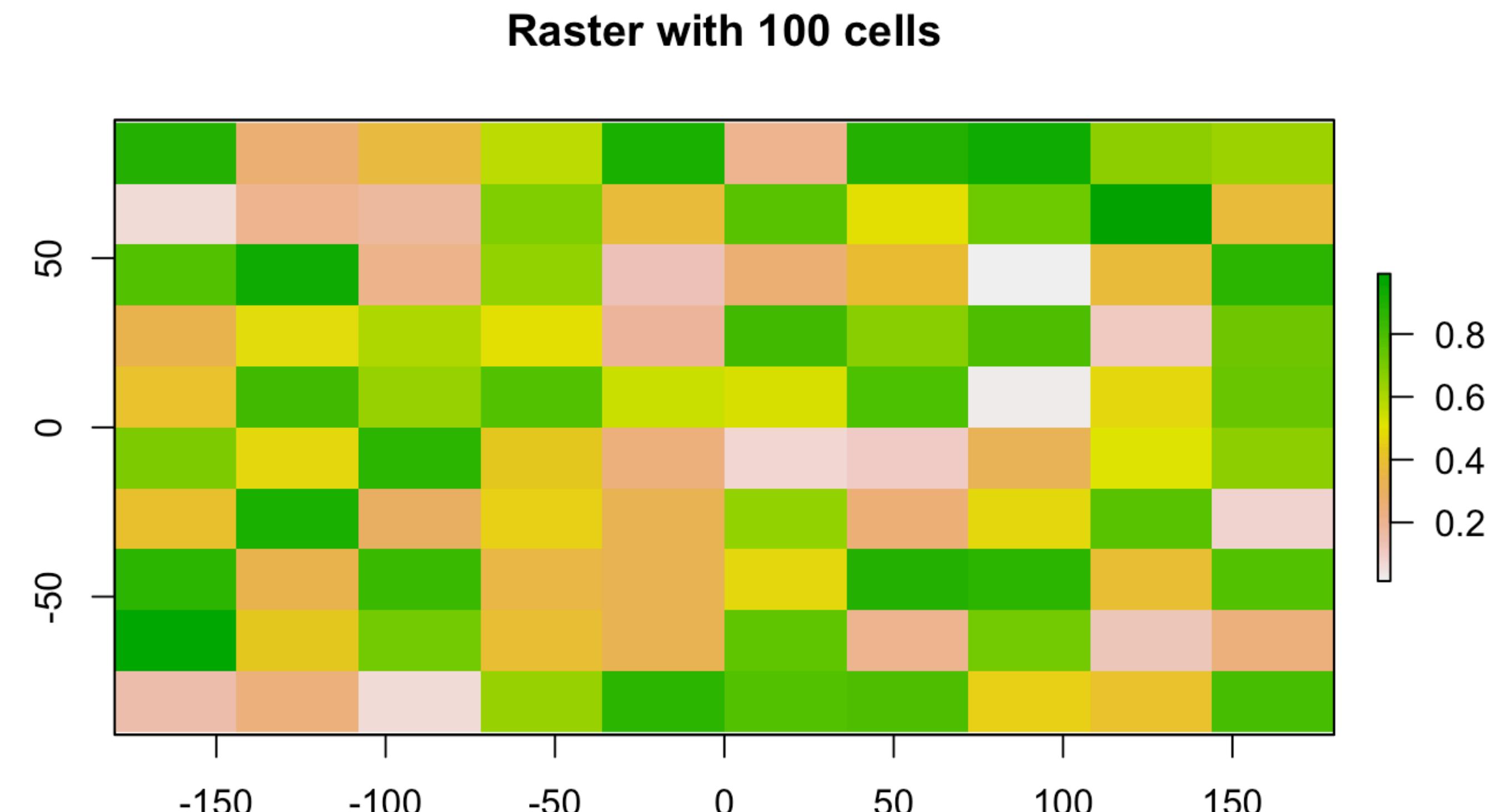
values(r) <- runif(ncell(r))

hasValues(r)

inMemory(r)

values(r)[1:10]

plot(r, main='Raster with 100 cells')
```



# Spatial raster data

## Raster algebra

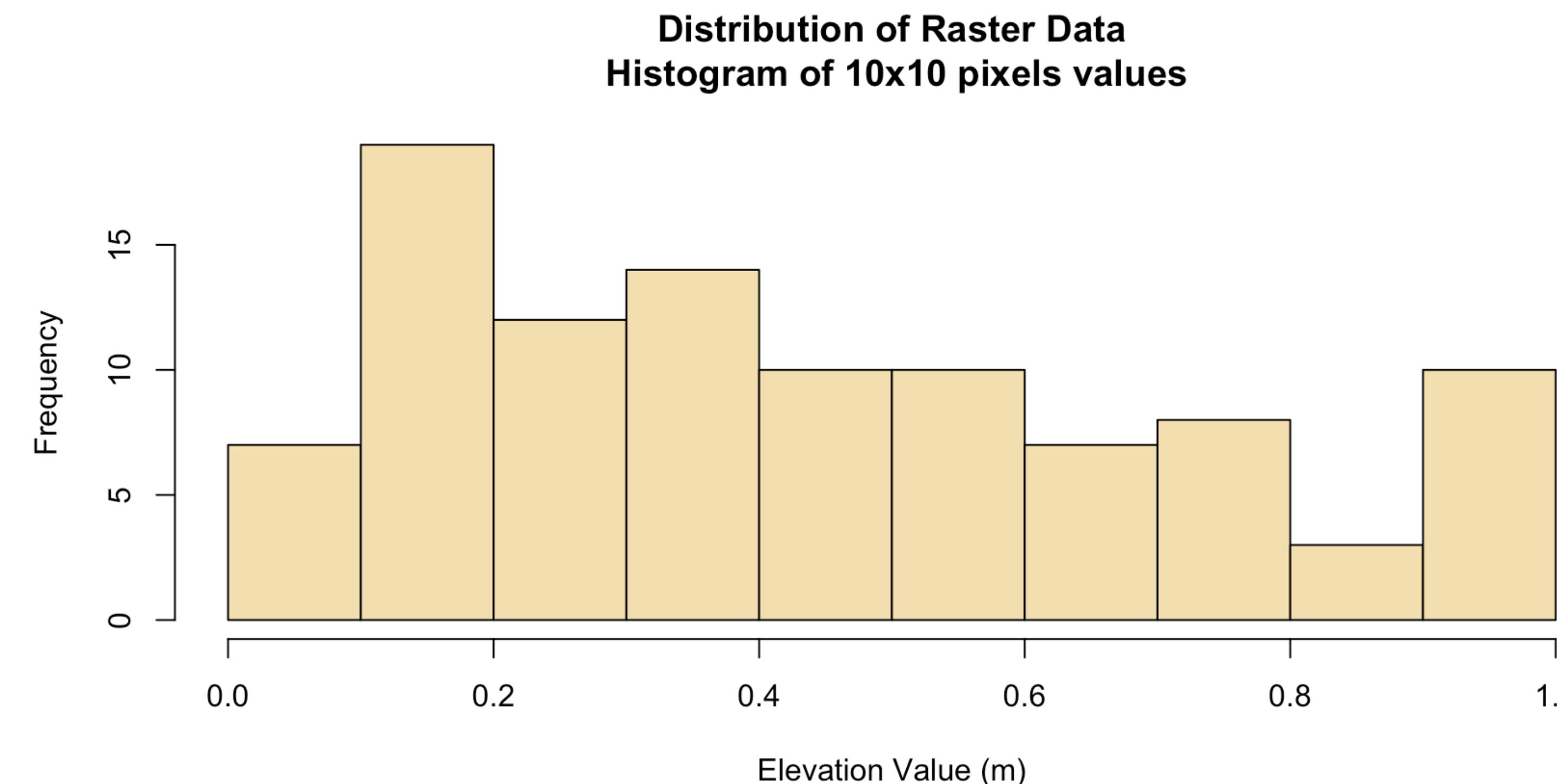
```
r  
values(r)[1:10]  
  
s <- r + 10  
values(s)[1:10]  
  
s <- sqrt(s)  
values(s)[1:10]  
  
s <- s * r + 5  
values(s)[1:10]  
  
r[] <- runif(ncell(r)) # replacements  
values(r)[1:10]  
  
r <- round(r)  
values(r)[1:10]  
  
r <- r == 1  
values(r)[1:10]
```

- Many generic functions that allow for simple and elegant raster algebra have been implemented for Raster objects

# Spatial raster data

## Raster histogram

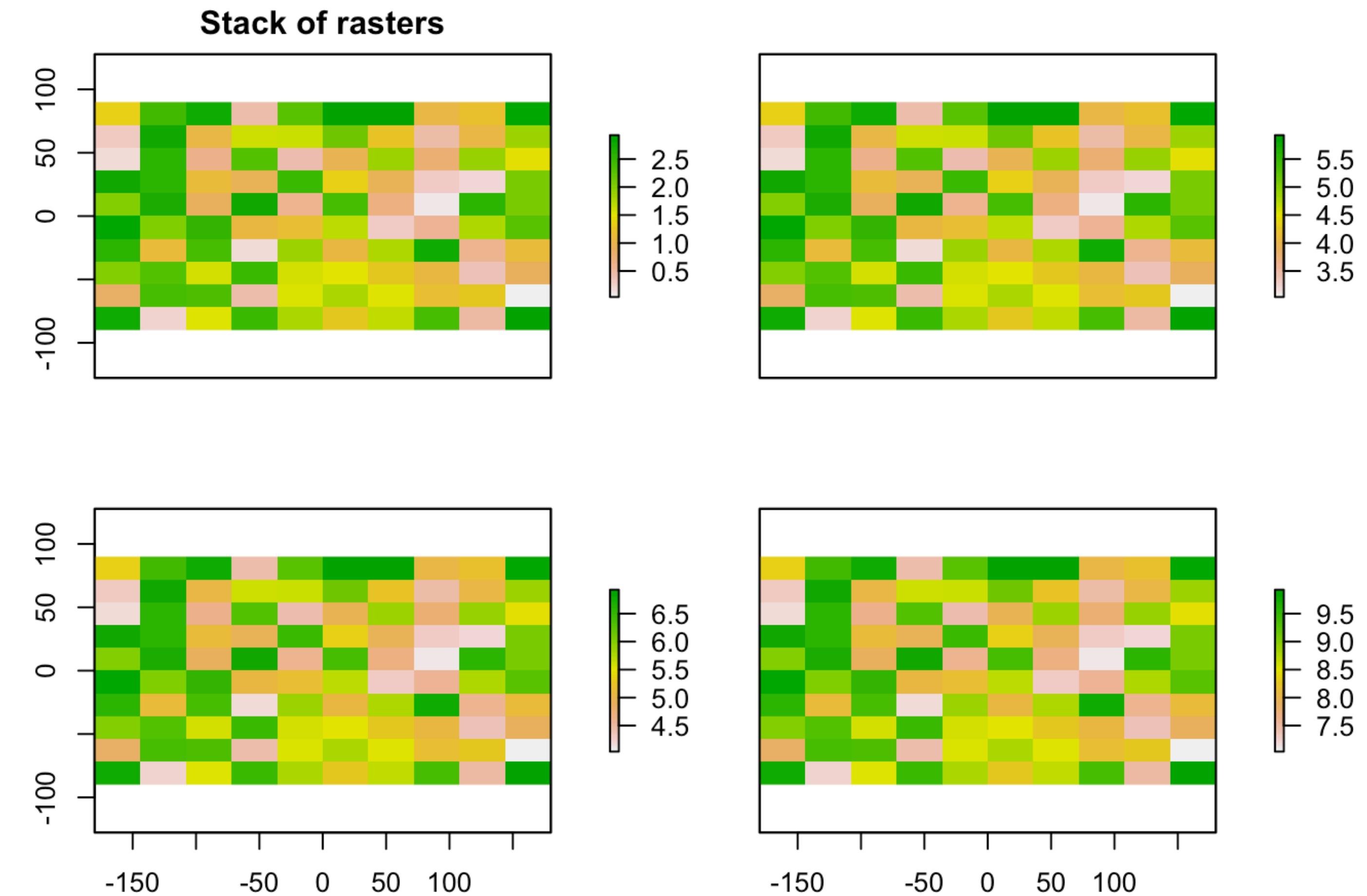
```
hist(s,  
  main="Distribution of Raster Data\n Histogram of 10x10 pixels values",  
  xlab="Elevation Value (m)",  
  ylab="Frequency",  
  col="wheat")
```



# Spatial raster data

## Multiple layers

```
r <- raster(ncol=10, nrow=10)  
  
r[] <- runif(ncell(r))  
  
s <- stack(r, r+1)  
  
q <- stack(r, r+2, r+4, r+6)  
  
x <- r + s + q  
  
x  
  
plot(x, main='Stack of rasters')
```



# Spatial raster data

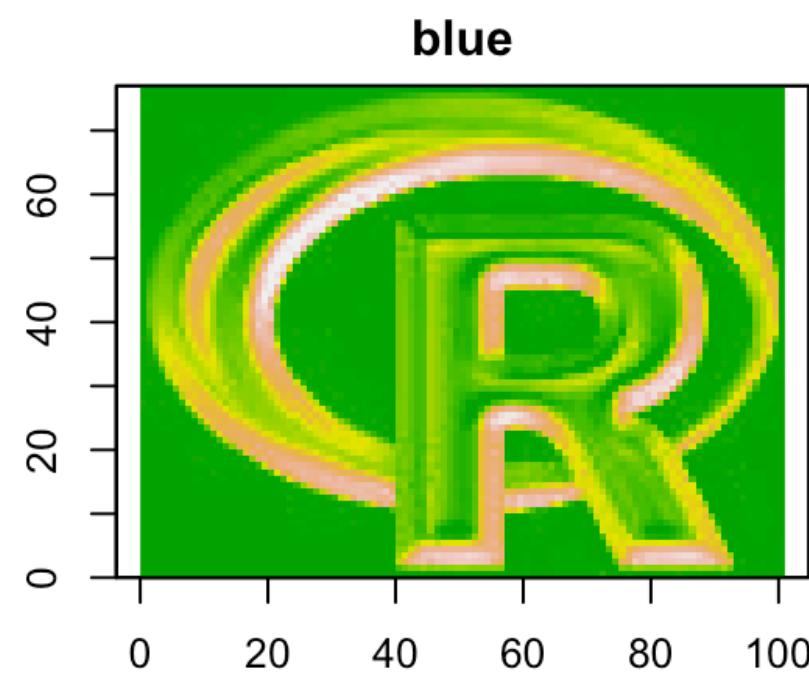
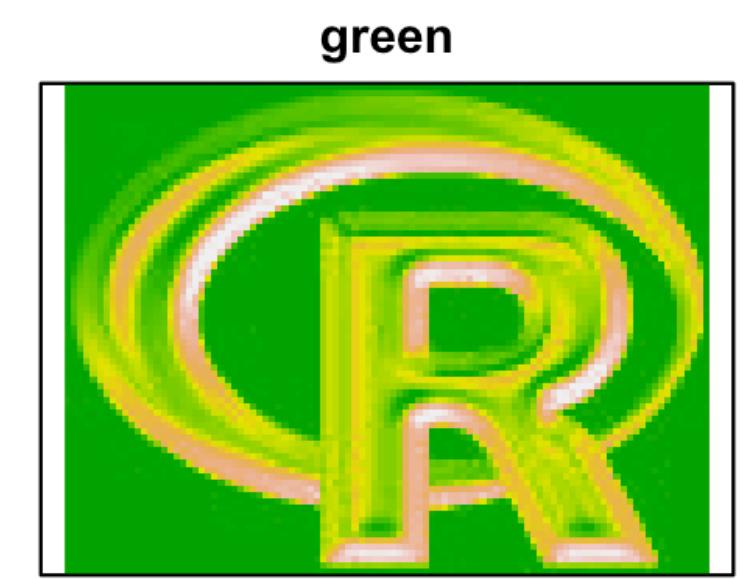
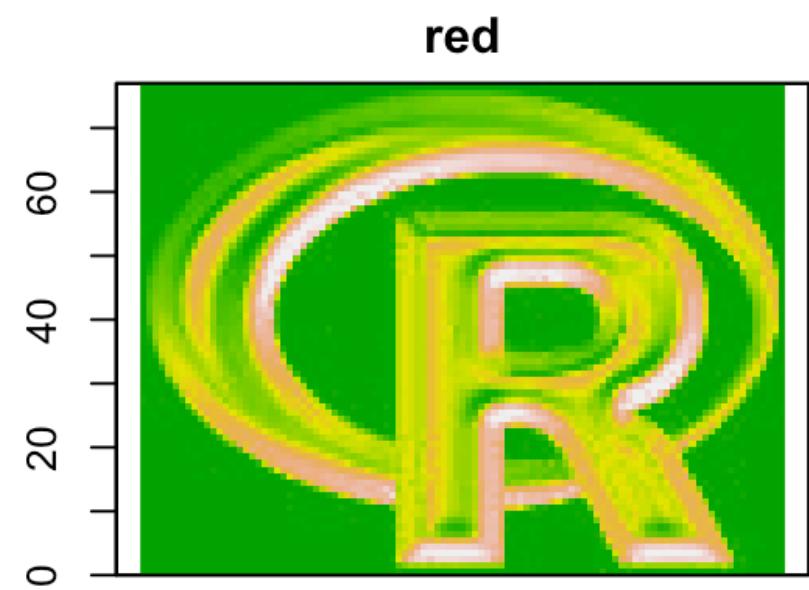
## Raster data types

- **RasterLayer** - a single layer raster
- **RasterStack** - multi-layer raster
- **RasterBrick** - multi-layer raster (from a single file on disk, faster)

# Spatial raster data

## Multiple layers

```
b <- brick(system.file("external/rlogo.grd", package="raster"))  
plot(b)  
plotRGB(b, r=1, g=2, b=3)
```



# Spatial raster data

## Summary functions

(min, max, mean, prod, sum, median, cv, range)

```
a <- mean(r,s)
values(a)[1:10]

b <- sum(r,s)

st <- stack(r, s, a, b)
sst <- sum(st)
sst

cellStats(st, "sum")
cellStats(sst, "sum")
```

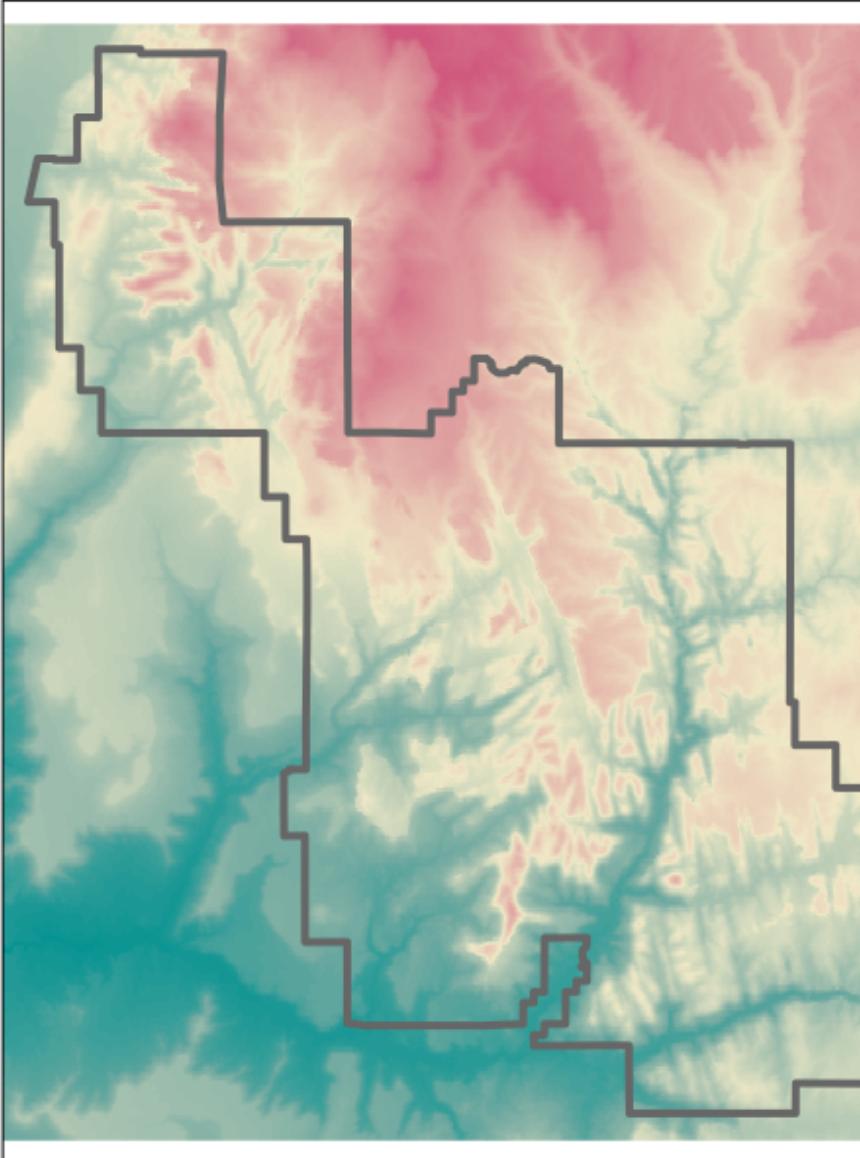
- Summary functions always return a RasterLayer object
- Perhaps this is not obvious when using functions like min, sum or mean

# Spatial raster data ... to be continued

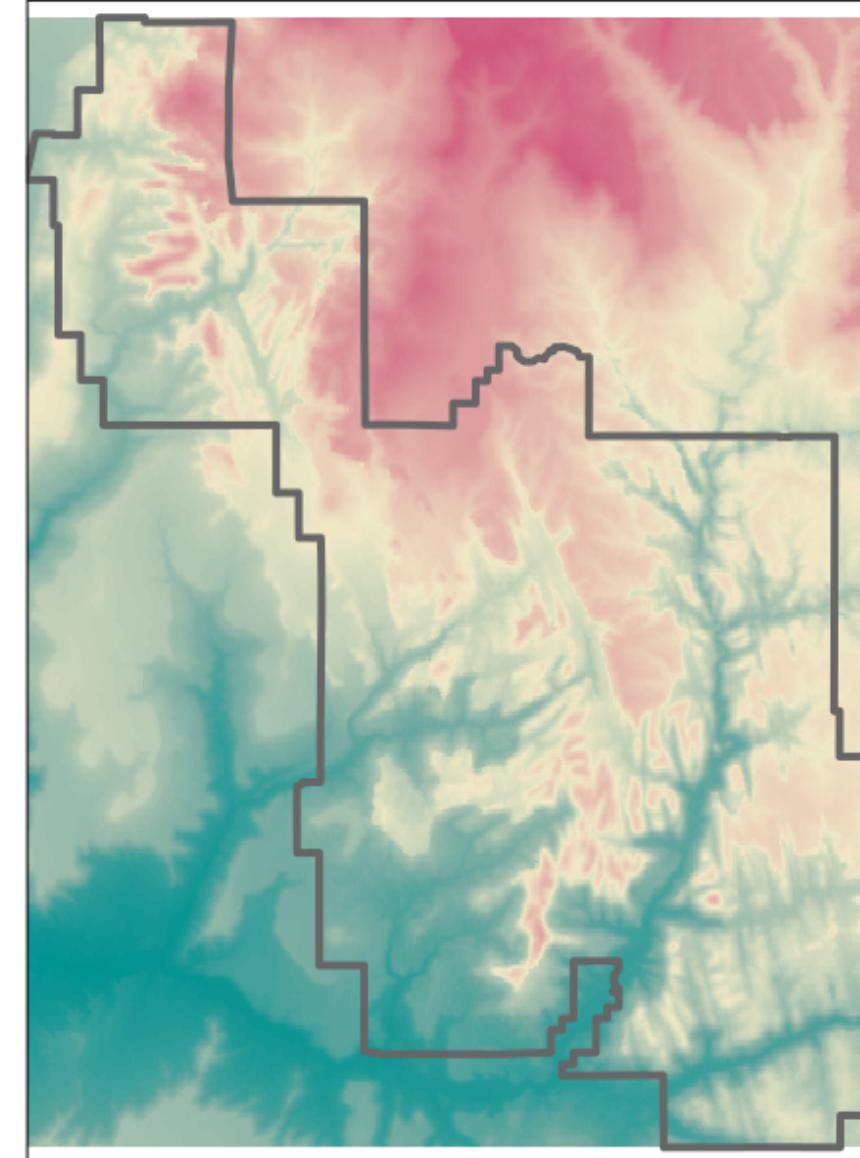
Many others geometric operations on raster data ...

Check: <https://geocompr.robinlovelace.net/geometric-operations.html#raster-vector>

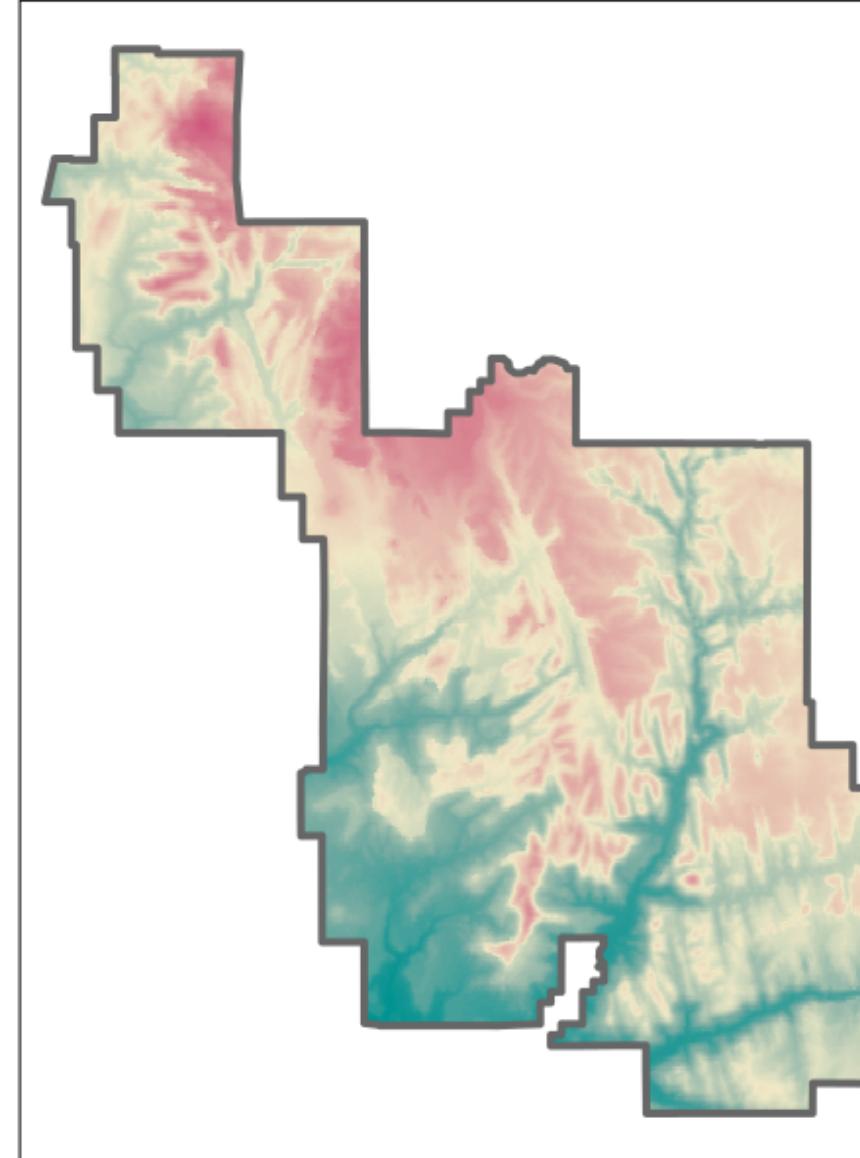
A. Original



B. Crop



C. Mask



D. Inverse mask

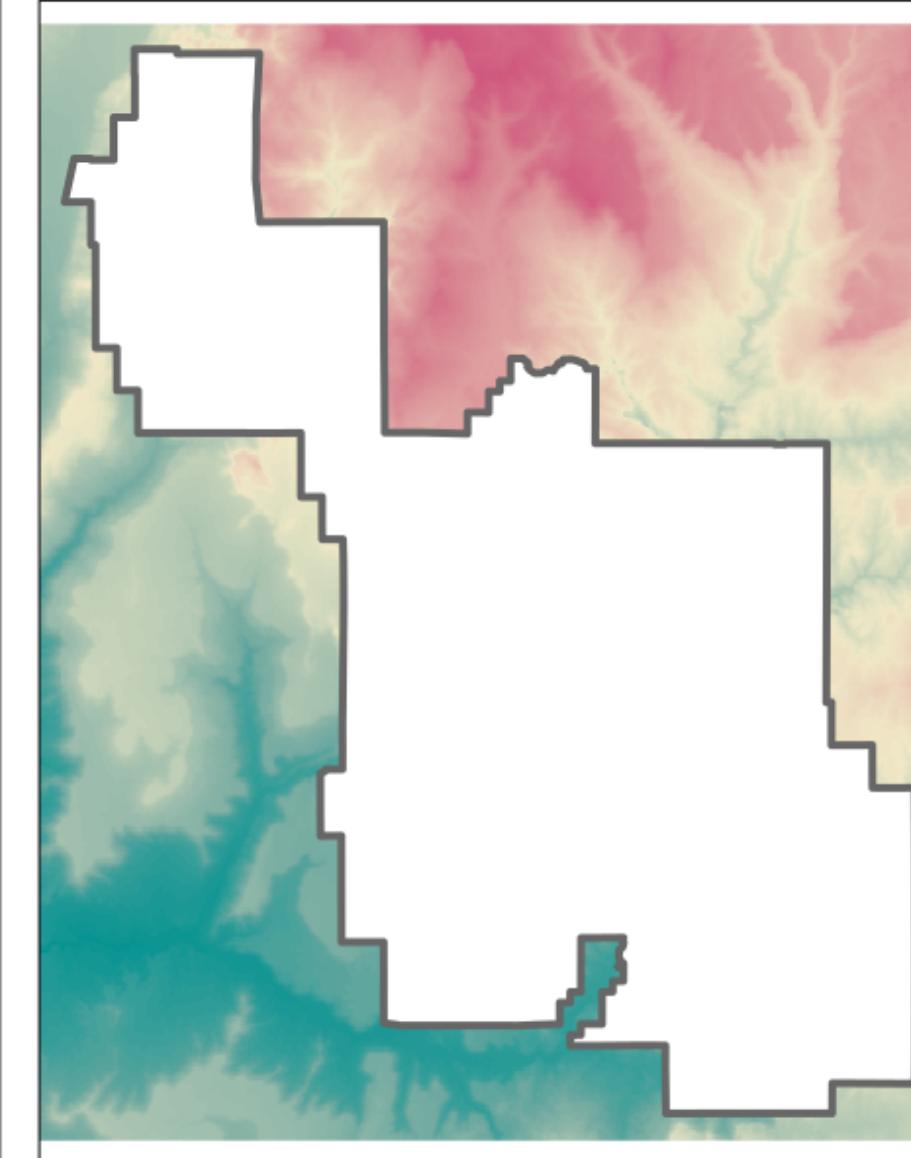


Illustration of raster cropping and raster masking

Image: Robin Lovelace, Jakub Nowosad, Jannes Muenchow

# **Spatial vector data**

# Spatial vector data

## The sf package

```
devtools::install_github("r-spatial/sf") # development version  
# or  
install.packages("sf") # stable version
```

The sf package is an R implementation of Simple Features. This package incorporates:

- A relatively new spatial data class system in R
- Functions for reading and writing data
- Tools for spatial operations on vectors

Most of the functions in this package start with a prefix `st_`.

You need a recent version of the **GDAL**, **GEOS**, **Proj.4**, and **UDUNITS** libraries installed for this to work on Mac and Linux. More information on that at <https://github.com/r-spatial/sf>

# Spatial vector data

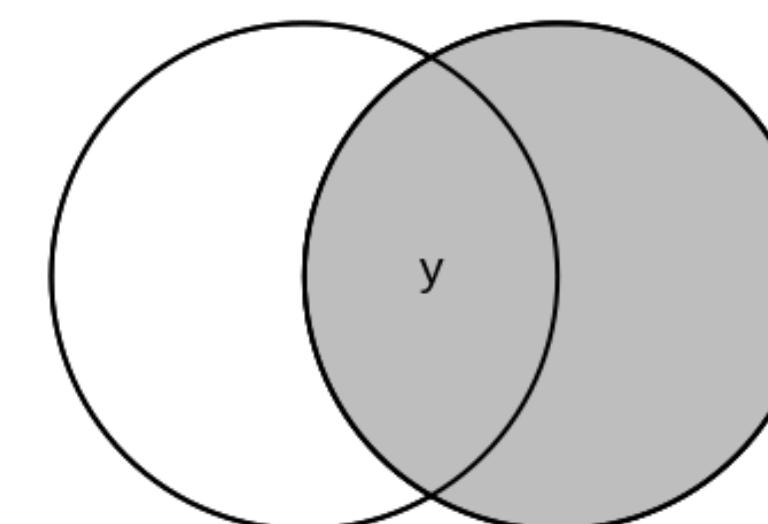
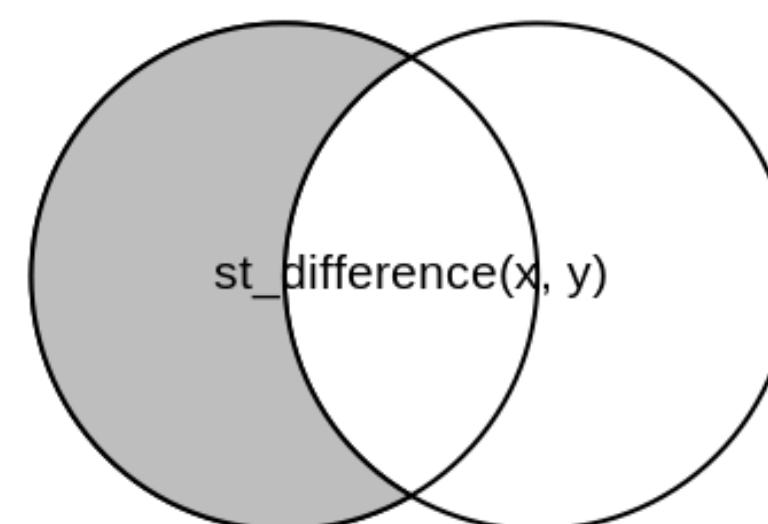
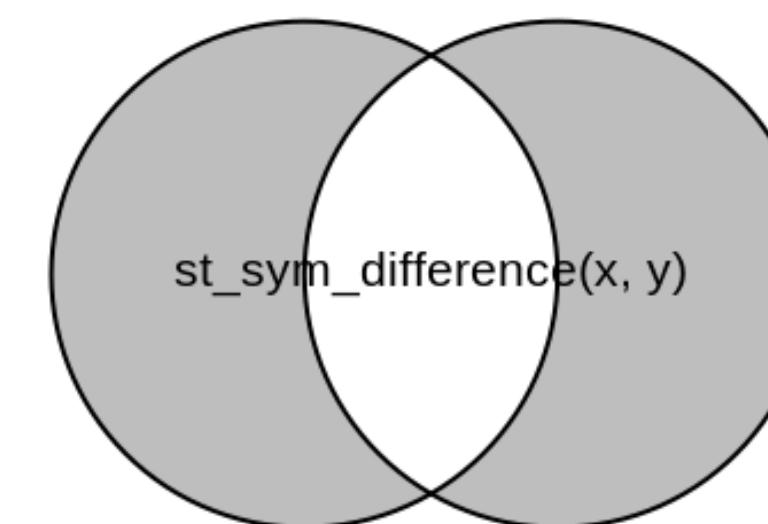
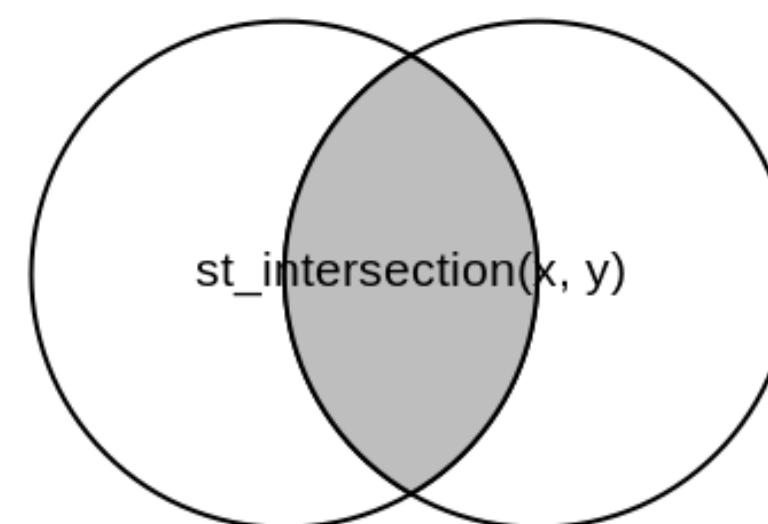
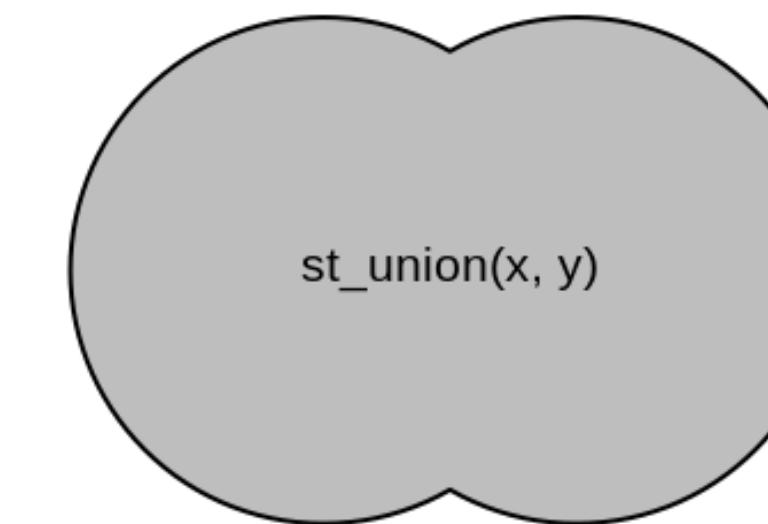
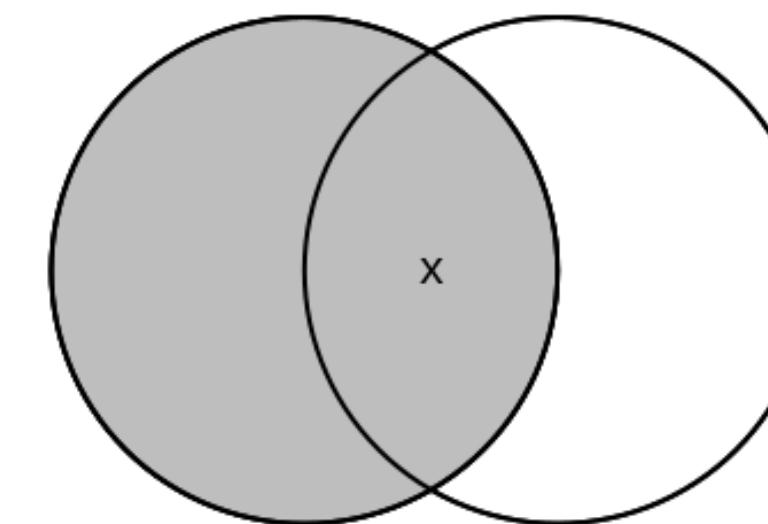
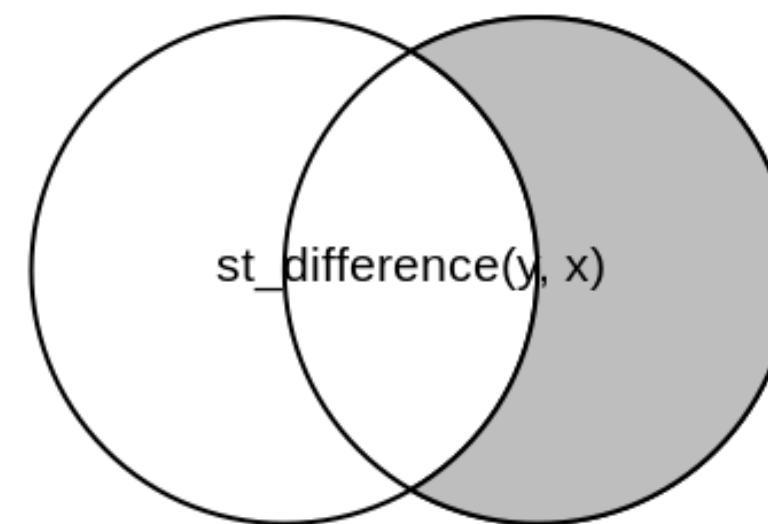
## Simple feature geometry types

- Simple feature geometries are a way to describe the geometries of features
- By features it mean things that have a geometry, and eventually other attributes
- The big seven (most commonly used simple features geometries):

type	description
POINT	single point geometry
MULTIPOINT	set of points
LINESTRING	single linestring (two or more points connected by straight lines)
MULTILINESTRING	set of linestrings
POLYGON	exterior ring with zero or more inner rings, denoting holes
MULTIPOLYGON	set of polygons
GEOMETRYCOLLECTION	set of the geometries above

# Spatial vector data

## Spatial equivalents of logical operators

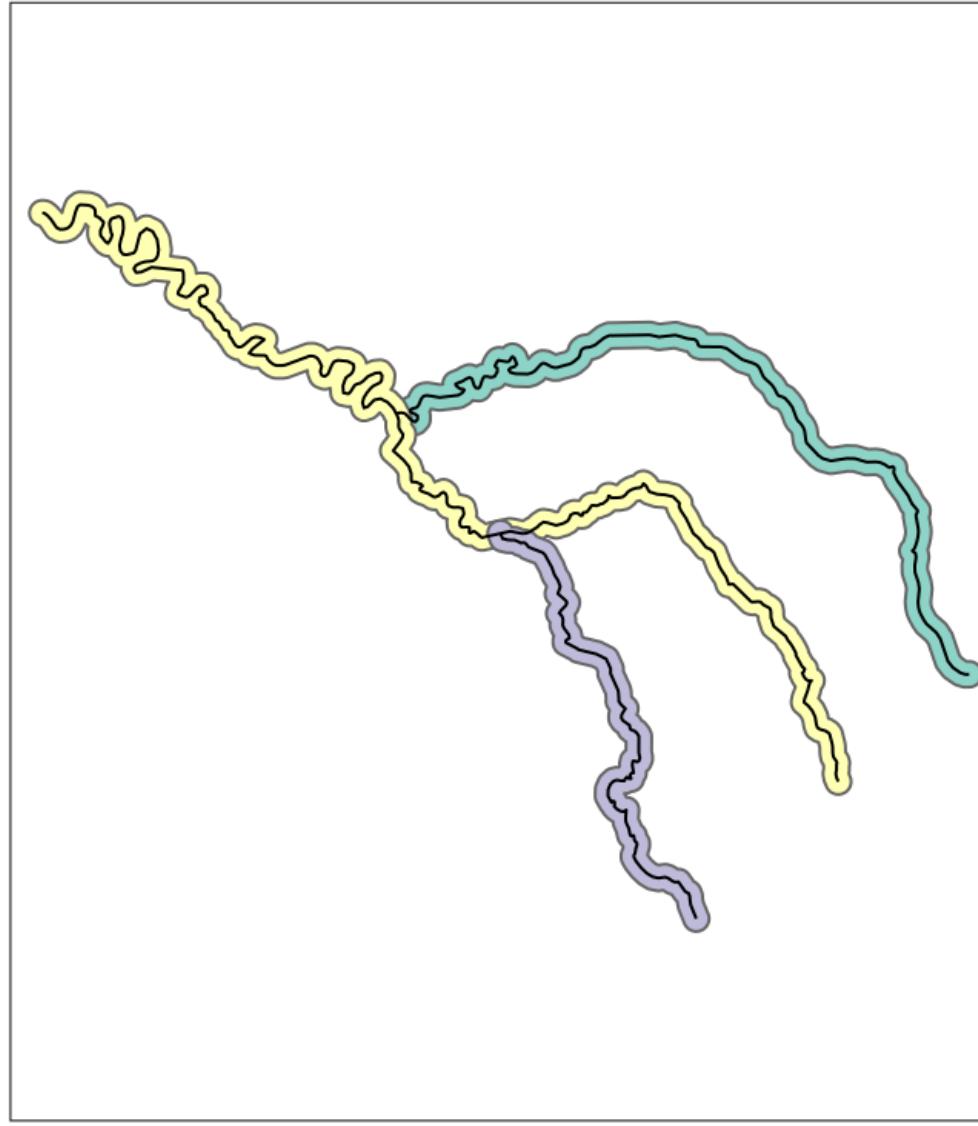


# Spatial vector data

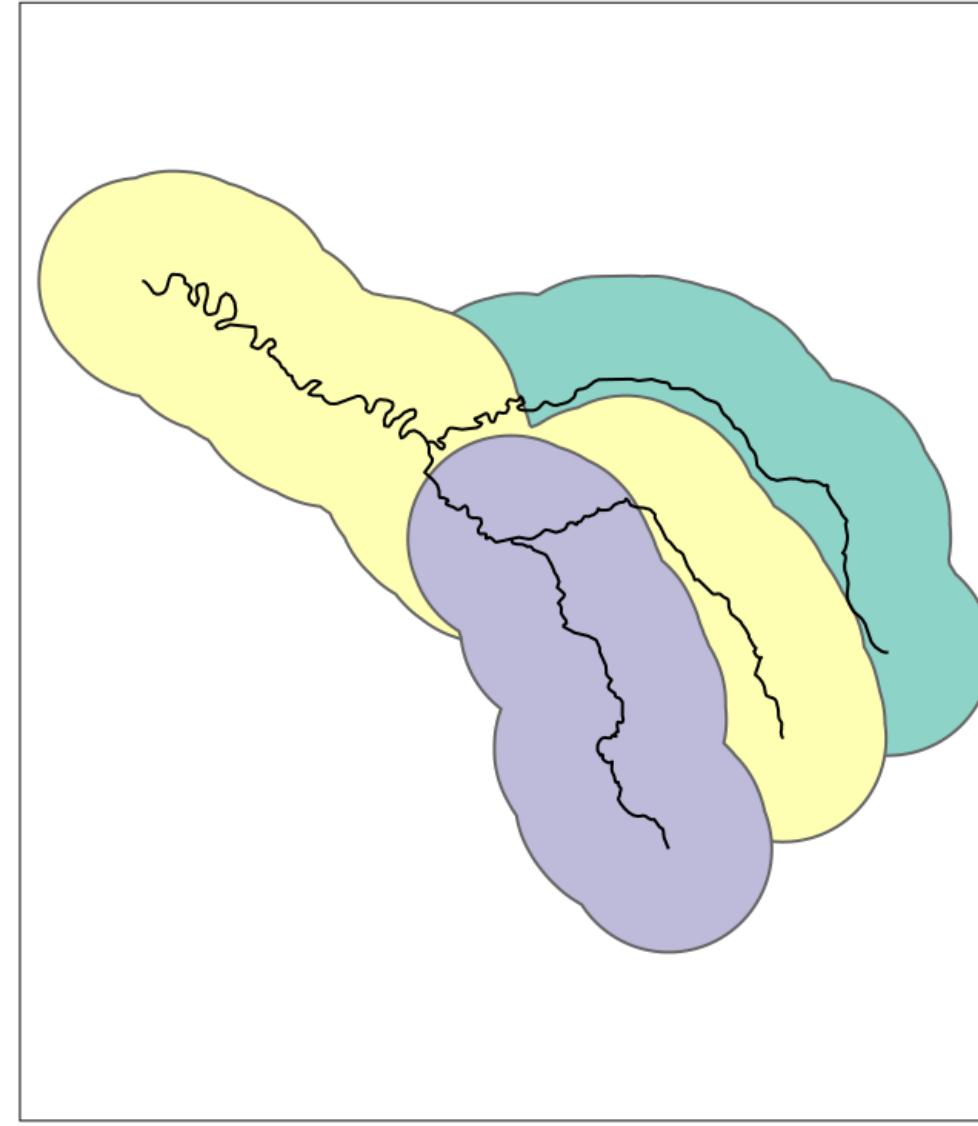
## Many other spatial operations

Check: <https://geocompr.robinlovelace.net/geometric-operations.html>

5 km buffer



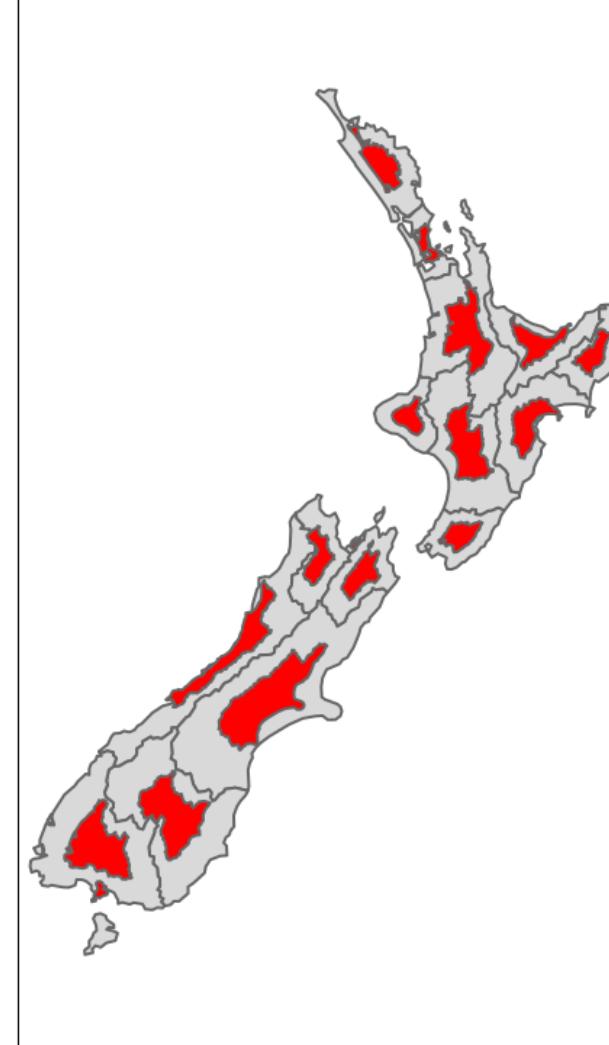
50 km buffer



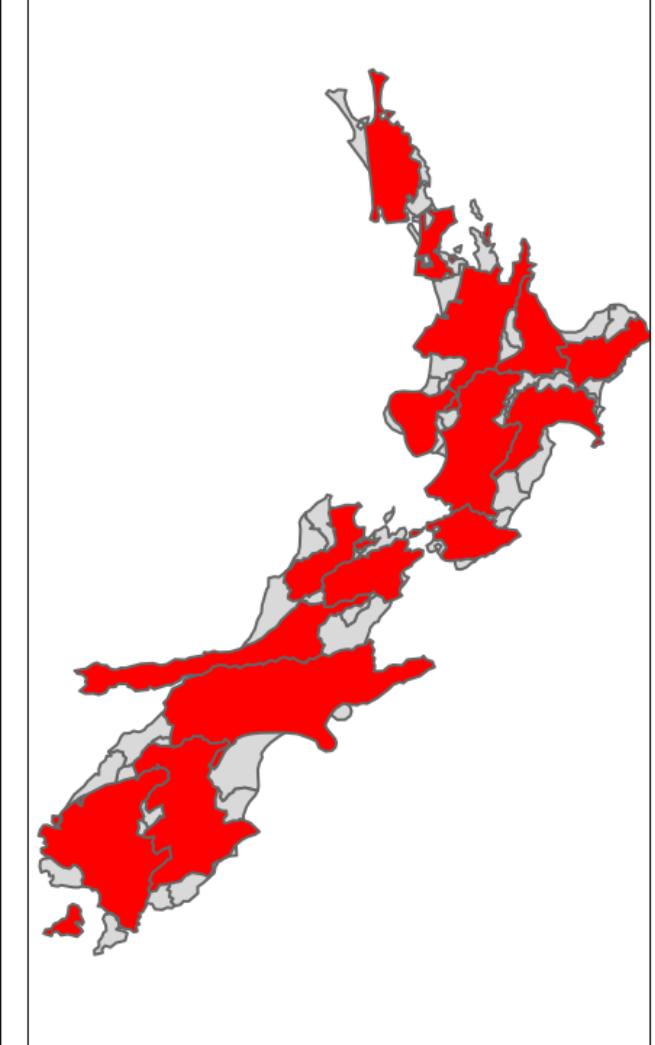
Shift



Scale



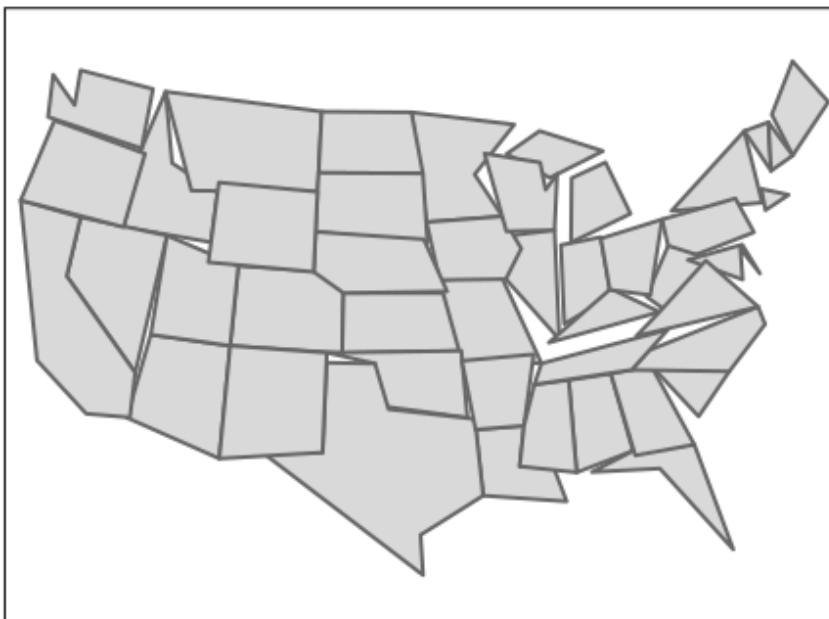
Rotate



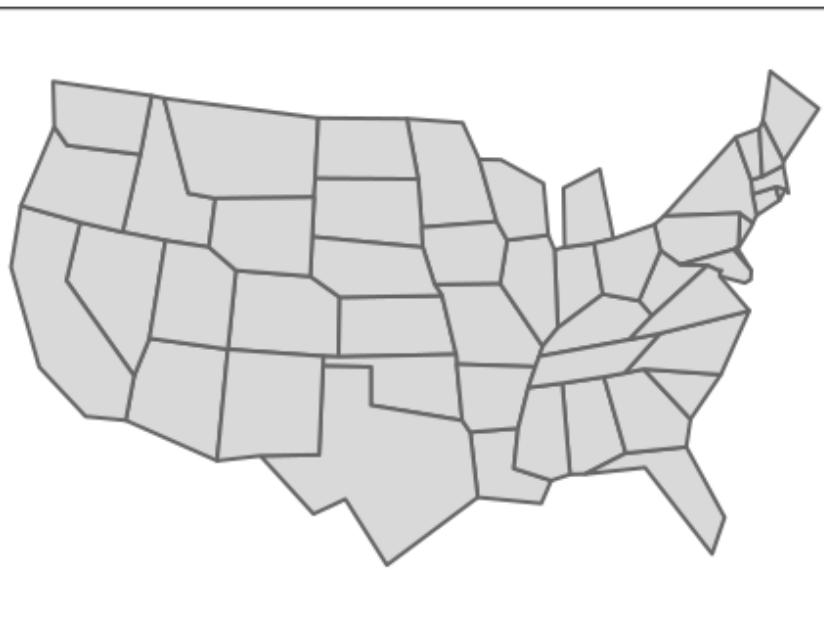
Original data



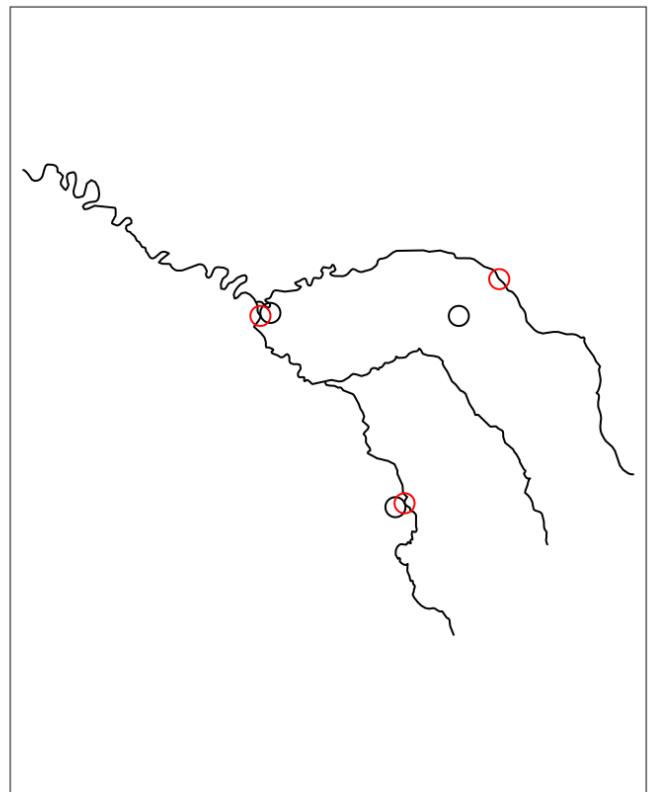
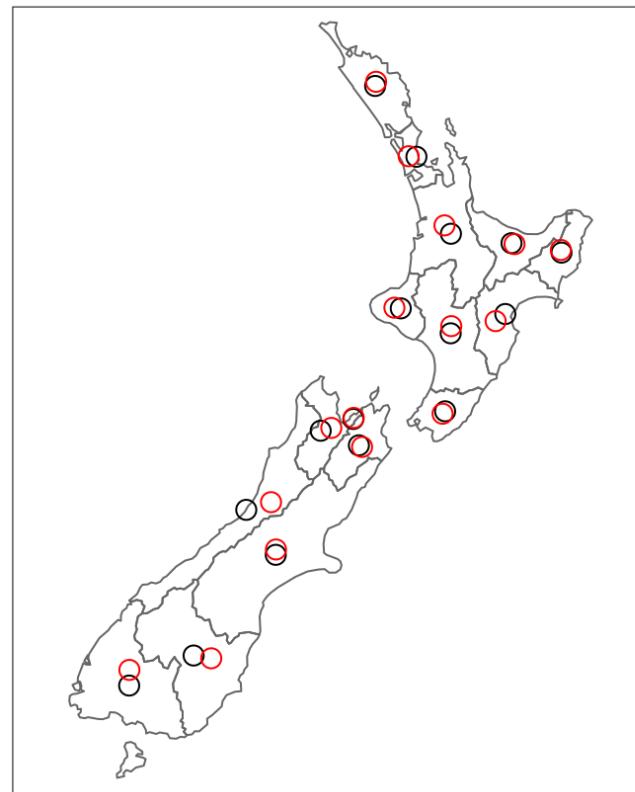
st\_simplify



ms\_simplify



Centroids (black points) and 'points on surface' (red points)



# Spatial vector data

## Basics

```
library(tidyverse)
```

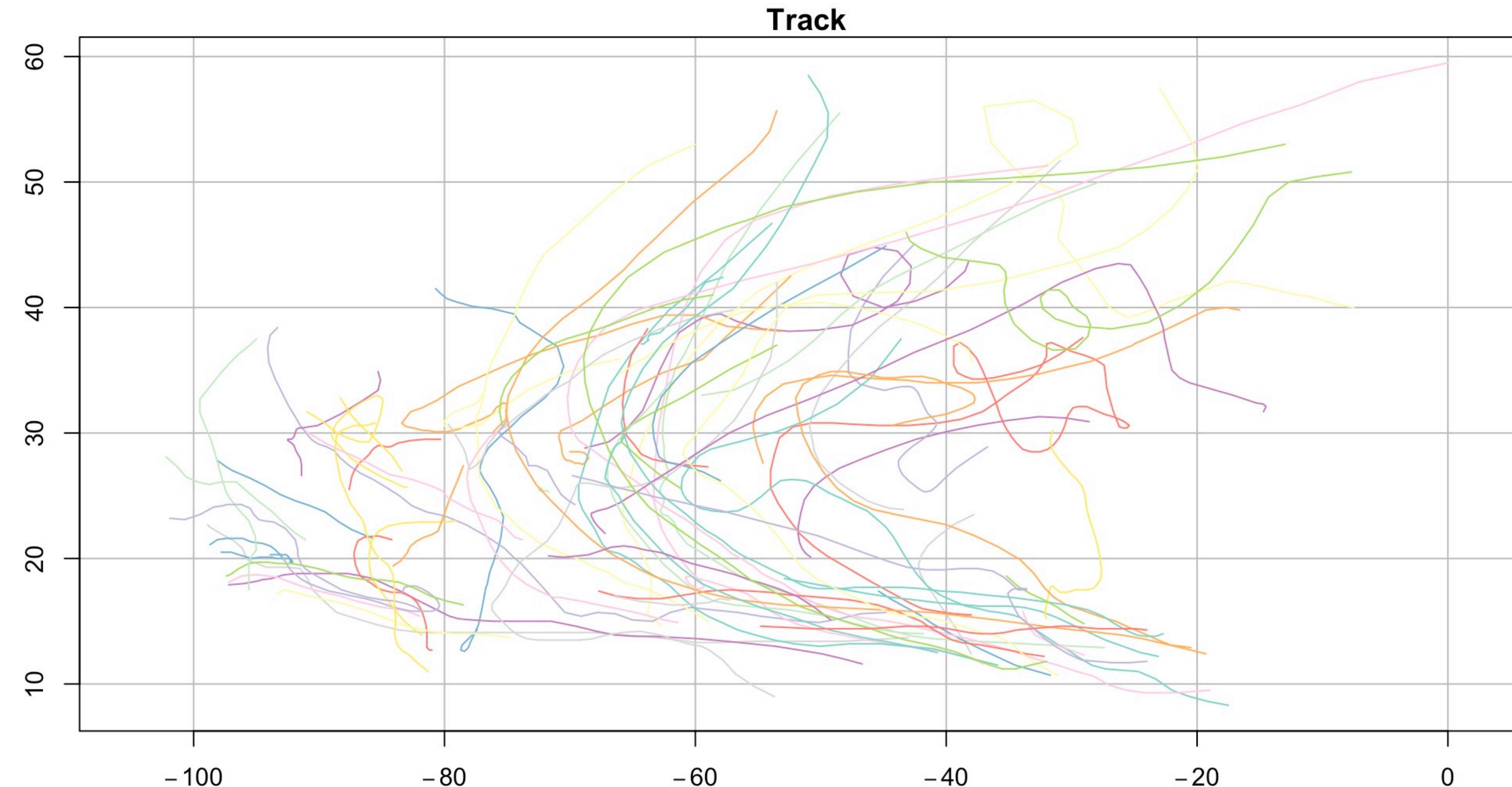
```
library(sf)
```

- Pipes (%>%)  
`a %>% b() %>% c() %>% d()`  
Get **a** do **b** then **c** then **d**
- Packaged data vs. Imported data
- Select and filter

# Spatial vector data

## Line example

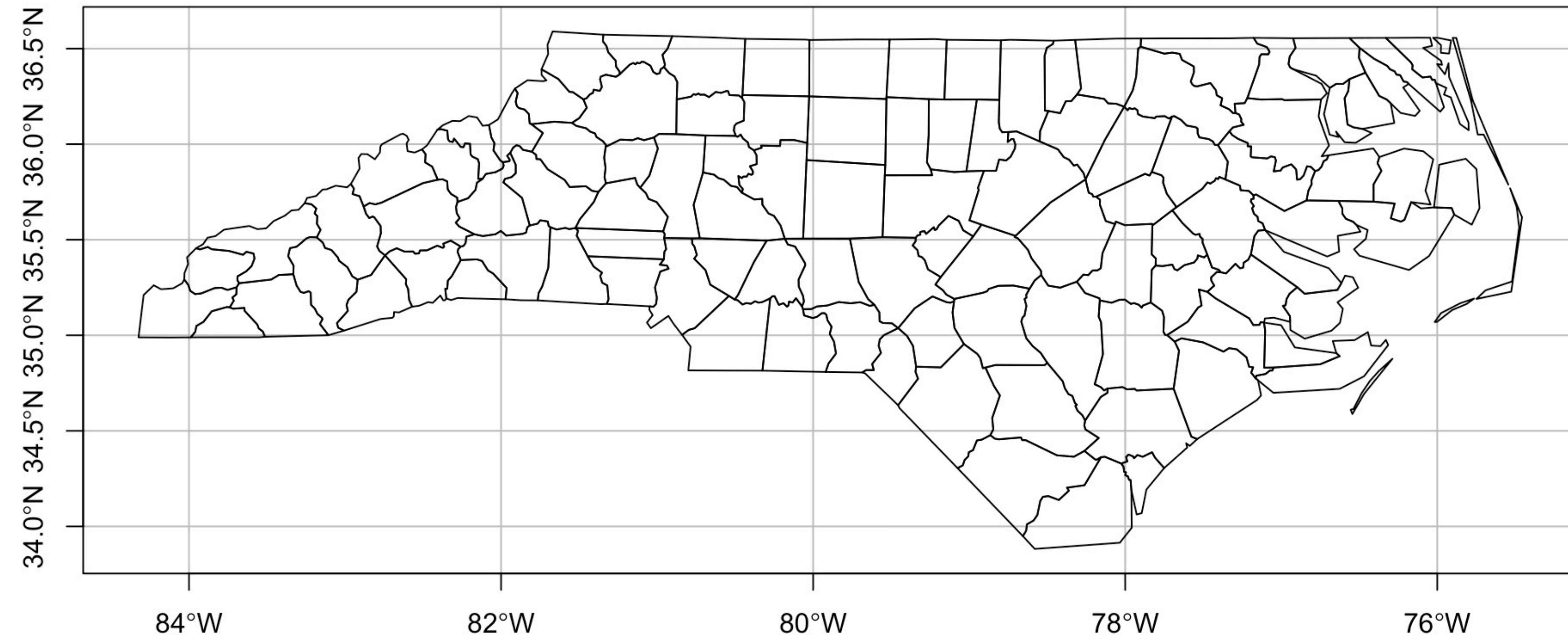
```
storms_xyz_feature <- system.file("shape/storms_xyz_feature.shp", package="sf") %>%  
  st_read() %>%  
  st_as_sf()  
  
plot(storms_xyz_feature, graticule = TRUE, axes = TRUE)
```



# Spatial vector data

## Polygon example

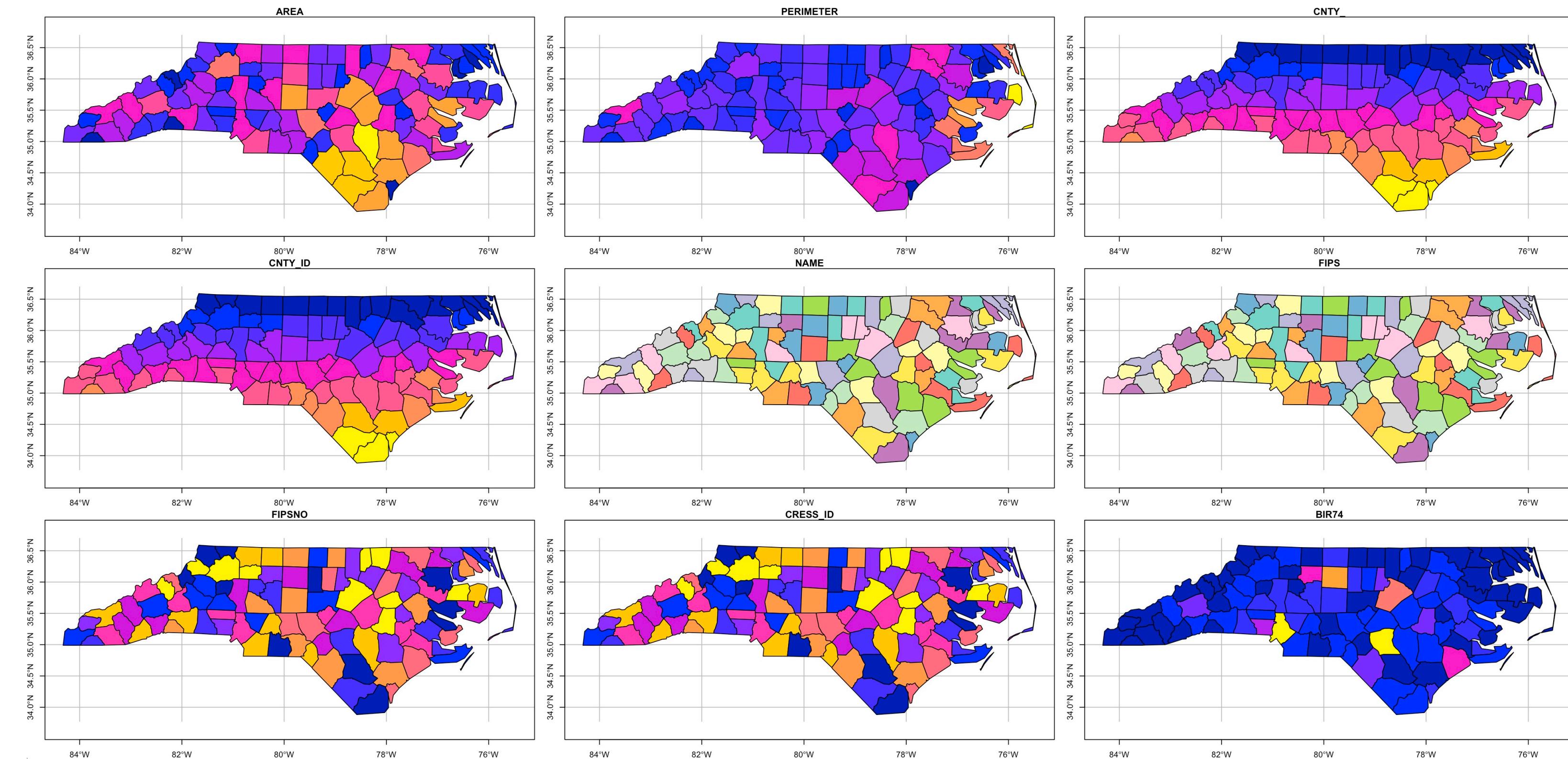
```
nc <- system.file("shape/nc.shp", package="sf") %>%  
  st_read() %>%  
  st_as_sf()  
  
plot(nc$geometry, graticule = TRUE, axes = TRUE)
```



# Spatial vector data

## Attributes

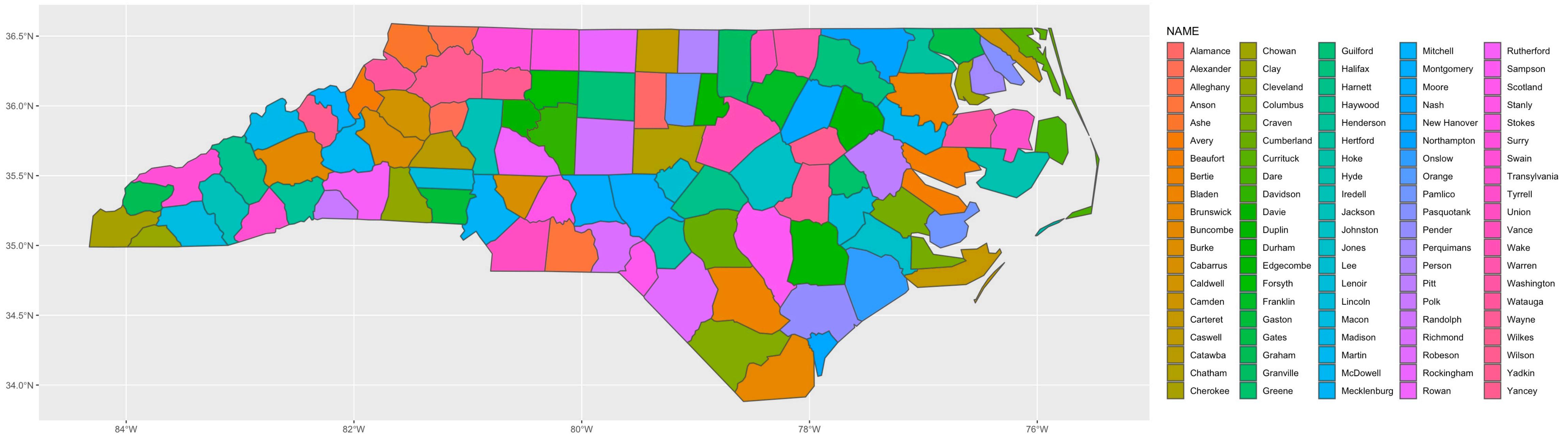
```
nc <- system.file("shape/nc.shp", package="sf") %>%  
  st_read() %>%  
  st_as_sf()  
  
plot(nc, graticule = TRUE, axes = TRUE)
```



# Spatial vector data

## ggplot (more next)

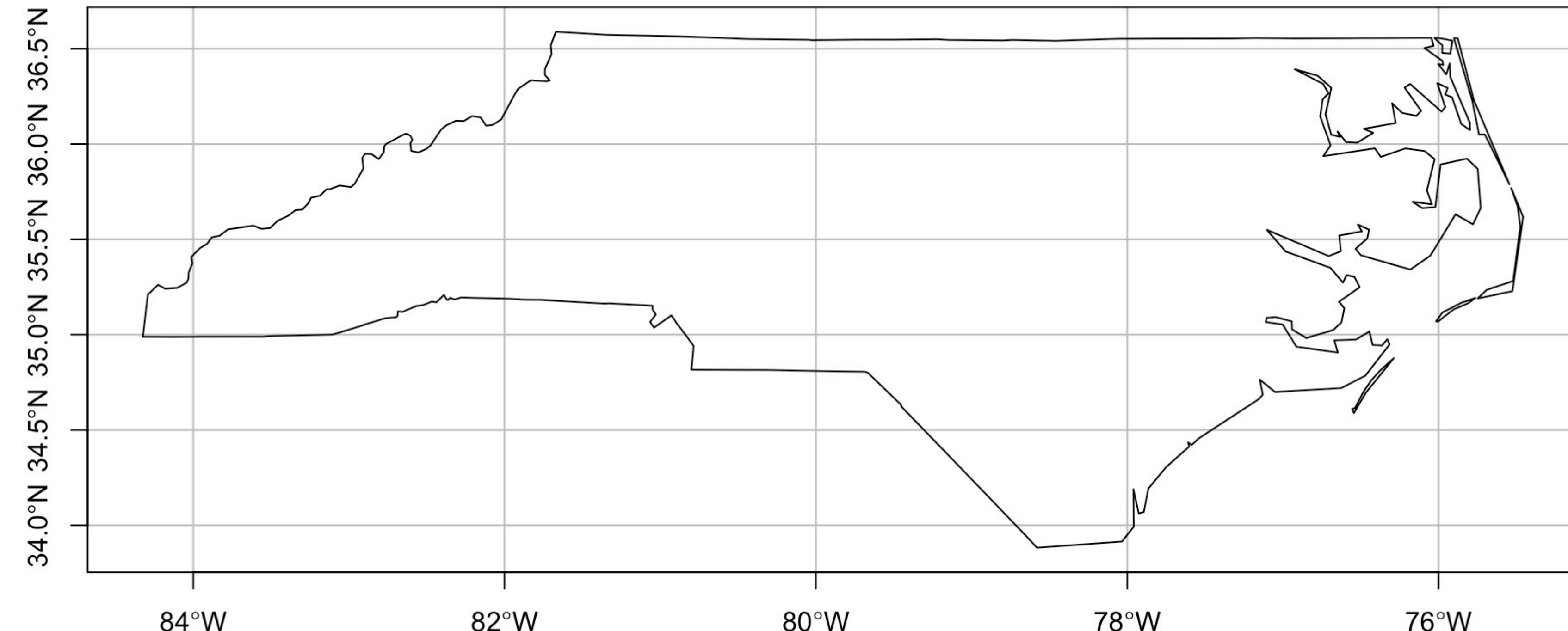
```
ggplot(nc) +  
  geom_sf(aes(fill = NAME))
```



# Spatial vector data

## Dissolve features

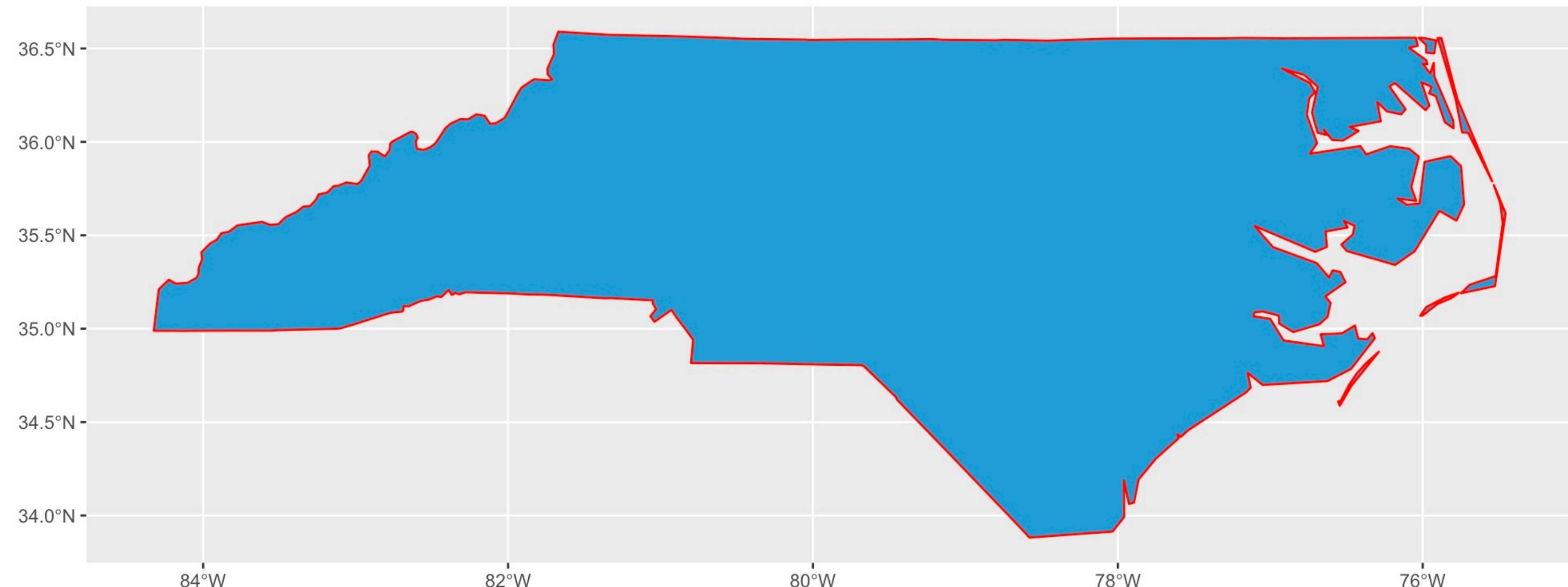
```
nc %>%
  group_by() %>%
  summarise() %>%
  plot(graticule = TRUE, axes = TRUE)
```



# Spatial vector data

## Dissolve features

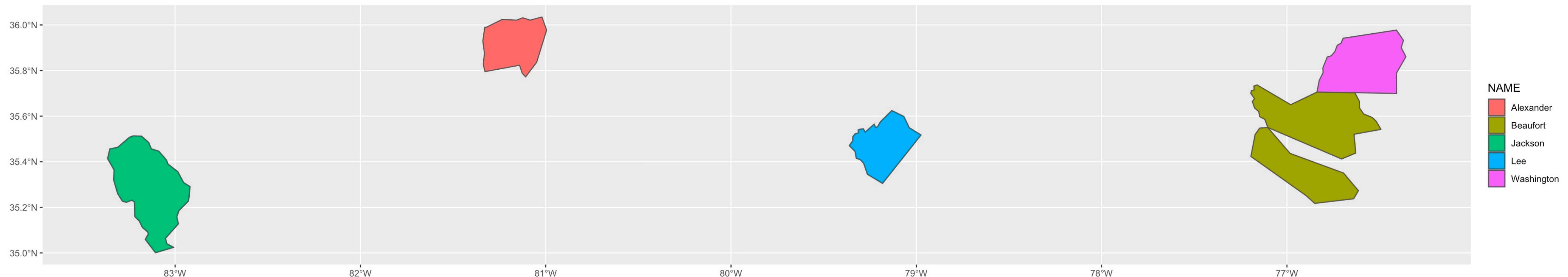
```
nc %>%
  group_by() %>%
  summarise() %>%
  ggplot() +
  geom_sf(fill = "#4B9CD3",
         color = "red")
```



# Spatial vector data

## Filter attributes

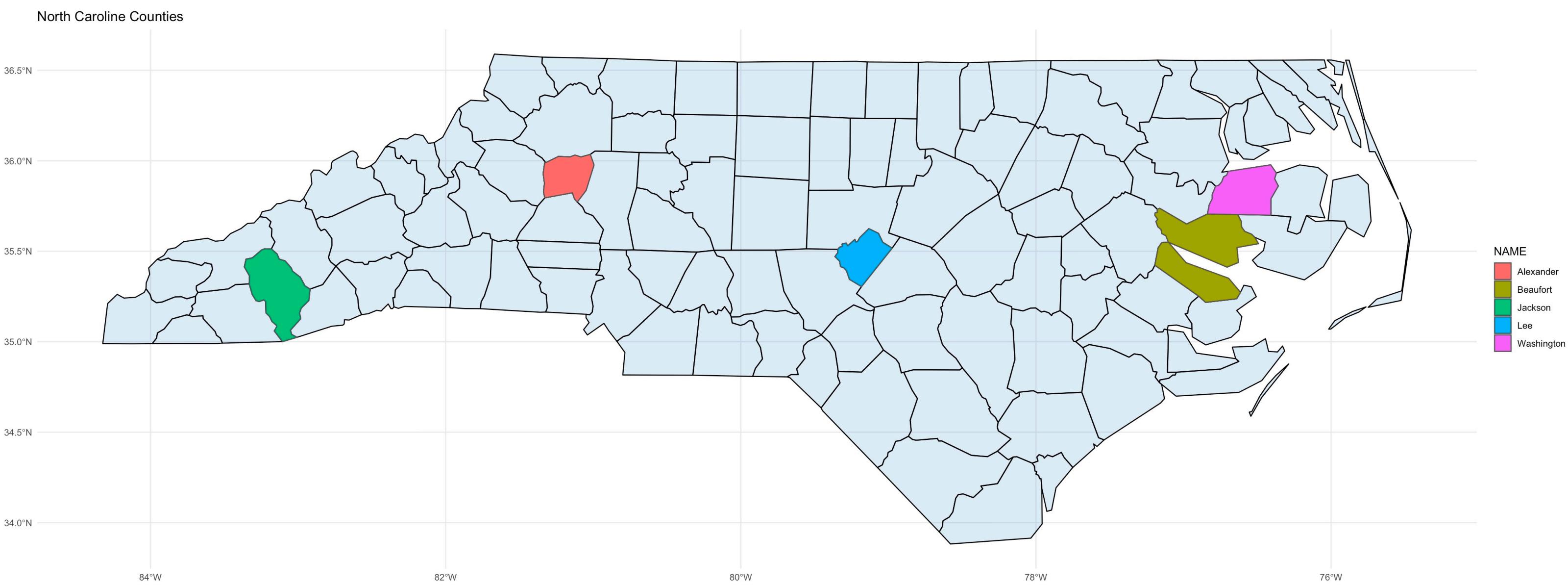
```
nc %>%
  filter(NAME %in% c("Alexander", "Beaufort", "Lee", "Jackson", "Washington")) %>%
  ggplot() +
  geom_sf(aes(fill = NAME))
```



# Mapping

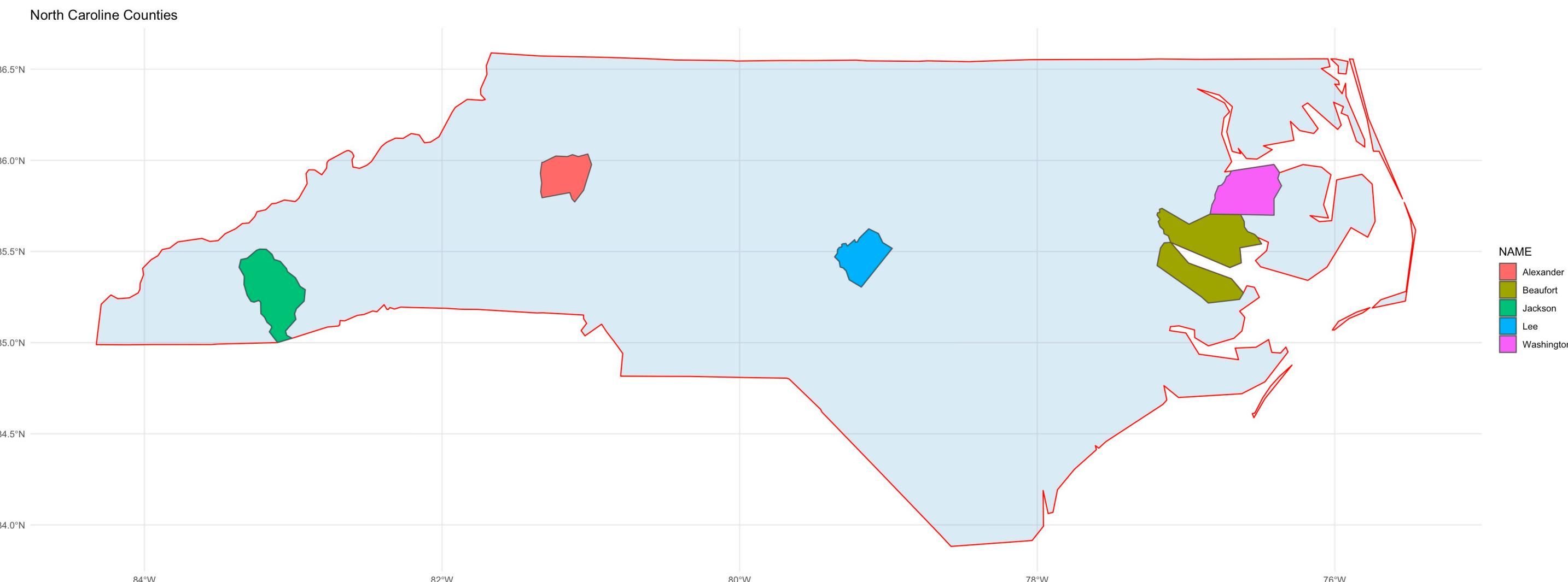
# Mapping ggplot

```
ggplot() +  
  geom_sf(data = nc,  
           fill = "#4B9CD3", alpha = 0.2,  
           color = "black") +  
  geom_sf(data = nc %>%  
           filter(NAME %in% c("Alexander", "Beaufort", "Lee", "Jackson", "Washington")),  
           aes(fill = NAME)) +  
  labs(title = "North Caroline Counties",  
       legend = "Selected Counties") +  
  theme_minimal()
```



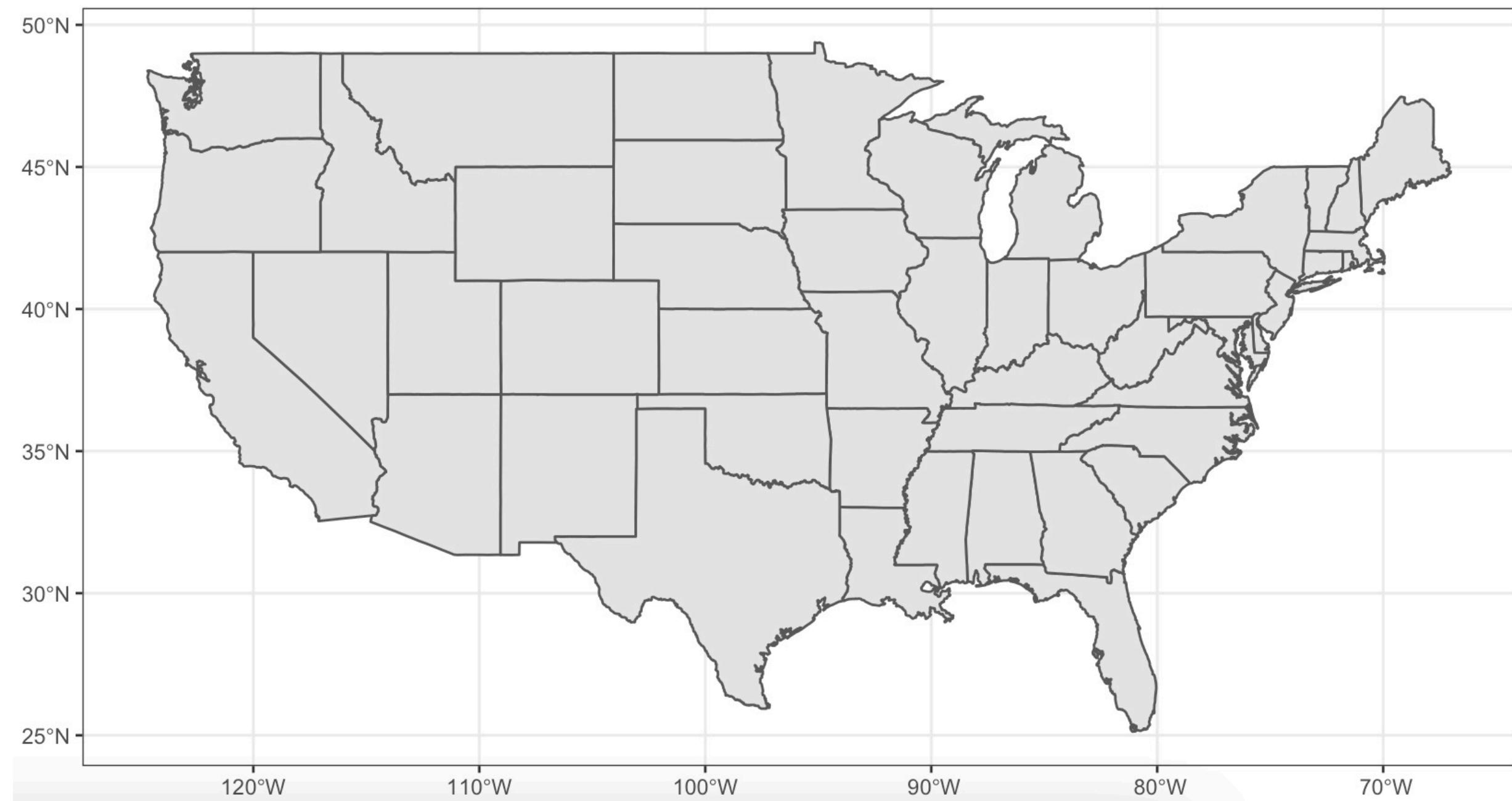
# Mapping ggplot

```
ggplot() +  
  geom_sf(data = nc %>%  
           group_by() %>%  
           summarise(),  
           fill = "#4B9CD3", alpha = 0.2,  
           color = "red") +  
  geom_sf(data = nc %>%  
           filter(NAME %in% c("Alexander", "Beaufort", "Lee", "Jackson", "Washington")),  
           aes(fill = NAME)) +  
  labs(title = "North Caroline Counties",  
       legend = "Selected Counties") +  
  theme_minimal()
```



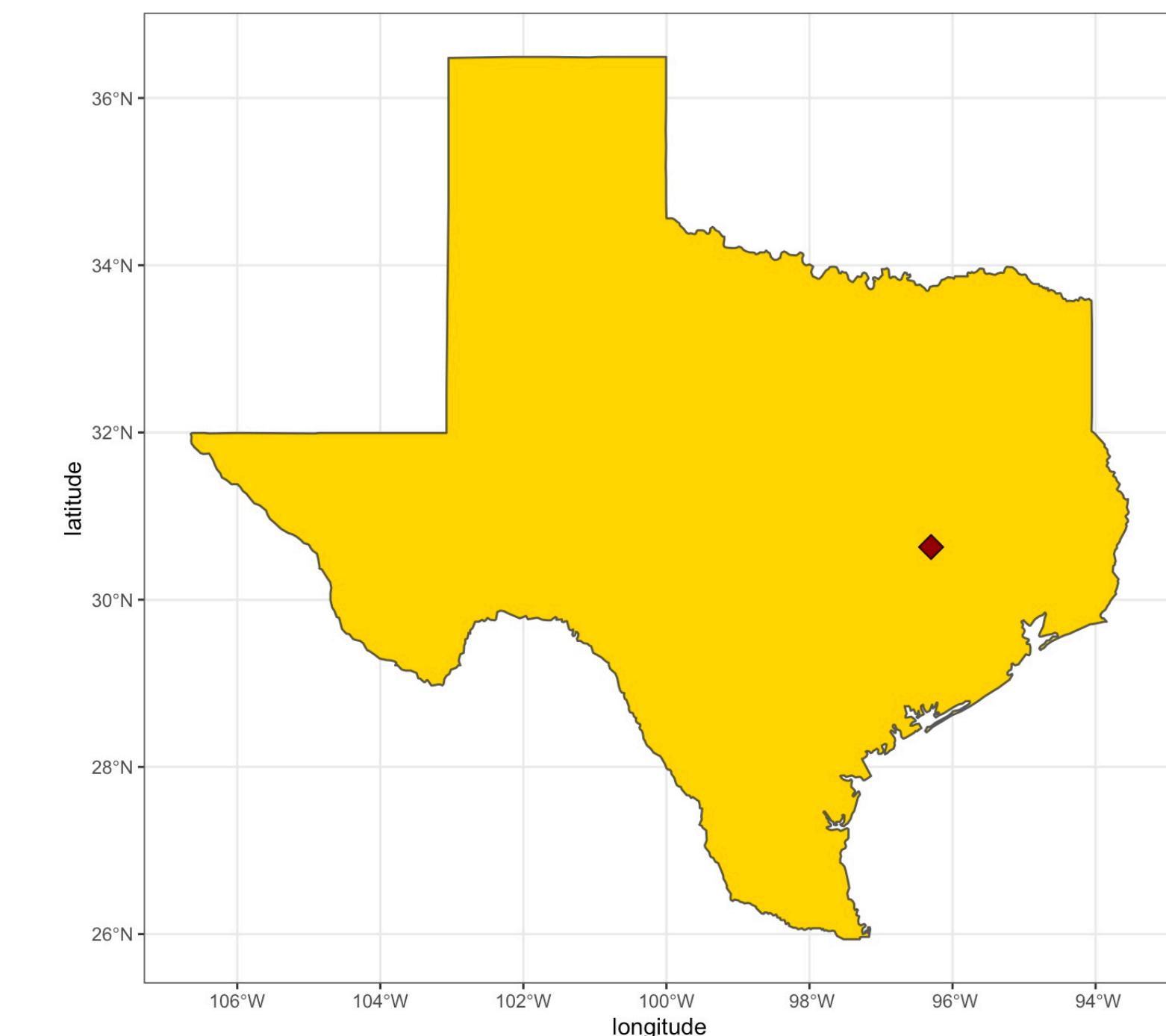
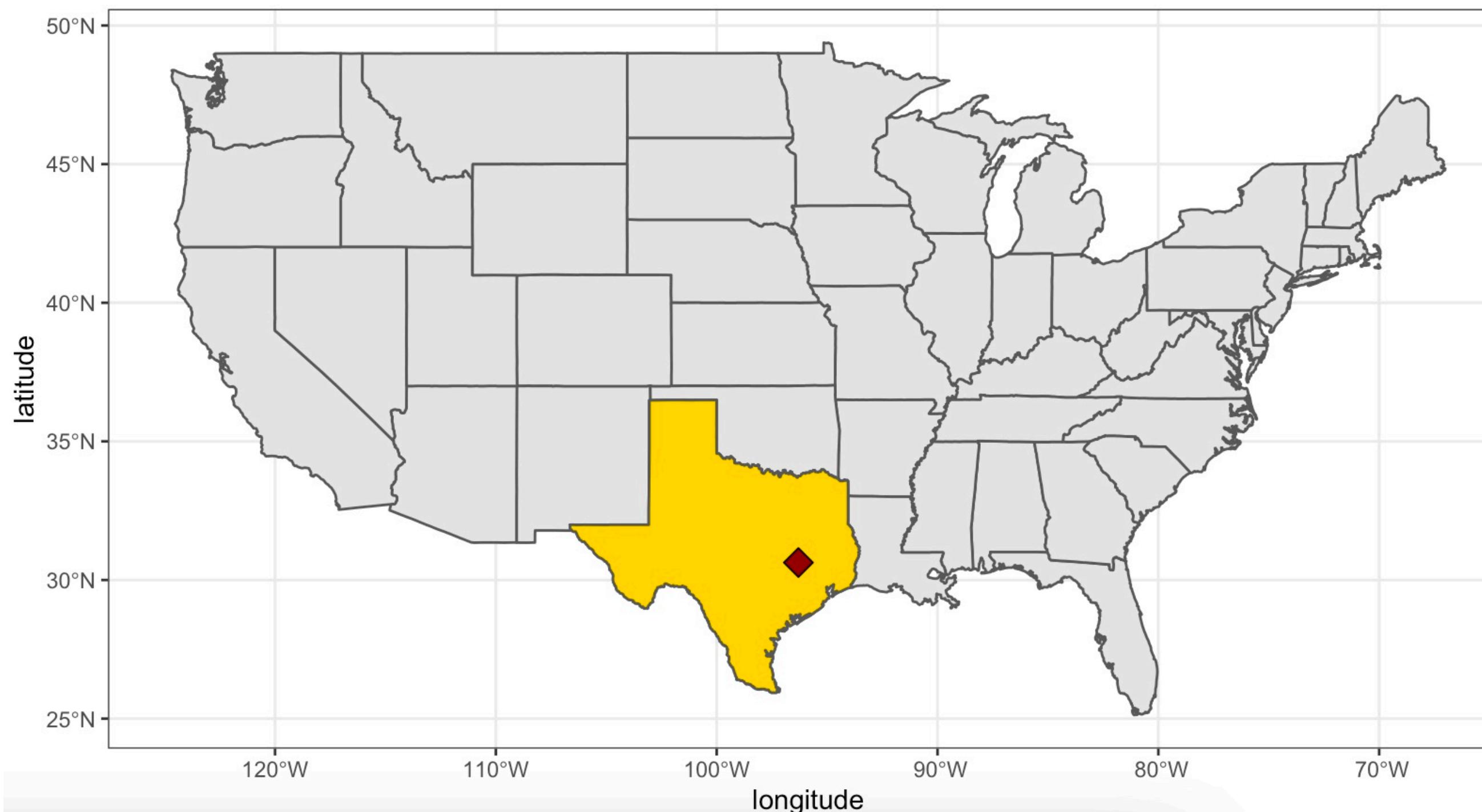
# Mapping ggplot

```
library(maps)  
  
some_point <- data.frame(longitude = c(-96.30),  
                           latitude  = c(30.63))  
  
us <- st_as_sf(map("state", plot = FALSE, fill = TRUE))  
us  
  
class(us)  
  
ggplot(data = us) +  
  geom_sf()
```



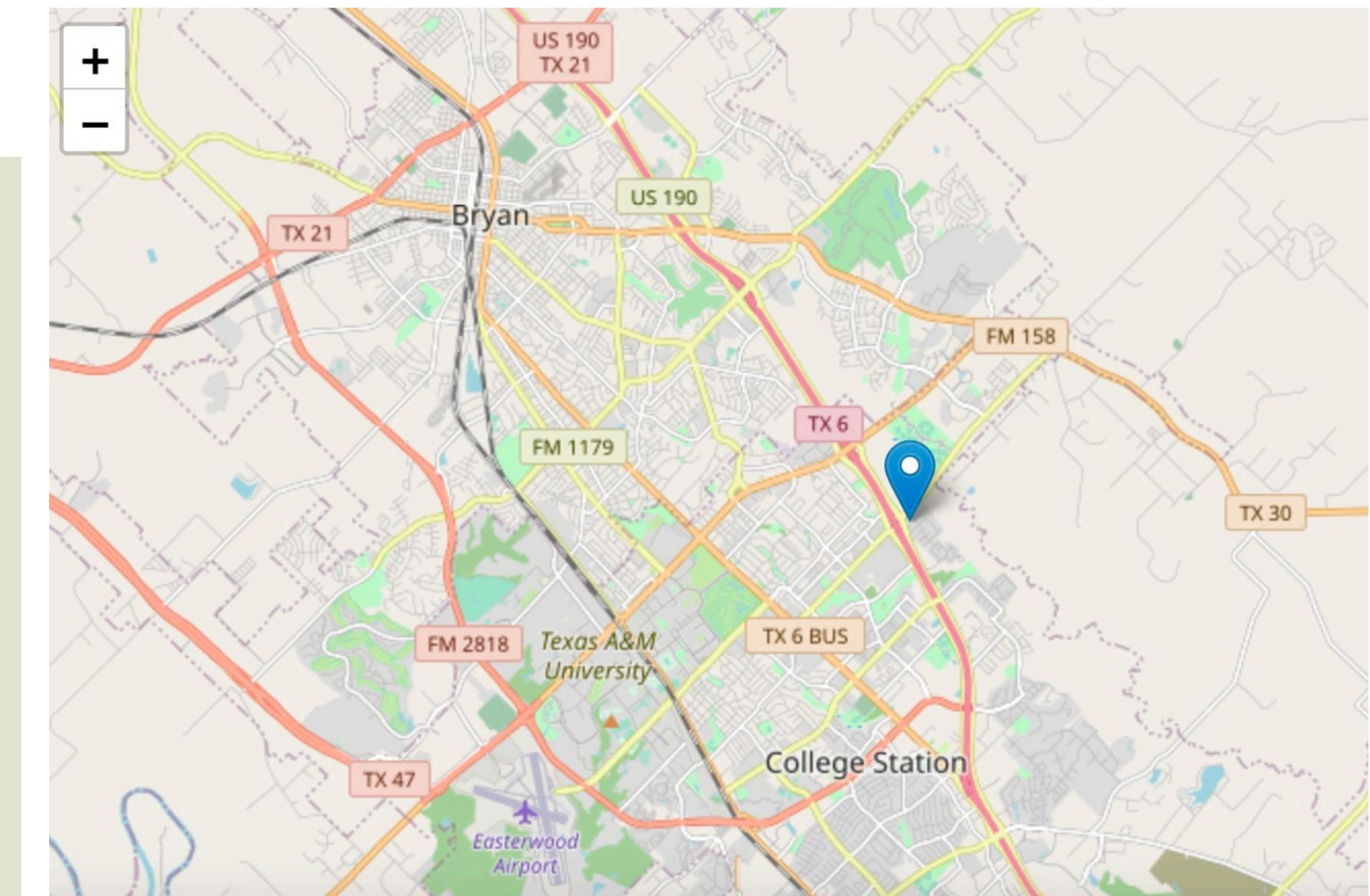
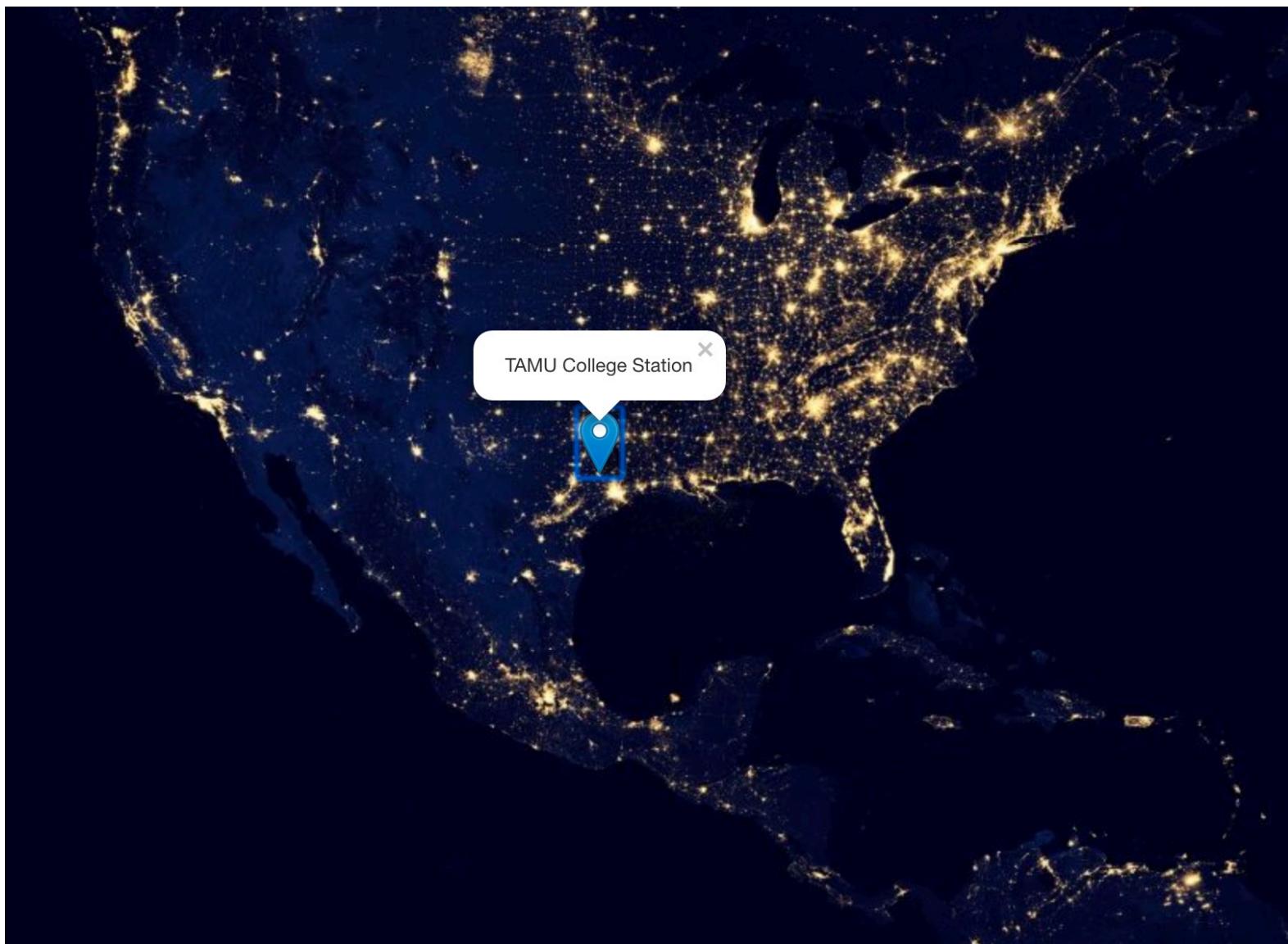
# Mapping ggplot

```
ggplot() +  
  geom_sf(data = us) +  
  # geom_sf(data = us %>% filter(ID == "texas"), fill = "gold") +  
  geom_point(data = some_point,  
             aes(x = longitude, y = latitude),  
             size = 4, shape = 23, fill = "darkred")
```



# Mapping leaflet

```
library(leaflet)  
  
TxGIS = c("TAMU College Station")  
  
leaflet() %>%  
addProviderTiles("NASAGIBS.ViirsEarthAtNight2012") %>%  
addMarkers(lng = c(-96.3),  
          lat = c(30.6),  
          popup = TxGIS)
```



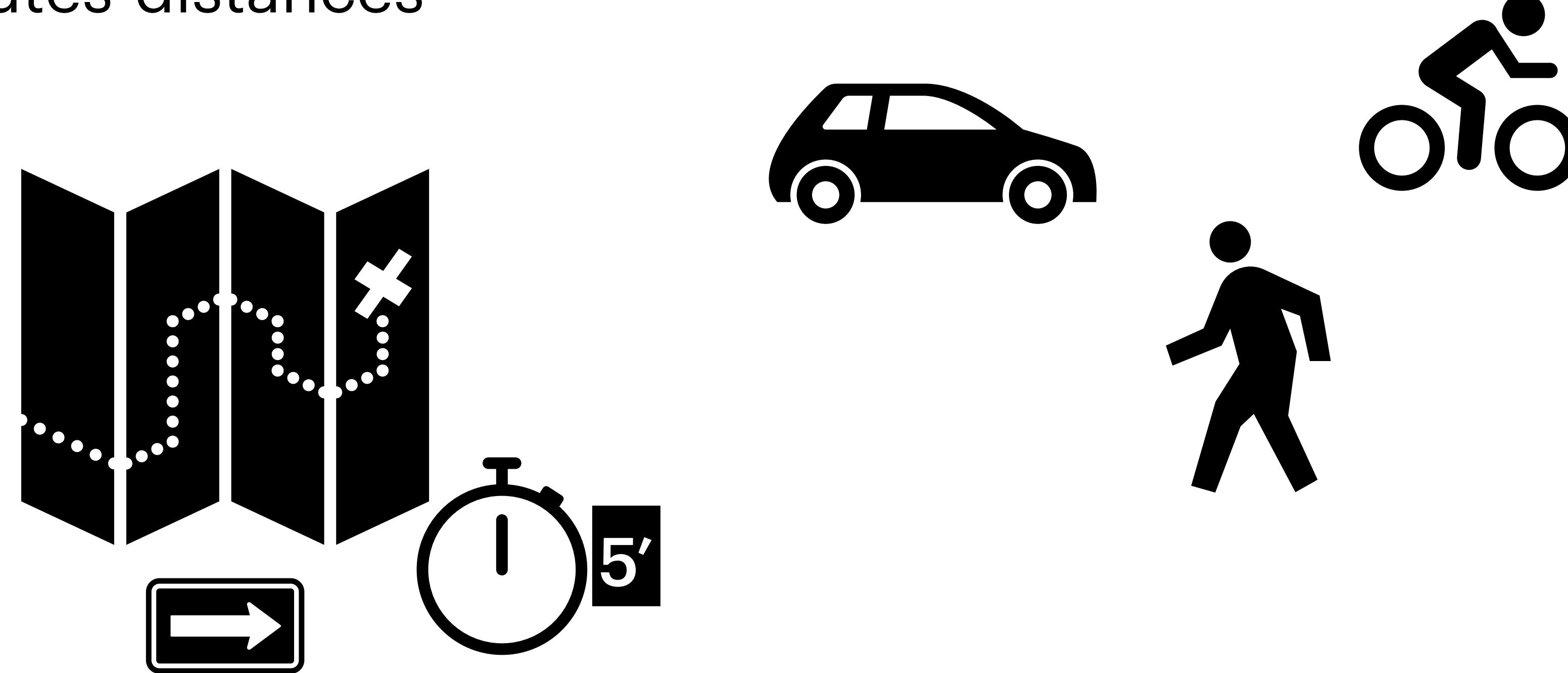
- Check other base-maps / tiles provider  
<http://leaflet-extras.github.io/leaflet-providers/preview/>

# **Walking, cycling, and driving time-distances**

# Walking, cycling, and driving time-distances

Goal

- Define a location
- Find 5 minutes-distances



# Walking, cycling, and driving time-distances

## Step 1: install libraries and get an API

```
remotes::install_github("walkerke/mapboxapi")  
  
library(mapboxapi)  
  
mb_access_token("pk.ey...", install = TRUE)
```

- To get started, sign up for a Mapbox account and generate an access token.
- Set your public or secret token for use in the package with `mb_access_token()`
- Once you've set your token, you are ready to get started using the package.

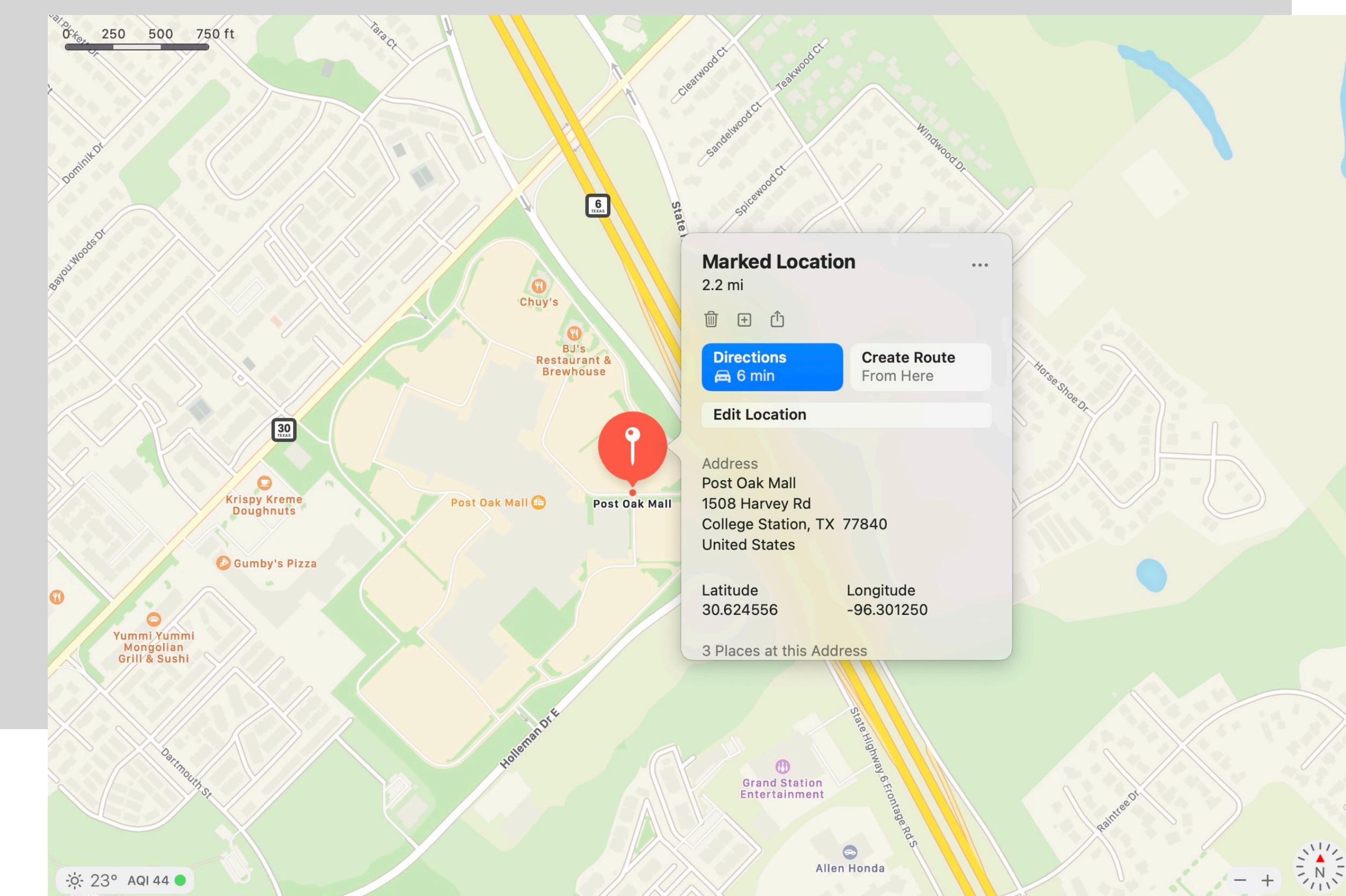
<https://docs.mapbox.com/api/>

<https://account.mapbox.com>

# Walking, cycling, and driving time-distances

## Step 2: geocode an address

```
address <- "Texas A&M University, Langford Architecture Bldg 3137, College Station, TX 77843"  
  
address <- c(-96.301250, 30.624556)  
  
walk_5min <- mb_isochrone(address,  
                           profile = "walking",  
                           time = 5)  
  
bike_5min <- mb_isochrone(address,  
                           profile = "cycling",  
                           time = 5)  
  
drive_5min <- mb_isochrone(address,  
                           profile = "driving",  
                           time = 5)
```



# Walking, cycling, and driving time-distances

# Step 3: pick a basemap

# Walking, cycling, and driving time-distances

## Step 3: pick a basemap

```
ggmap(map) +  
  theme_void() +  
  theme(plot.title = element_text(colour = "orange"),  
        panel.border = element_rect(colour = "black",  
                                     fill = NA, size = .7))
```



# Walking, cycling, and driving time-distances

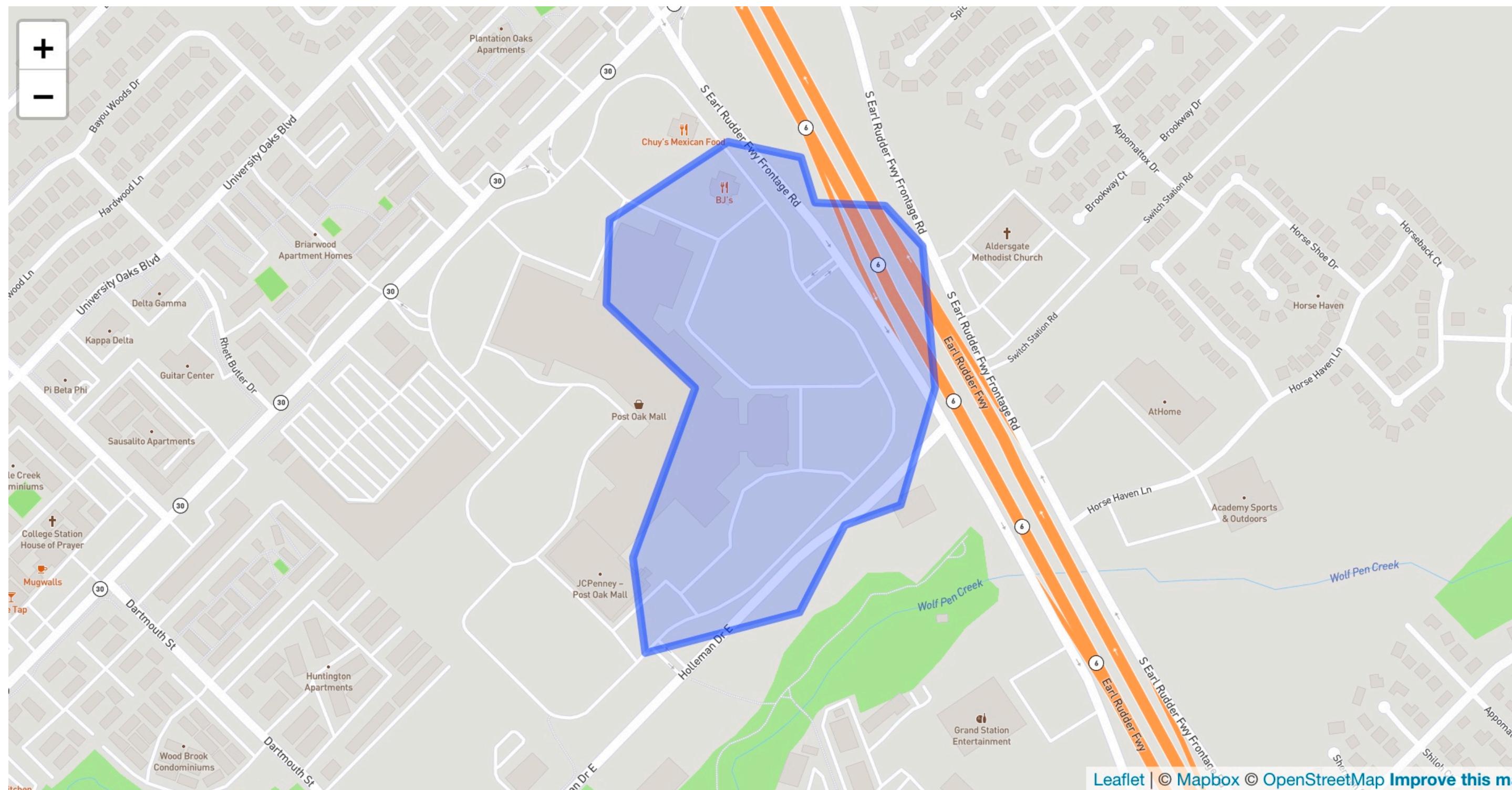
## Step 4: the distance map

```
ggmap(map) +
  geom_sf(data = drive_5min,
          aes(fill = "driving"),
          color = "white", alpha=0.6, linetype = "blank",
          show.legend = TRUE, inherit.aes = FALSE) +
  geom_sf(data=bike_5min,
          aes(fill="cycling"),
          color="white", alpha=0.8, linetype = "blank",
          show.legend = TRUE, inherit.aes = FALSE) +
  geom_sf(data=walk_5min,
          aes(fill="walking"),
          color="white", alpha=0.7, linetype = "blank",
          show.legend = TRUE, inherit.aes = FALSE) +
  scale_fill_manual(values = c("walking" = "green",
                               "cycling" = "blue2",
                               "driving" = "red"),
                     breaks = c("walking", "cycling", "driving")) +
  labs(x="",
       y="",
       title="5-minute time-distances",
       fill="") +
  theme(legend.justification="left",
        panel.background=element_rect(fill="grey", colour="grey", size=0.5, linetype="solid"),
        panel.grid.major=element_line(size=0.2, linetype="solid", colour="white"),
        panel.grid.minor=element_line(size=0.2, linetype="solid", colour="white")) # -> temp2
```

# Walking, cycling, and driving time-distances

## Step 5: an interactive Leaflet map

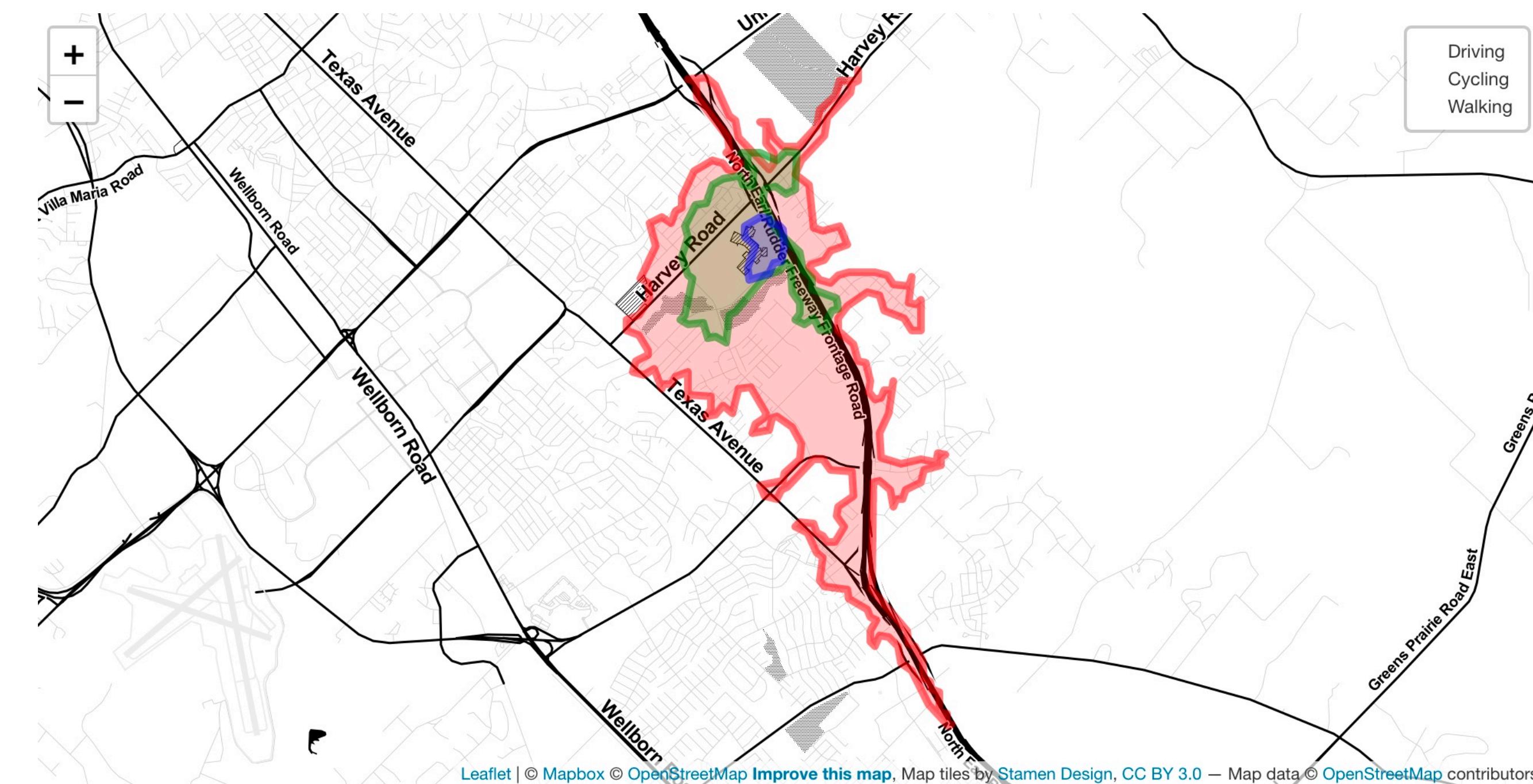
```
leaflet(walk_5min) %>%
  addMapboxTiles(style_id = "streets-v11",
                 username = "mapbox") %>%
  addPolygons()
```



# Walking, cycling, and driving time-distances

## Step 6: a more complex interactive Leaflet map

```
leaflet() %>%
  addMapboxTiles(style_id = "streets-v11",
                 username = "mapbox") %>%
  addProviderTiles(providers$Stamen) %>% # Stamen Stamen.Toner Stamen.TonerHybrid OpenTopoMap Esri.WorldImagery
  addPolygons(data = drive_5min, color = "red", group = "Driving") %>%
  addPolygons(data = bike_5min, color = "green", group = "Cycling") %>%
  addPolygons(data = walk_5min, color = "blue", group = "Walking") %>%
  addLayersControl(overlayGroups = c("Driving", "Cycling", "Walking"),
                  options = layersControlOptions(collapsed = FALSE)) # -> m
```

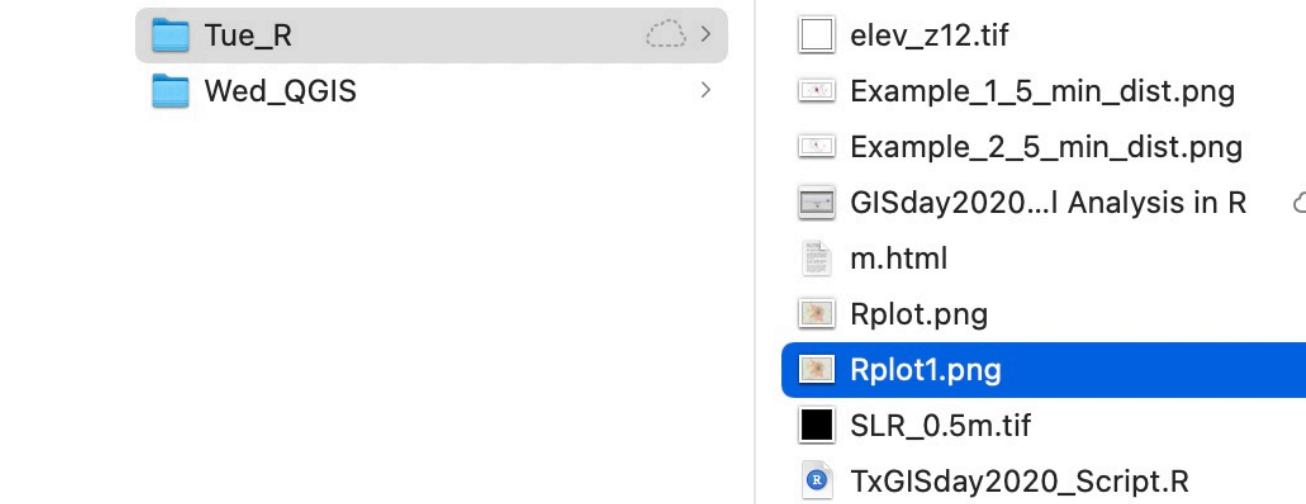


# Walking, cycling, and driving time-distances

## Step 7: save it

```
library(htmlwidgets)
saveWidget(m, file="m.html")

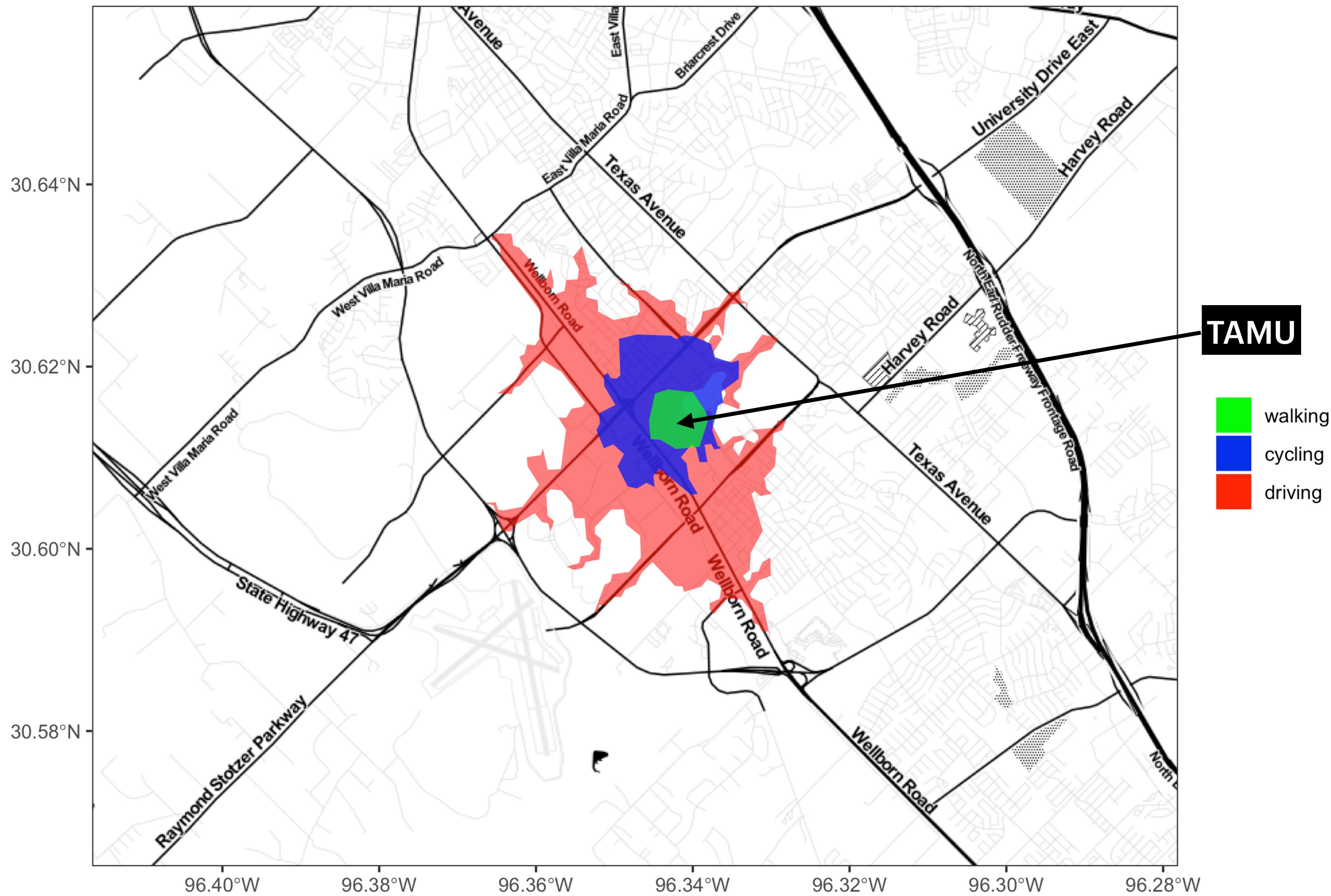
library(webshot)
webshot("m.html", file = "Rplot1.png", cliprect = "viewport")
```



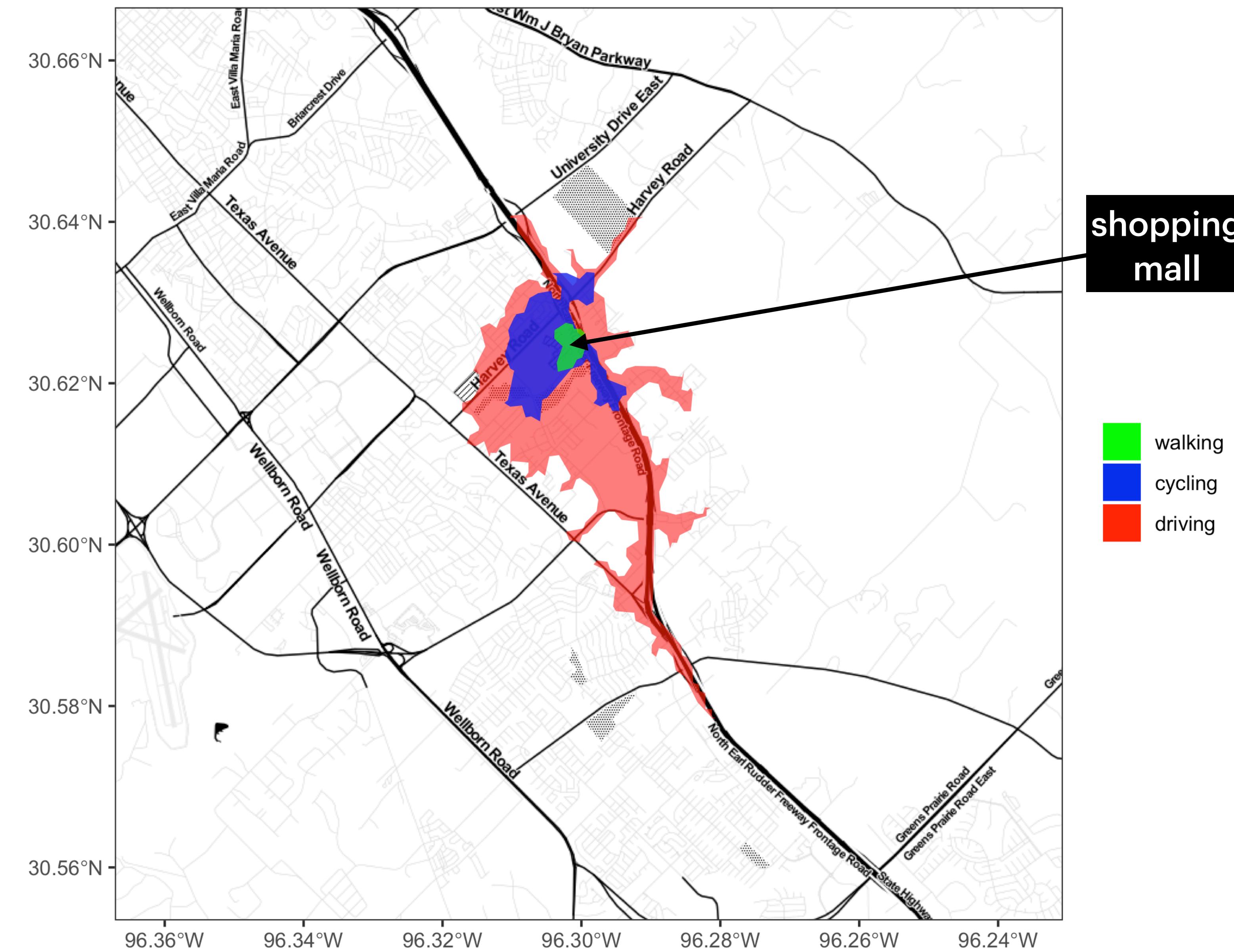
A screenshot of a file explorer window. On the left, there's a sidebar with 'Tue\_R' and 'Wed\_QGIS' folders. The main area shows several files: 'elev\_z12.tif', 'Example\_1\_5\_min\_dist.png', 'Example\_2\_5\_min\_dist.png', 'GISday2020...l Analysis in R', 'm.html', 'Rplot.png', 'Rplot1.png' (which is highlighted with a blue selection bar), 'SLR\_0.5m.tif', and 'TxGISday2020\_Script.R'. To the right of the file list is a map visualization titled 'Rplot1.png'. The map displays a city area with various colored regions (yellow, red, green) representing different travel zones or distances. A legend in the top right corner of the map interface includes options for 'Driving', 'Cycling', and 'Walking'. Below the map is a detailed file information panel:

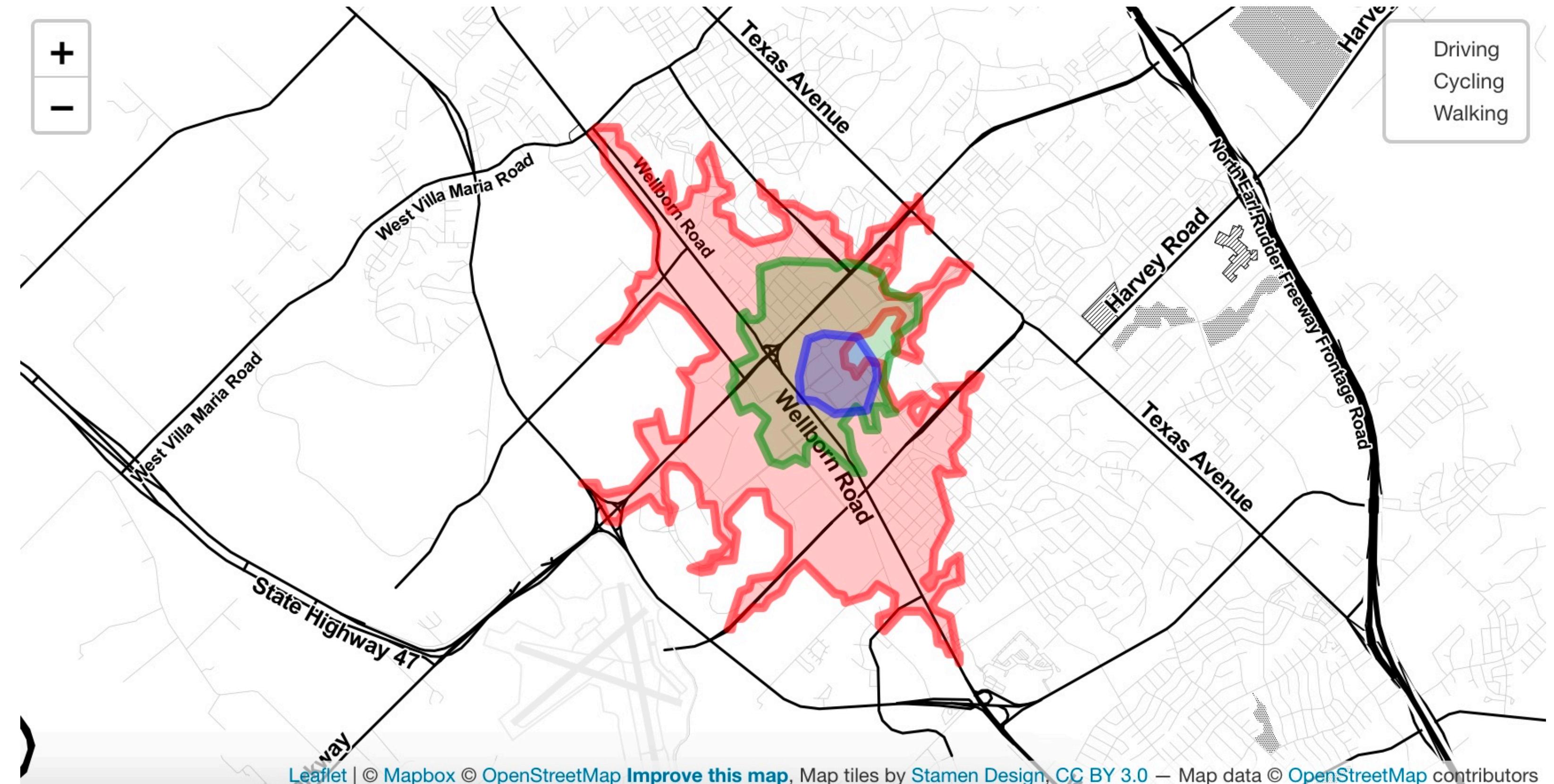
**Rplot1.png**  
PNG image - 735 KB  
**Information** Show More  
Created Today, 1:53 AM  
Modified Today, 1:53 AM  
Dimensions 992x744  
Resolution 72x72

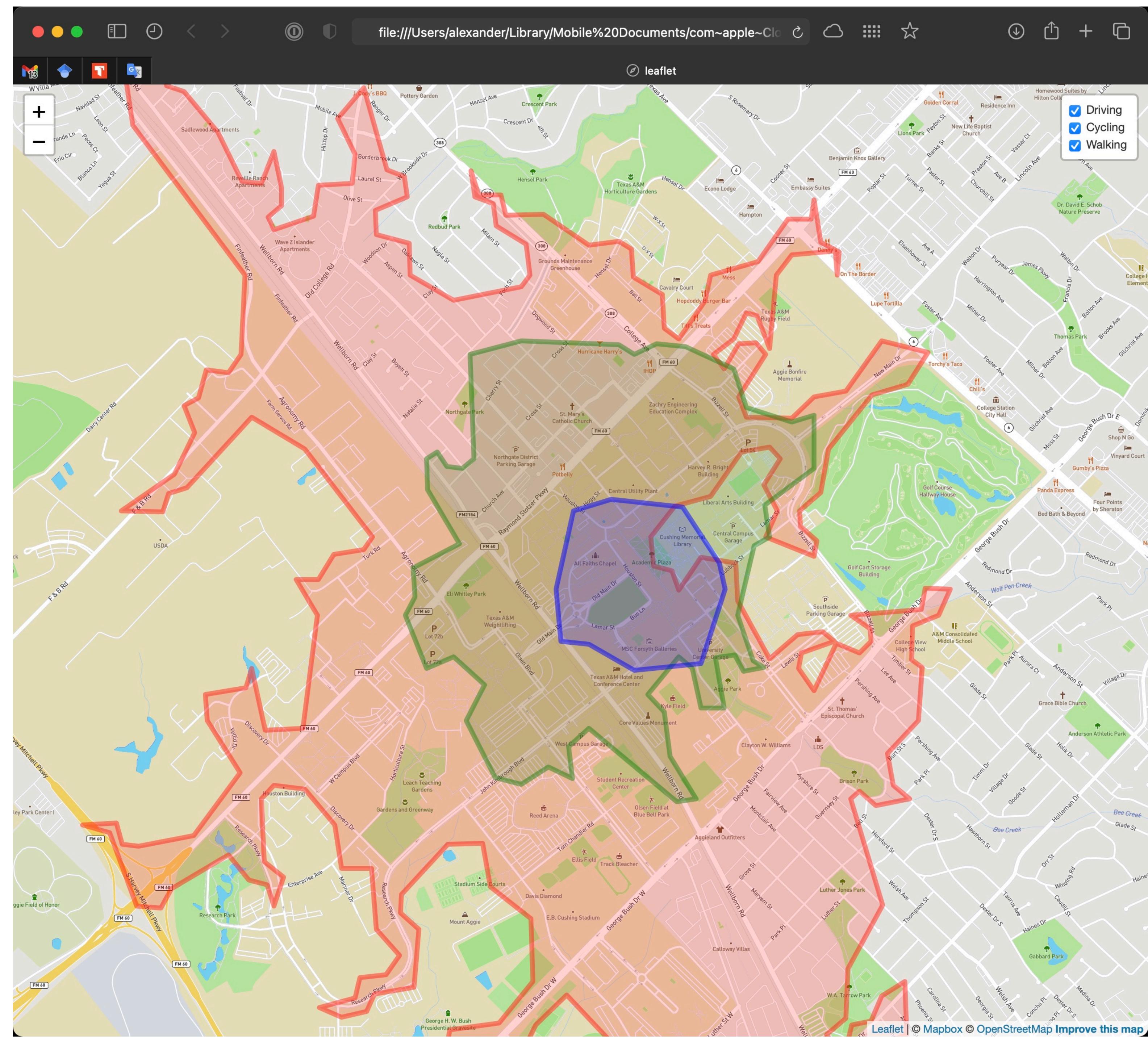
## 5-minute time-distances



## 5-minute time-distances







# Elevation analysis

# Elevation analysis

## Step 1: Load packages, and define a projection and boundary box

```
if(!require(pacman)){install.packages("pacman"); library(pacman)}
p_load(dplyr, elevatr, haven, lwgeom, maps, raster, rgdal, rgeos,
sf, sp, tigris)

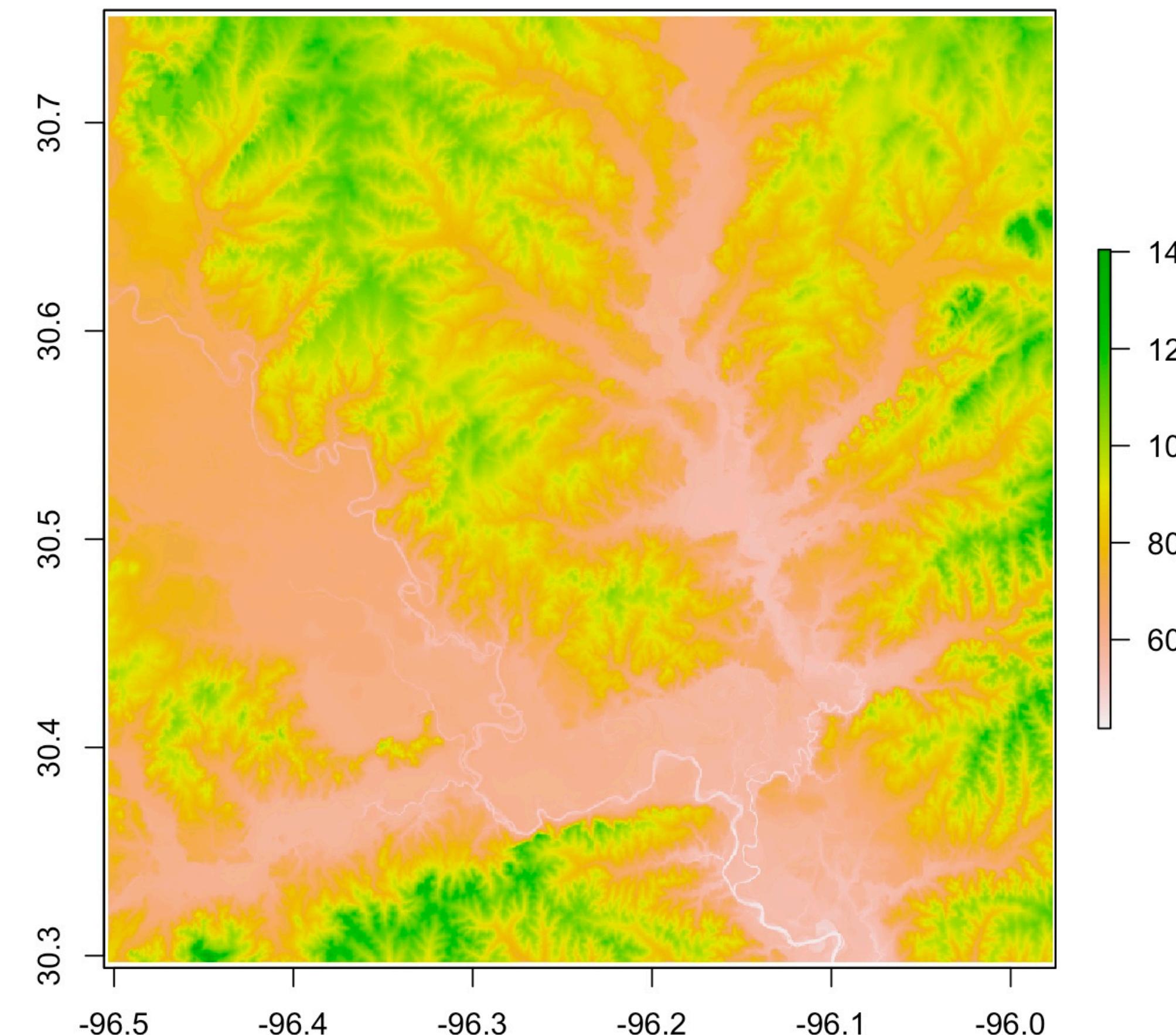
bbox

aux <- as.numeric(bbox)
e <- as(raster::extent(aux[1], aux[3], aux[2], aux[4]), "SpatialPolygons")
proj4string(e) <- st_crs(drive_5min)$proj4string
# proj4string(e) <- CRS("+proj=utm +zone=10 +datum=WGS84")
```

# Elevation analysis

## Step 2: Get elevation

```
x <- get_elev_raster(e, prj=sp::proj4string(e), z=11, src="aws") # 1 to 14  
plot(x)
```



# Elevation analysis

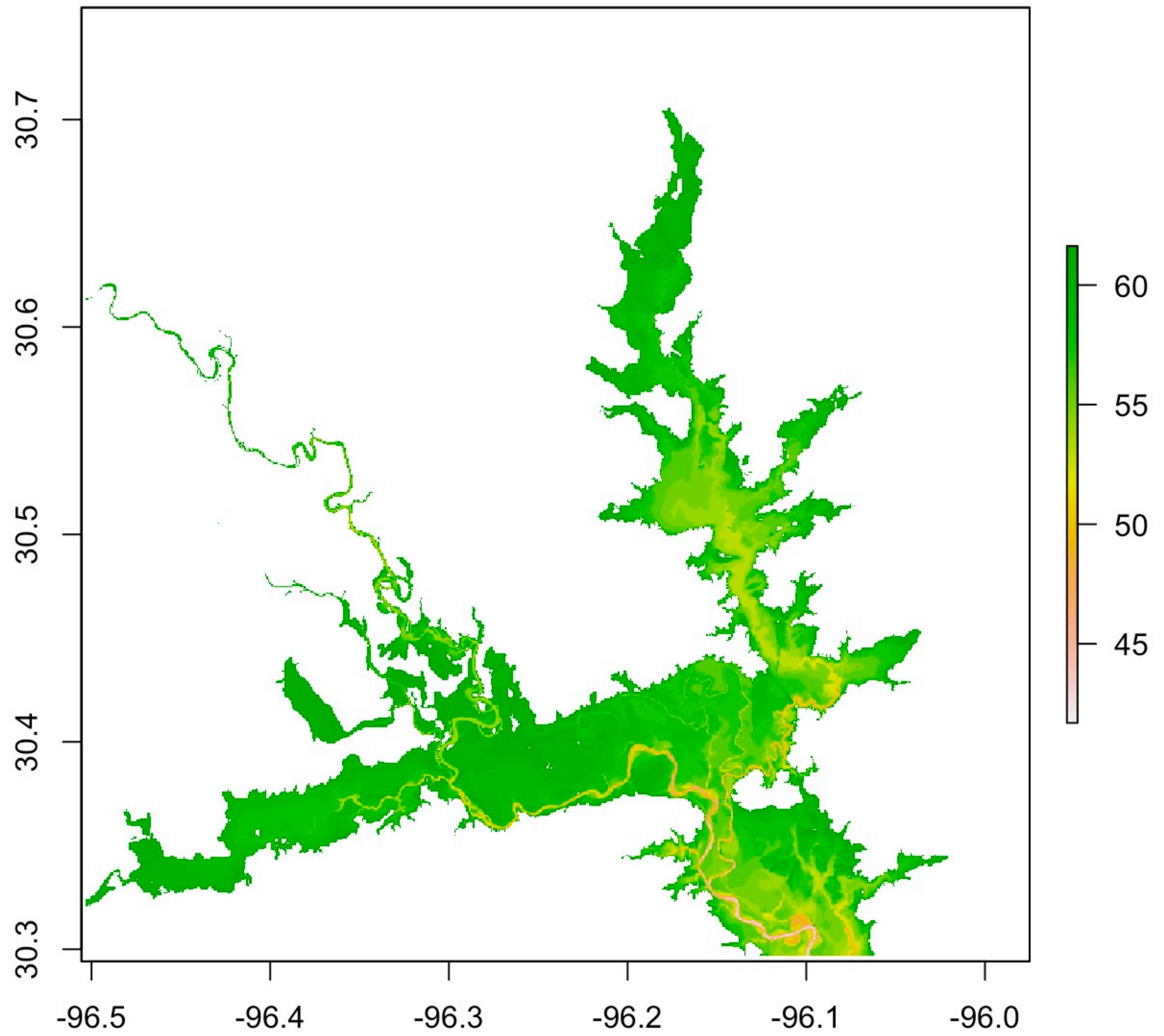
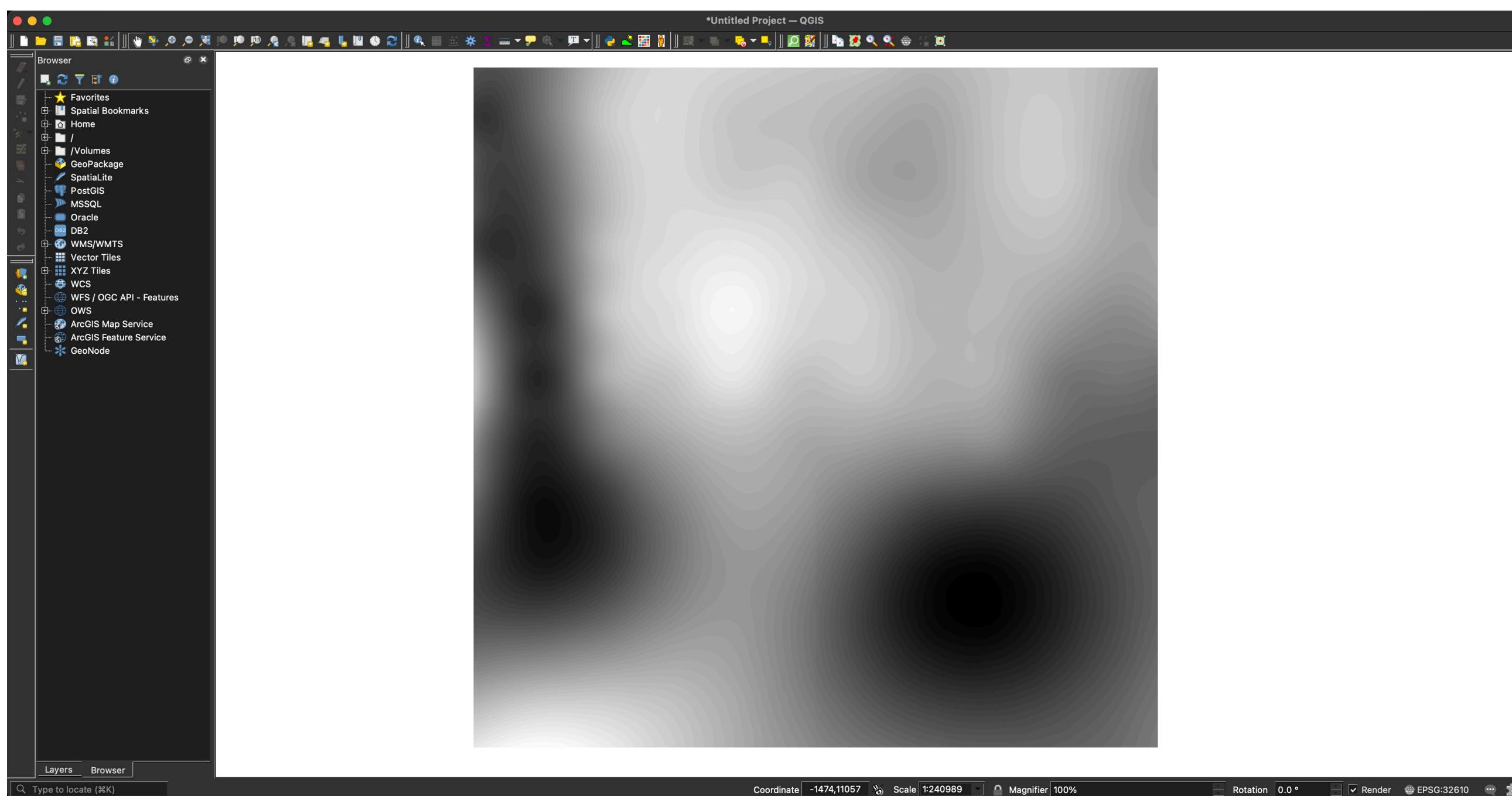
## Step 3: 20 m elevation mask

```
min_value <- minValue(x)
maxValue(x)

x2 <- x

x2[x2 <= min_value] <- NA; x2[x2 >= (min_value + 20)] <- NA
plot(x2)

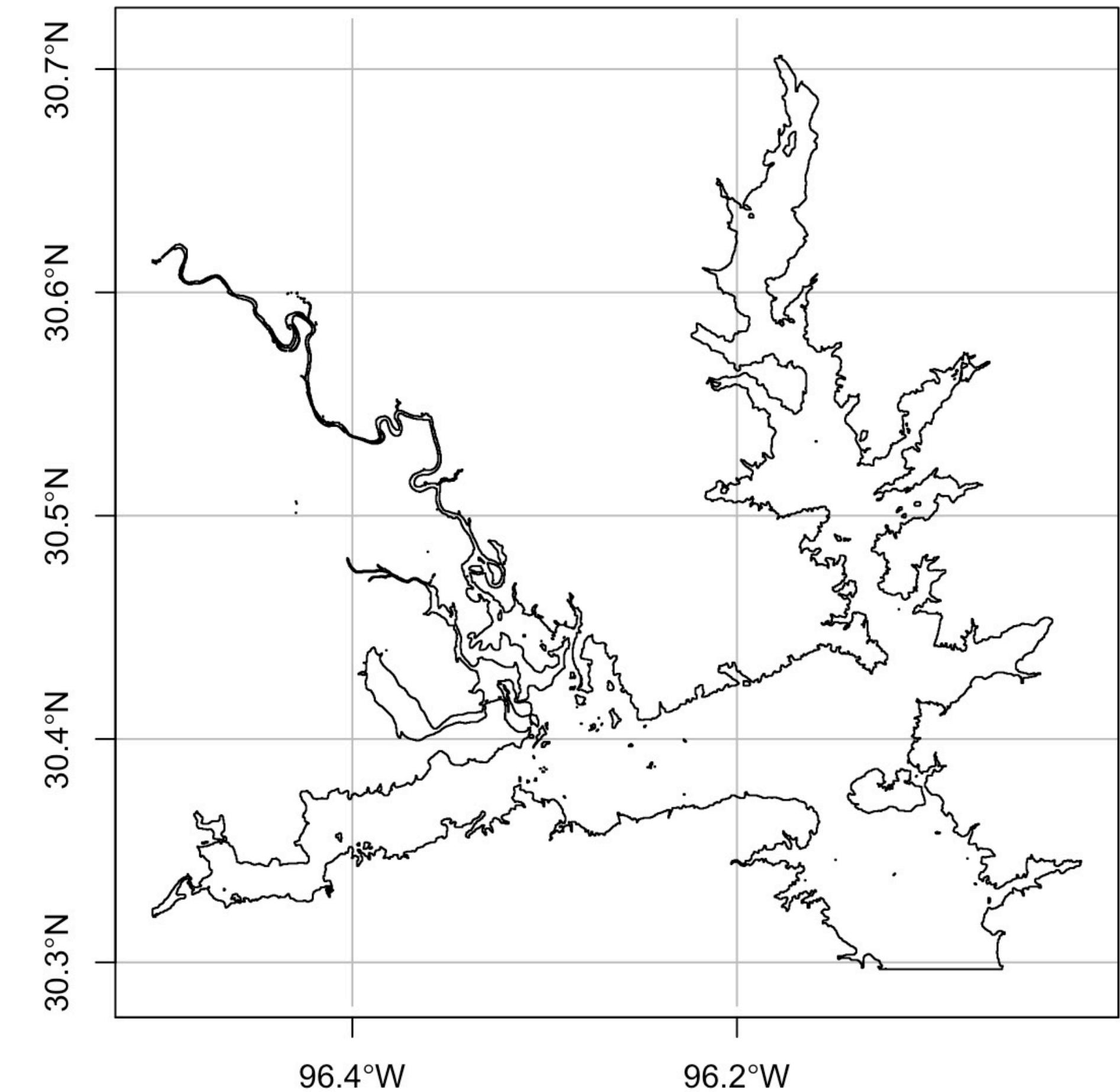
writeRaster(x, filename="elev_z12.tif", format="GTiff", overwrite=T)
```



# Elevation analysis

## Step 4: 20 m elevation polygonized

```
x2_poly <- spex::polygonize(x2) %>%  
  group_by() %>%  
  st_union() %>%  
  st_make_valid()  
  
st_area(x2_poly)  
  
plot(st_geometry(x2_poly), graticule=TRUE, axes=TRUE)
```



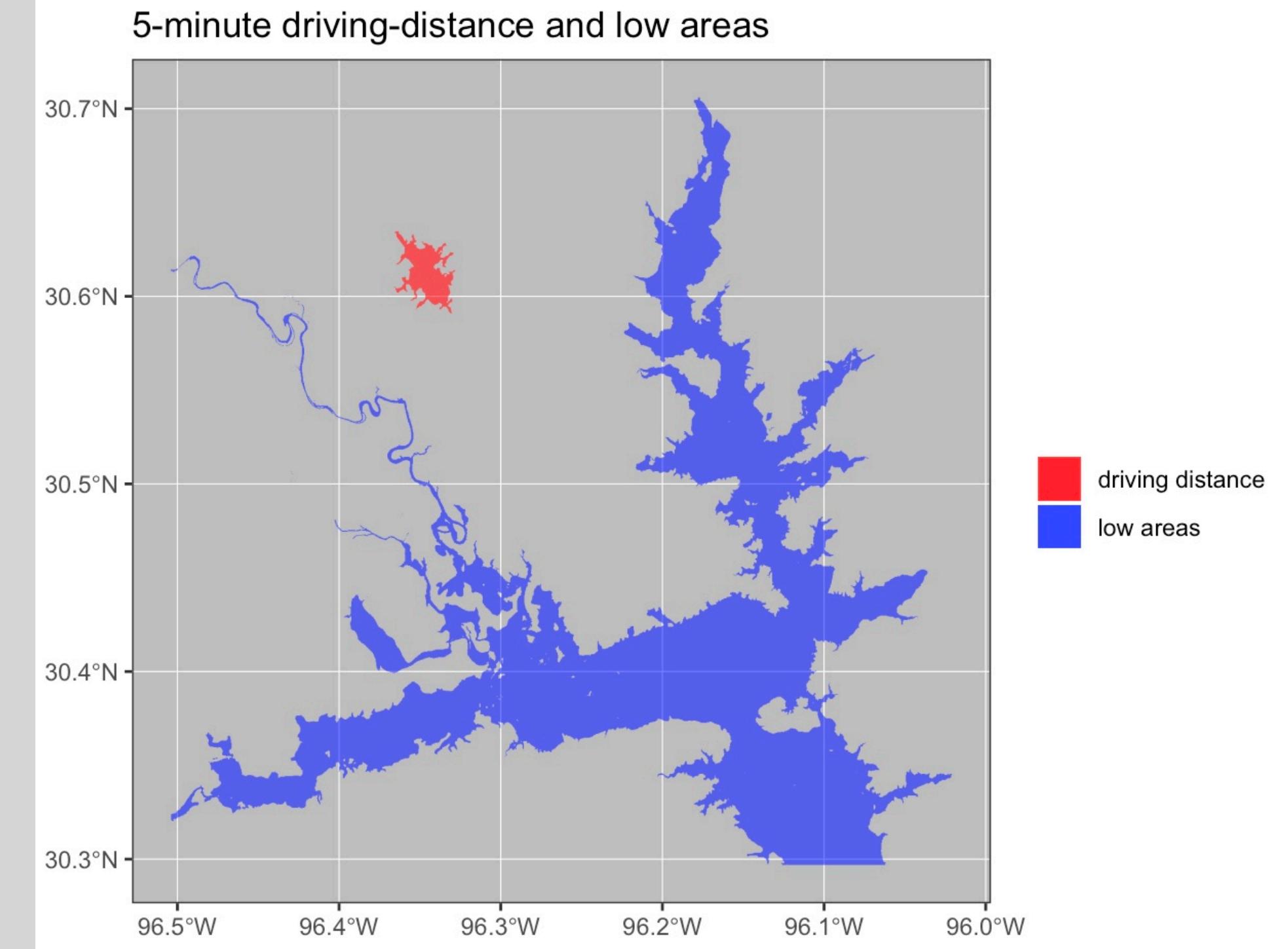
# Elevation analysis

## Step 5: Plot

```
bbox = c(left = as.numeric(st_bbox(x2_poly)$xmin-0.05),
        bottom = as.numeric(st_bbox(x2_poly)$ymin-0.025),
        right = as.numeric(st_bbox(x2_poly)$xmax+0.05),
        top = as.numeric(st_bbox(x2_poly)$ymax+0.025))

map <- get_stamenmap(bbox,
                      maptype = "toner-2011",
                      zoom = 12)

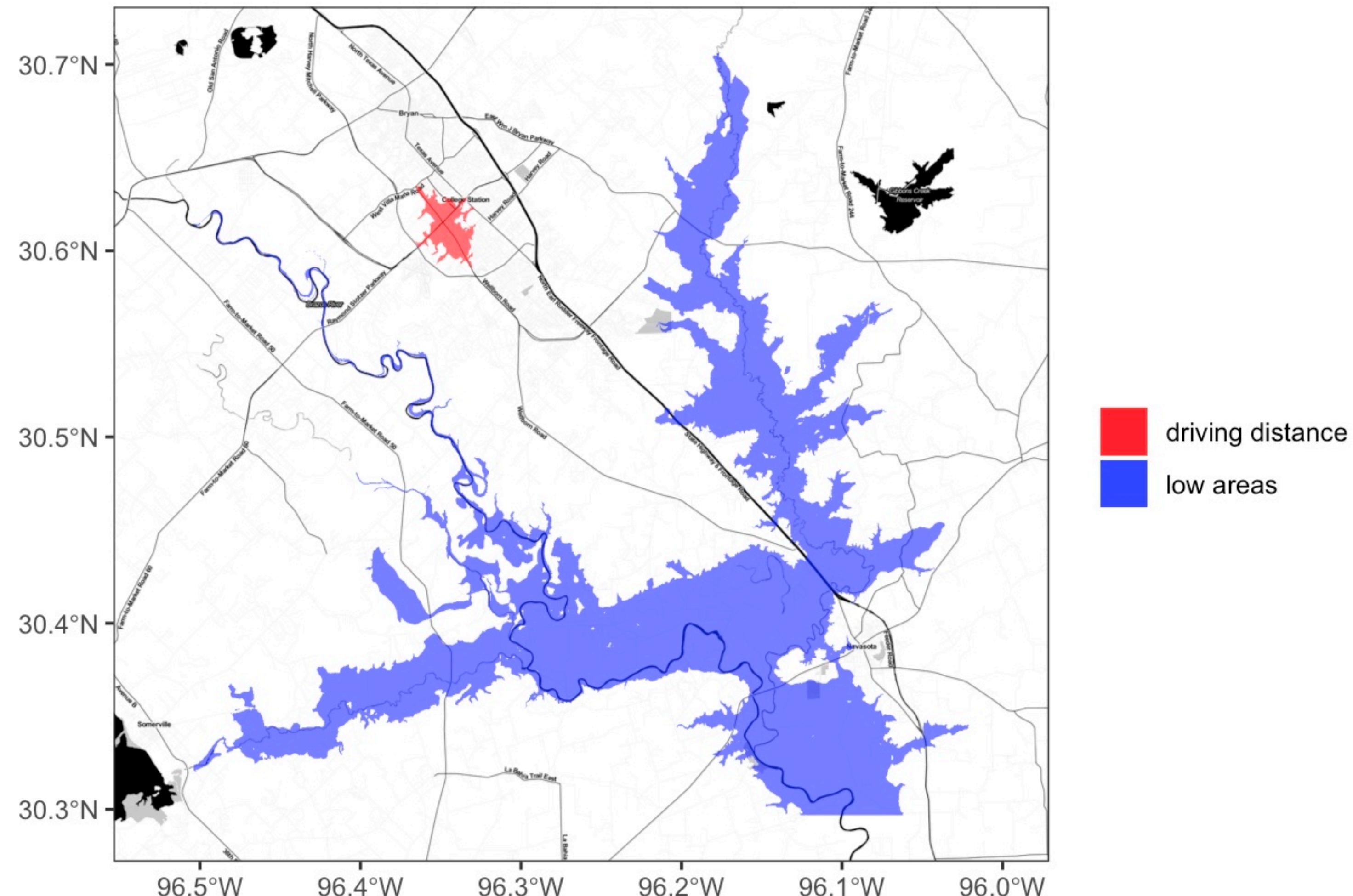
ggplot() +
  # ggmap(map) +
  geom_sf(data = x2_poly,
          aes(fill = "low areas"),
          color = "white", alpha = 0.6, linetype = "blank",
          show.legend = TRUE, inherit.aes = FALSE) +
  geom_sf(data = drive_5min,
          aes(fill = "driving distance"),
          color = "white", alpha = 0.6, linetype = "blank",
          show.legend = TRUE, inherit.aes = FALSE) +
  scale_fill_manual(values = c("driving distance" = "red",
                               "low areas" = "blue"),
                    breaks = c("driving distance", "low areas")) +
  labs(x="", y="", title="5-minute driving-distance and low areas", fill="") +
  theme(legend.justification="left",
        panel.background=element_rect(fill="grey", colour="grey",
                                      size=0.5, linetype="solid"),
        panel.grid.major=element_line(size=0.2, linetype="solid", colour="white"),
        panel.grid.minor=element_line(size=0.2, linetype="solid", colour="white"))
```



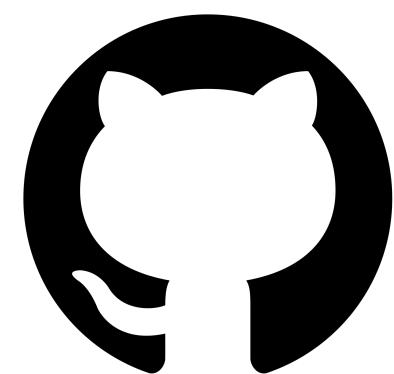
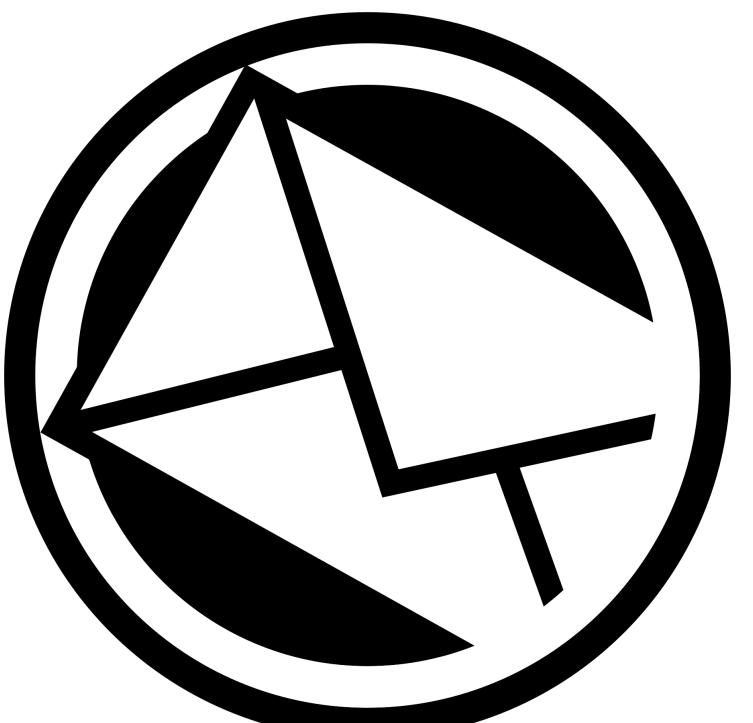
# Elevation analysis

## Step 6: 20 m elevation polygonized

5-minute driving-distance and low areas



# Thank you!



COLLEGE OF ARCHITECTURE



**Alexander Abuabara**

*PhD Student in Urban and Regional Science*

*Hazard Reduction and Recovery Center*

*Department of Landscape Architecture and Urban Planning*

Scoates Hall, Room 125

3137 TAMU

College Station, TX 77843-3137

Office phone: 979.845.7813

Email: abuabara@tamu.edu



# References

- Bivand, Roger. 2000. "Using the R Statistical Data Analysis Language on GRASS 5.0 GIS Database Files." *Computers & Geosciences* 26 (9): 1043–52.
- Bivand, Roger. 2001. "More on Spatial Data Analysis." *R News* 1 (3): 13–17.
- Bivand, Roger, and Colin Rundel. 2018. **Rgeos: Interface to Geometry Engine - Open Source** ('Geos'). <https://CRAN.R-project.org/package=rgeos>
- Bivand, Roger, Edzer J Pebesma, and Virgilio Gómez-Rubio. 2013. **Applied Spatial Data Analysis with R**. Vol. 74724817. Springer.
- Hijmans, Robert. (2019). Raster: Geographic Data Analysis and Modeling. R package version 2.8-19. URL: <https://CRAN.R-project.org/package=raster>
- Longley, Paul, Michael Goodchild, David Maguire, et al. 2015. **Geographic Information Science & Systems**. Fourth edition. Hoboken, NJ: Wiley. 477 pp. ISBN: 978-1-118-67695-0.
- Lovelace, Robin, Jakub Nowosad and Jannes Muenchow 2019. **Geocomputation with R**. The R Series. CRC Press.
- Muenchow, Jannes, Patrick Schratz and Alexander Brenning 2017. "RQGIS: Integrating R with QGIS for Statistical Geocomputing". In: *The R Journal* 9.2, pp. 409-428.
- Pebesma, Edzer. 2018. "**Simple Features for R: Standardized Support for Spatial Vector Data**". In: *The R Journal* 10.1, pp. 439-446. URL: <https://journal.r-project.org/archive/2018/RJ-2018-009/index.html>
- Pebesma, Edzer and Roger Bivand. 2005. "Classes and Methods for Spatial Data in R." *R News* 5 (2): 9–13.
- Pebesma, Edzer and Roger Bivand. 2018. Sp: Classes and Methods for Spatial Data. <https://CRAN.R-project.org/package=sp>
- Pebesma, Edzer, Daniel Nüst, and Roger Bivand. 2012. "The R Software Environment in Reproducible Geoscientific Research." *Eos, Transactions American Geophysical Union* 93 (16): 163–63. <https://doi.org/10.1029/2012EO160003>
- Pebesma, Edzer, Thomas Mailund, and James Hiebert. 2016. "Measurement Units in R." *The R Journal* 8 (2): 486–94.
- Pebesma, Edzer. 2018. "**Simple Features for R: Standardized Support for Spatial Vector Data**." *The R Journal*.