R Advanced Spatial Lessons

Ben Best

2017-09-24

Contents

Prerequisites

Lessons presented here are a continuation of the Geospatial workshop using R of Data Carpentry described more specifically for the Lawrence Berkeley National Lab: Sep 27-28, 2017.

This content is setup for now using bookdown (using the bookdown-demo) for expediency, and meant to eventually be folded into the Software Carpentry style.

Chapter 1

Tidy Spatial Analysis

Resources:

- Tidy spatial data in R: using dplyr, tidyr, and ggplot2 with sf

1.1 Overview

Questions - How to elegantly conduct complex spatial analysis?

Objectives - Use the `%>%` operator (aka "then" or "pipe") to pass output from one function into input of the next. - Calculate metrics on spatial attributes. - Aggregate spatial data with metrics.

1.2 Things You'll Need to Complete this Tutorial

R Skill Level: Intermediate - you've got basics of R down.

You'll need …

We will use the `sf` package for vector data along with the `dplyr` package for calculating and manipulating attribute data.

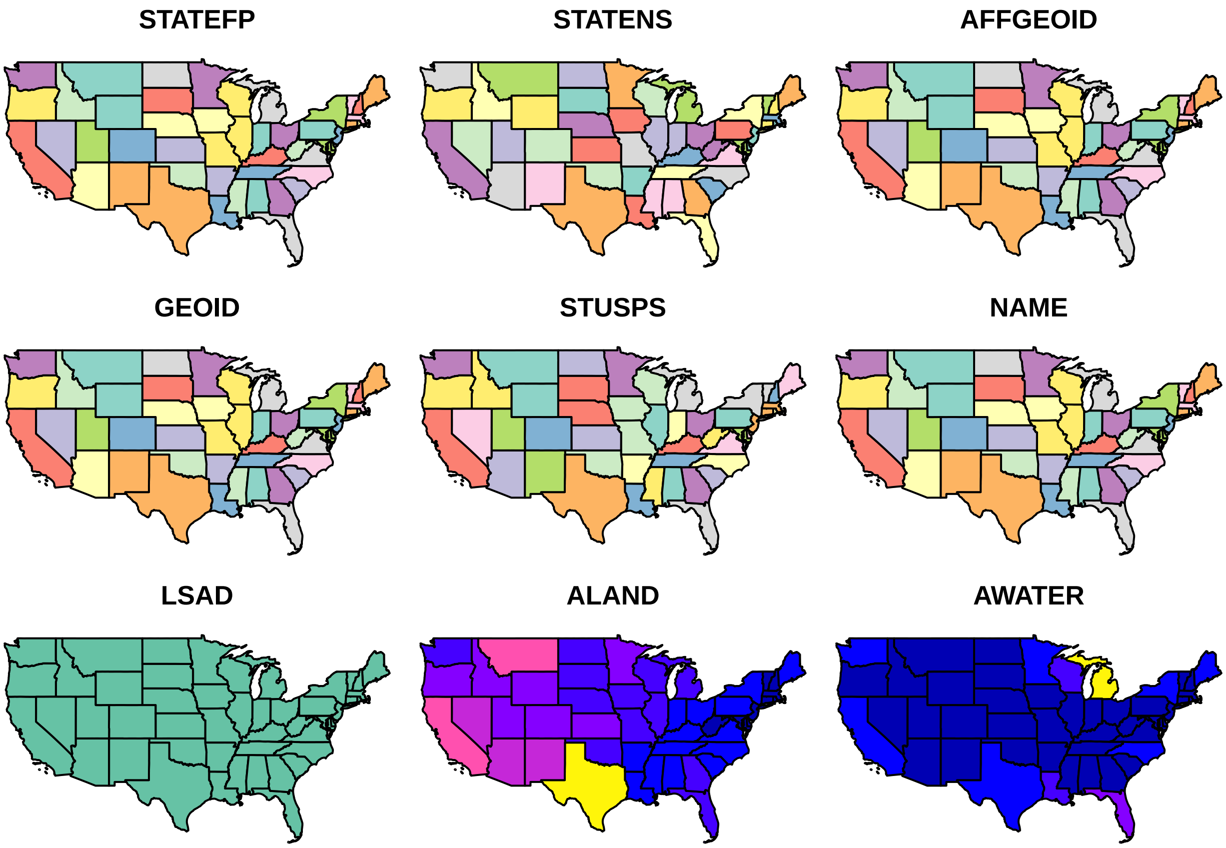
```
# load packages
library(tidyverse)  # dplyr, tidyr, ggplot2
library(sf)         # vector reading & analysis

# set working directory to data folder
# setwd("pathToDirHere")
```

1.3 US States: Read and Plot

Similar to Lesson 9: Handling Spatial Projection & CRS in R, we'll start by reading in a polygon shapefile using the `sf` package. Then use the default `plot()` function to see what it looks like.

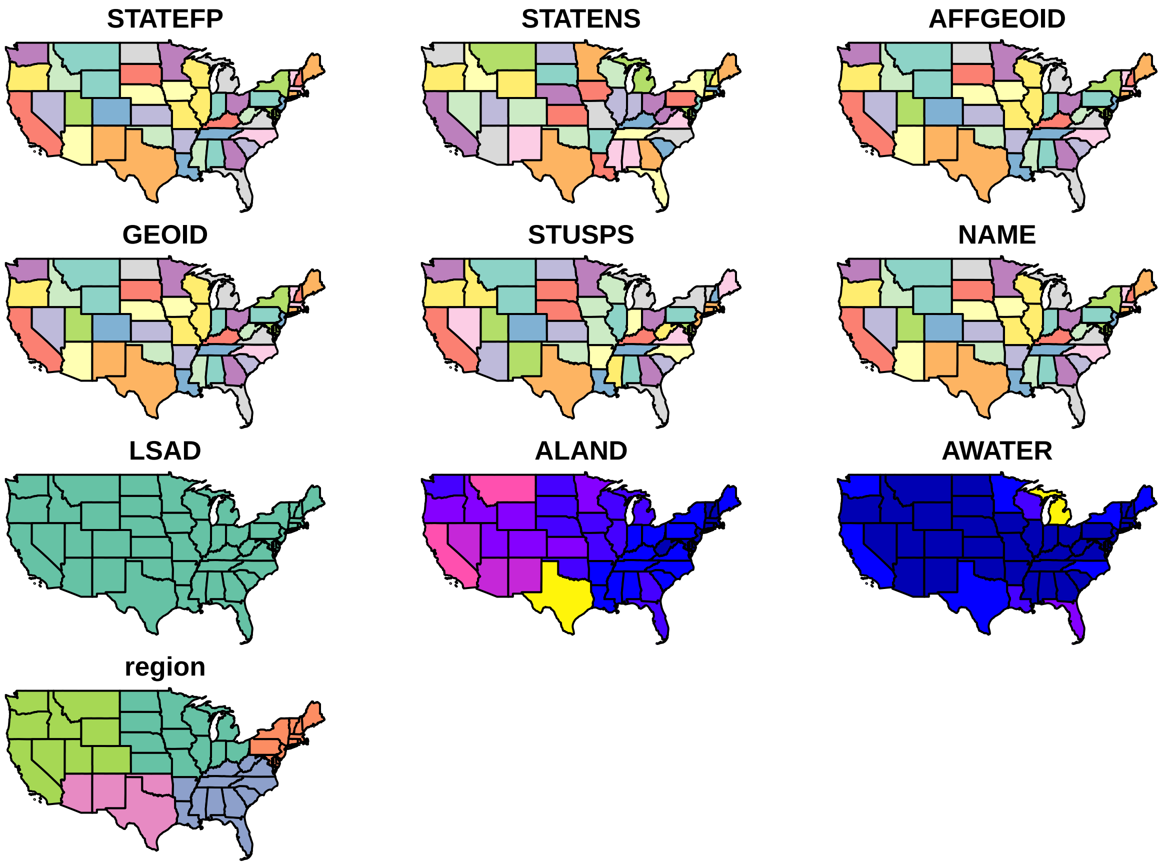
```
# read in states
states <- read_sf("data/NEON-DS-Site-Layout-Files/US-Boundary-Layers/US-State-Boundaries-Census-2014.shp
plot(states)
```

```
## Warning: plotting the first 9 out of 10 attributes; use max.plot = 10 to
## plot all
```



You'll notice that the default plot on `sf` objects outputs colorized values of the first 9 of 10 columns. Use the suggestion from the warning to plot the 10th column.

```
# plot 10th column
plot(states, max.plot = 10)
```




```r
# show columns of the data frame
names(states)
```

```
##  [1] "STATEFP"  "STATENS"  "AFFGEOID" "GEOID"    "STUSPS"   "NAME"
##  [7] "LSAD"     "ALAND"    "AWATER"   "region"   "geometry"
```

```r
# look at table
glimpse(states)
```

```
## Observations: 58
## Variables: 11
## $ STATEFP  <chr> "06", "11", "12", "13", "16", "17", "19", "21", "22",...
## $ STATENS  <chr> "01779778", "01702382", "00294478", "01705317", "0177...
## $ AFFGEOID <chr> "0400000US06", "0400000US11", "0400000US12", "0400000...
## $ GEOID    <chr> "06", "11", "12", "13", "16", "17", "19", "21", "22",...
## $ STUSPS   <chr> "CA", "DC", "FL", "GA", "ID", "IL", "IA", "KY", "LA",...
## $ NAME     <chr> "California", "District of Columbia", "Florida", "Geo...
## $ LSAD     <chr> "00", "00", "00", "00", "00", "00", "00", "00", "00",...
## $ ALAND    <dbl> 403483823181, 158350578, 138903200855, 148963503399, ...
## $ AWATER   <dbl> 20483271881, 18633500, 31407883551, 4947080103, 23977...
## $ region   <chr> "West", "Northeast", "Southeast", "Southeast", "West"...
## $ geometry <simple_feature> MULTIPOLYGONZ(((-118.593969..., MULTIPOLYG...
```

```r
# convert to tibble for nicer printing
as_tibble(states)
```

```
## Simple feature collection with 58 features and 10 fields
## geometry type:  MULTIPOLYGON
```

```
## dimension:      XYZ
## bbox:           xmin: -124.7258 ymin: 24.49813 xmax: -66.9499 ymax: 49.38436
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## # A tibble: 58 x 11
##    STATEFP  STATENS    AFFGEOID GEOID STUSPS                 NAME  LSAD
##      <chr>    <chr>       <chr> <chr>  <chr>                <chr> <chr>
##  1      06 01779778 0400000US06    06     CA           California    00
##  2      11 01702382 0400000US11    11     DC District of Columbia    00
##  3      12 00294478 0400000US12    12     FL              Florida    00
##  4      13 01705317 0400000US13    13     GA              Georgia    00
##  5      16 01779783 0400000US16    16     ID                Idaho    00
##  6      17 01779784 0400000US17    17     IL             Illinois    00
##  7      19 01779785 0400000US19    19     IA                 Iowa    00
##  8      21 01779786 0400000US21    21     KY             Kentucky    00
##  9      22 01629543 0400000US22    22     LA            Louisiana    00
## 10      24 01714934 0400000US24    24     MD             Maryland    00
## # ... with 48 more rows, and 4 more variables: ALAND <dbl>, AWATER <dbl>,
## #   region <chr>, geometry <simple_feature>
```

```
names(states)
```

```
##  [1] "STATEFP"  "STATENS"  "AFFGEOID" "GEOID"    "STUSPS"   "NAME"
##  [7] "LSAD"     "ALAND"    "AWATER"   "region"   "geometry"
```

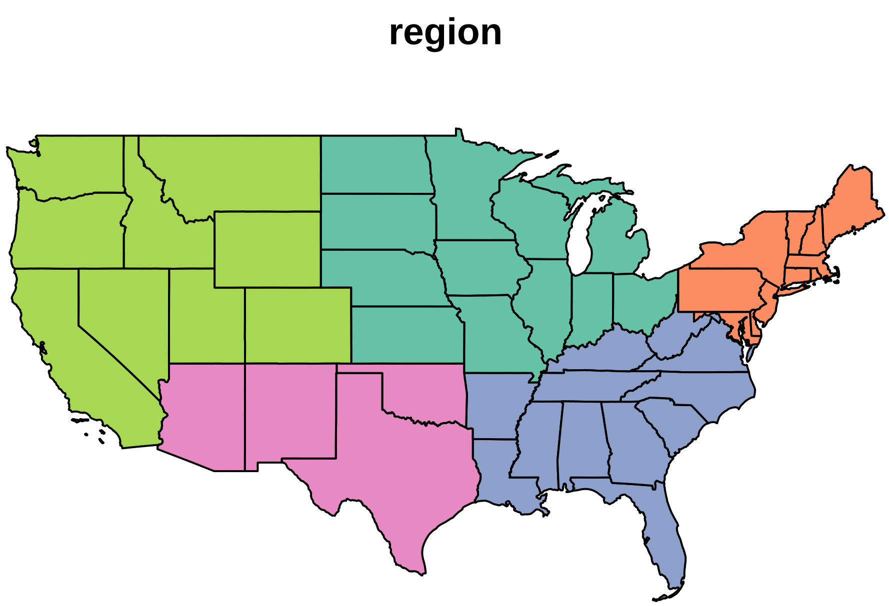
```
# inspect the class(es) of the states object
class(states)
```

```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

The class of the `states` object is both a simple feature (`sf`) as well as a data frame, which means the many useful functions available to a data frame (or "tibble") can be applied.

To plot the column of interest, feed the "slice" of that column to the `plot()` function.

```
plot(states['region'])
```



Question: To motivate the spatial analysis for the rest of this lesson, let's answer this question: "***Show***

which regions have the highest ratio of water to land?"

1.4 Challenge: What analytical steps are required to answer the question?

Outline a sequence of analytical steps needed to arrive at the answer.

1.4.1 Answers

1. **Sum** the area of water and land per region.
2. **Divide** the area of water by the area of land per region to arrive at density of water.
3. Show **table** of regions sorted by density of water.
4. Show **map** of regions by density of water with a color ramp and legend.

1.5 US States: Analyze Attributes

```
regions = states %>%
  group_by(region) %>%
  summarize(
    water = sum(AWATER),
    land  = sum(ALAND)) %>%
  mutate(
    water_land = water / land)

# object
regions
```

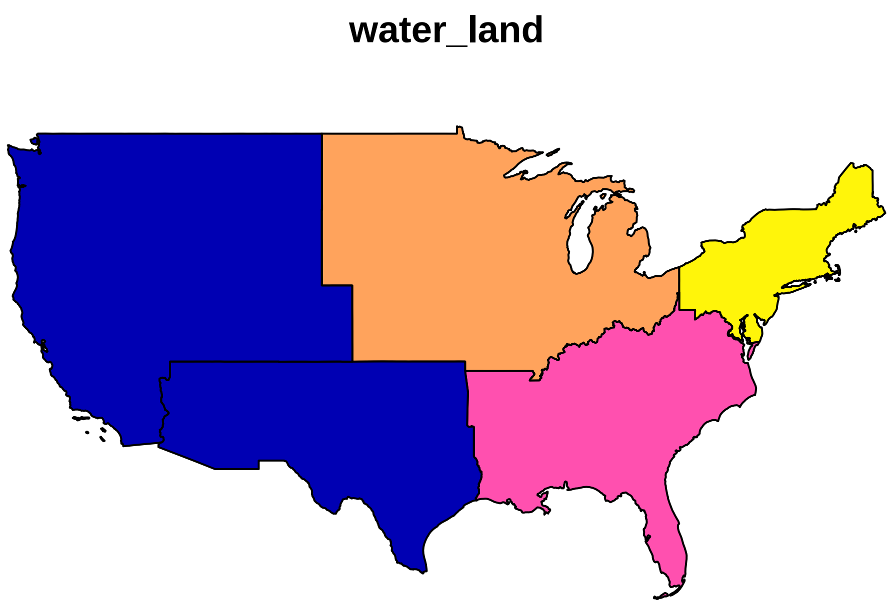
```
## Simple feature collection with 5 features and 4 fields
## geometry type:  GEOMETRY
## dimension:      XYZ
## bbox:           xmin: -124.7258 ymin: 24.49813 xmax: -66.9499 ymax: 49.38436
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## # A tibble: 5 x 5
##      region        water         land water_land          geometry
##       <chr>        <dbl>        <dbl>      <dbl>  <simple_feature>
## 1   Midwest 184383393833 1.943869e+12 0.09485380 <MULTIPOLYGON...>
## 2 Northeast 108922434345 8.690661e+11 0.12533273 <MULTIPOLYGON...>
## 3 Southeast 103876652998 1.364632e+12 0.07612063 <MULTIPOLYGON...>
## 4 Southwest  24217682268 1.462632e+12 0.01655761 <POLYGONZ((-9...>
## 5      West  57568049509 2.432336e+12 0.02366780 <MULTIPOLYGON...>
```

```
# table
regions %>%
  st_set_geometry(NULL) %>%
  arrange(desc(water_land))
```

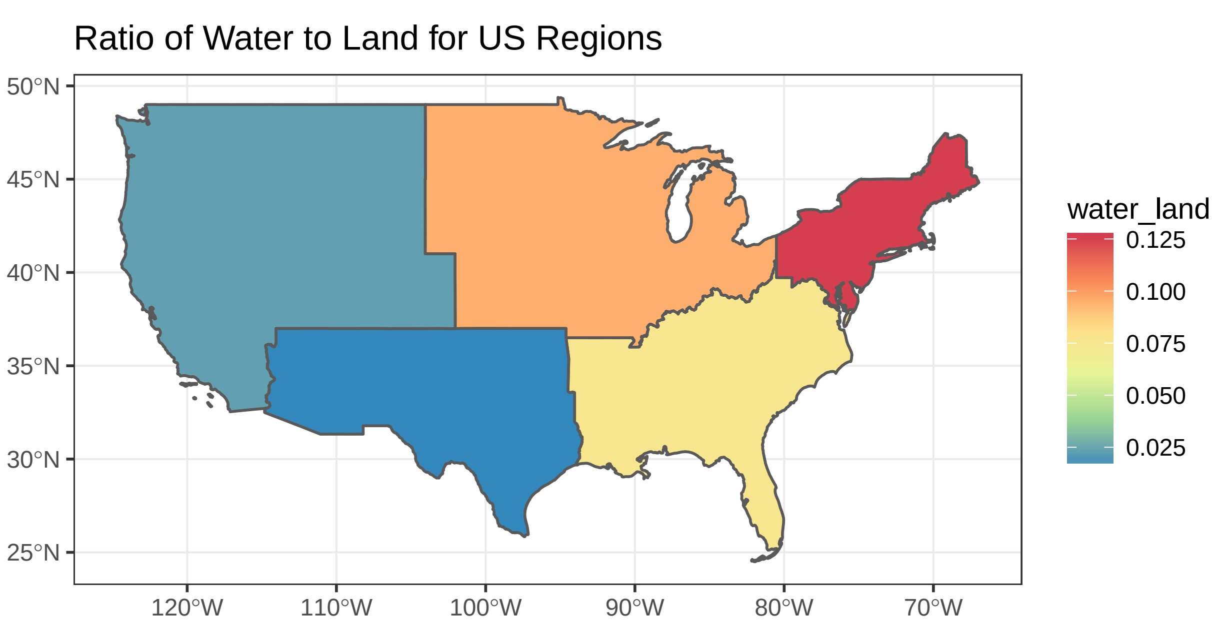
```
## # A tibble: 5 x 4
##      region        water         land water_land
##       <chr>        <dbl>        <dbl>      <dbl>
```

```
## 1 Northeast 108922434345 8.690661e+11 0.12533273
## 2   Midwest 184383393833 1.943869e+12 0.09485380
## 3 Southeast 103876652998 1.364632e+12 0.07612063
## 4      West  57568049509 2.432336e+12 0.02366780
## 5 Southwest  24217682268 1.462632e+12 0.01655761
```

```
# plot, default
plot(regions['water_land'])
```




```
# plot, ggplot
ggplot(regions) +
  geom_sf(aes(fill = water_land)) +
  scale_fill_distiller("water_land", palette = "Spectral") +
  theme_bw() +
  ggtitle("Ratio of Water to Land for US Regions")
```



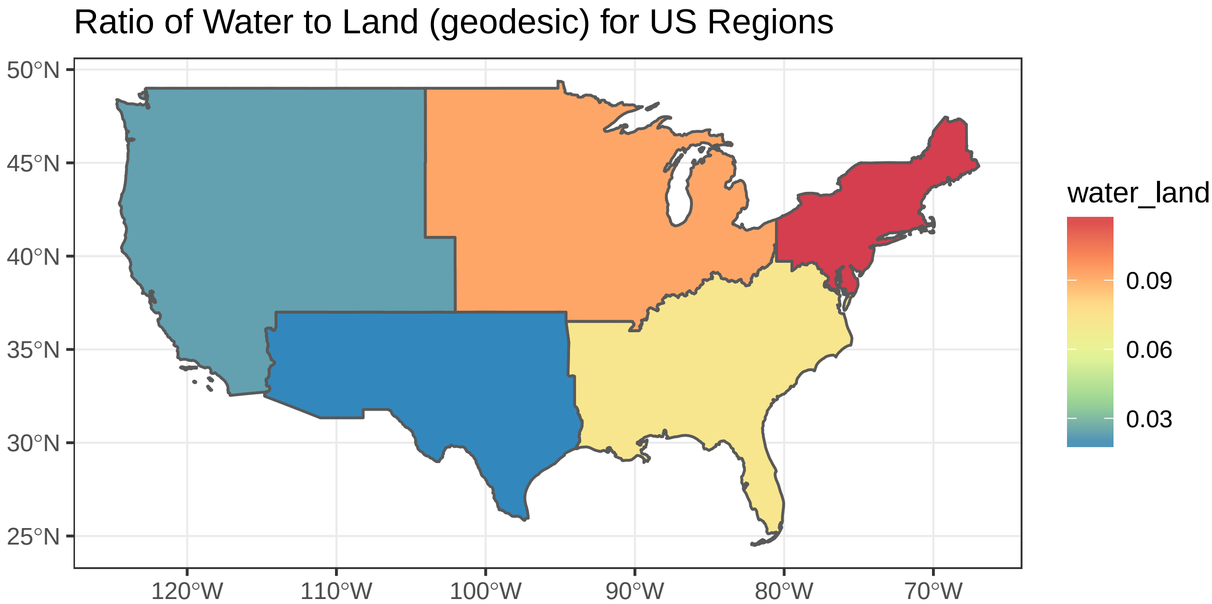
1.6 US States: Recalculate Area

Because in geographic coordinates, need to either transform to projection and calculate area, or use geodesic calculations.

```
library(geosphere)
library(units)

regions = states %>%
  mutate(
    water_m2 = AWATER %>% set_units(m^2),
    land_m2  = geometry %>% st_zm() %>% st_area()) %>%
  group_by(region) %>%
  summarize(
    water_m2 = sum(water_m2),
    land_m2  = sum(land_m2)) %>%
  mutate(
    water_land = water_m2 / land_m2)

# table
regions %>%
  st_set_geometry(NULL) %>%
  arrange(desc(water_land))
```

```
## # A tibble: 5 x 4
##      region        water_m2          land_m2   water_land
##       <chr>         <units>          <units>      <units>
## 1 Northeast 108922434345 m^2 9.117041e+11 m^2 0.11947126 1
## 2   Midwest 184383393833 m^2 1.987268e+12 m^2 0.09278233 1
## 3 Southeast 103876652998 m^2 1.427079e+12 m^2 0.07278971 1
## 4      West  57568049509 m^2 2.467170e+12 m^2 0.02333363 1
## 5 Southwest  24217682268 m^2 1.483765e+12 m^2 0.01632178 1
```

```
# plot, ggplot
ggplot(regions) +
  geom_sf(aes(fill = as.numeric(water_land))) +
  scale_fill_distiller("water_land", palette = "Spectral") +
  theme_bw() +
  ggtitle("Ratio of Water to Land (geodesic) for US Regions ")
```



1.7 Pipe Operator

- Help > Keyboard Shortcuts Help.

1.8 Challenge: Project States and Calculate Area

Use `st_transform()` USA Contiguous Albers Equal Area Conic: ESRI Projection – Spatial Reference.

1.8.1 Answers

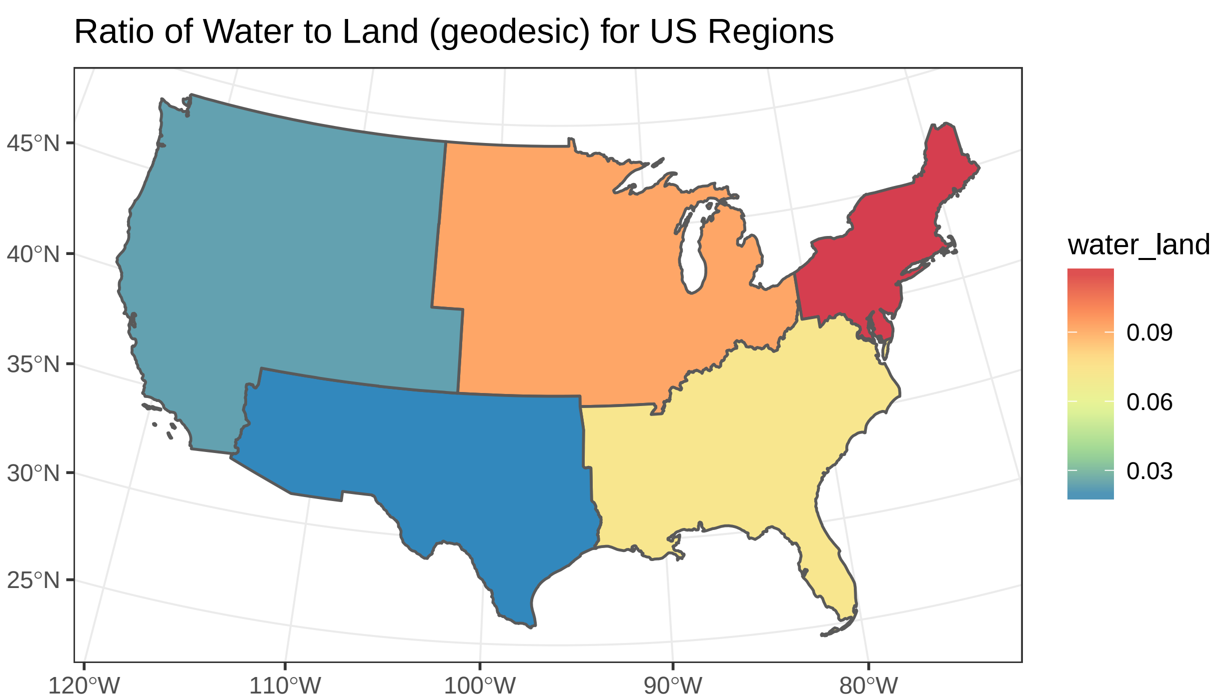
- ESRI:102003

```
library(geosphere)
library(units)

# Proj4 of http://spatialreference.org/ref/esri/usa-contiguous-albers-equal-area-conic/
crs_usa = '+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=37.5 +lon_0=-96 +x_0=0 +y_0=0 +ellps=GRS80 +datum=N

regions = states %>%
  st_transform(crs_usa) %>%
  mutate(
    water_m2 = AWATER %>% set_units(m^2),
    land_m2  = geometry %>% st_zm() %>% st_area()) %>%
  group_by(region) %>%
  summarize(
    water_m2 = sum(water_m2),
    land_m2  = sum(land_m2)) %>%
  mutate(
    water_land = water_m2 / land_m2)

# table
```

```
regions %>%
  st_set_geometry(NULL) %>%
  arrange(desc(water_land))
```

```
## # A tibble: 5 x 4
##      region         water_m2          land_m2   water_land
##       <chr>          <units>          <units>      <units>
## 1 Northeast 108922434345 m^2 9.117031e+11 m^2 0.11947138 1
## 2   Midwest 184383393833 m^2 1.987266e+12 m^2 0.09278246 1
## 3 Southeast 103876652998 m^2 1.427078e+12 m^2 0.07278973 1
## 4      West  57568049509 m^2 2.467167e+12 m^2 0.02333367 1
## 5 Southwest  24217682268 m^2 1.483758e+12 m^2 0.01632185 1
```

```
# plot, ggplot
ggplot(regions) +
  geom_sf(aes(fill = as.numeric(water_land))) +
  scale_fill_distiller("water_land", palette = "Spectral") +
  theme_bw() +
  ggtitle("Ratio of Water to Land (geodesic) for US Regions ")
```



1.9 Key Points

- Area can be calculated a variety of ways. Geodesic is preferred if starting with geographic coordinates (vs projected).

1.10 Issues

- `sf::st_is_valid()`

Chapter 2

Interactive Maps

Resources:

- Visualization in R
- Leaflet for R - Introduction
- mapedit - interactively edit spatial data in R
- Interactive Viewing of Spatial Objects in R • mapview

2.1 Overview

Questions - How do you generate interactive plots of spatial data?

2.2 Objectives

2.3 Things You'll Need to Complete this Tutorial

R Skill Level: Intermediate - you've got basics of R down.

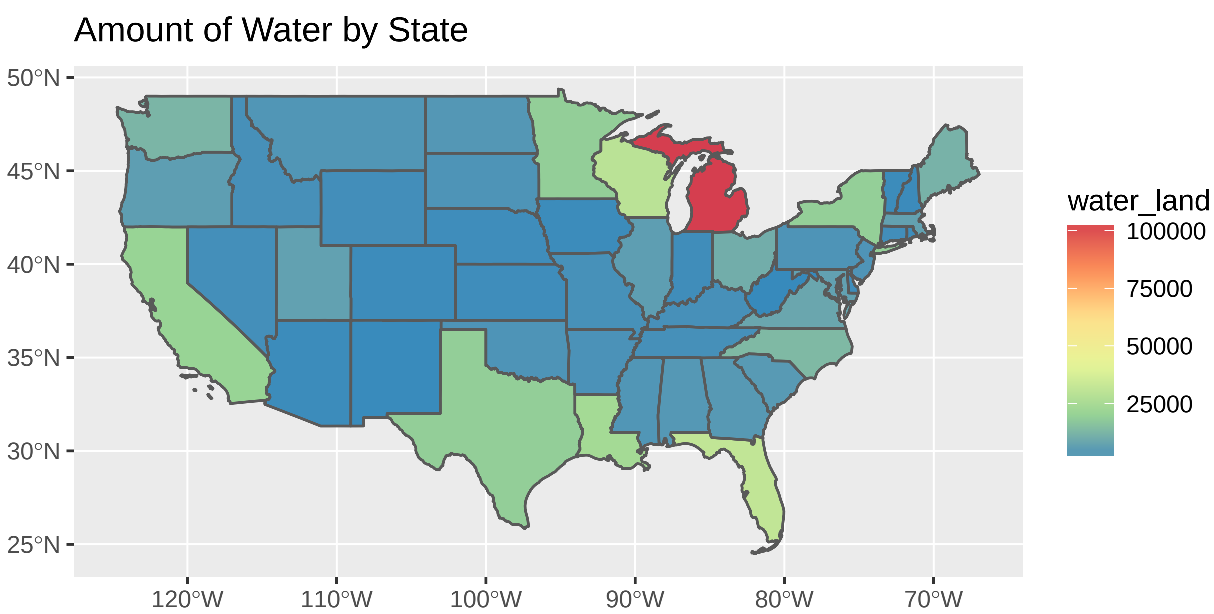
You'll need …

We will continue to use the `sf` and `raster` packages and introduce the `leaflet` package in this tutorial.

```
# load packages
library(tidyverse)  # loads dplyr, tidyr, ggplot2 packages
library(sf)         # simple features package - vector
library(raster)     # raster
library(leaflet)    # interactive

# set working directory to data folder
# setwd("pathToDirHere")
```

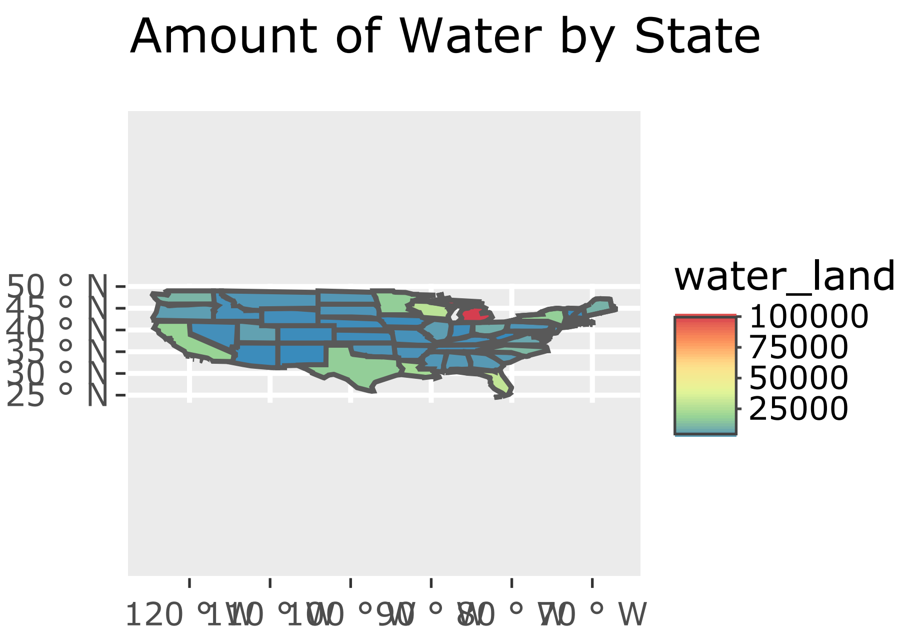
2.4 States: ggplot2

```r
# read in states
states <- read_sf("data/NEON-DS-Site-Layout-Files/US-Boundary-Layers/US-State-Boundaries-Census-2014.shp
  st_zm() %>%
  mutate(
    water_km2 = (AWATER / (1000*1000)) %>% round(1))

# plot, ggplot
g = ggplot(states) +
  geom_sf(aes(fill = water_km2)) +
  scale_fill_distiller("water_land", palette = "Spectral") +
  ggtitle("Amount of Water by State")
g
```



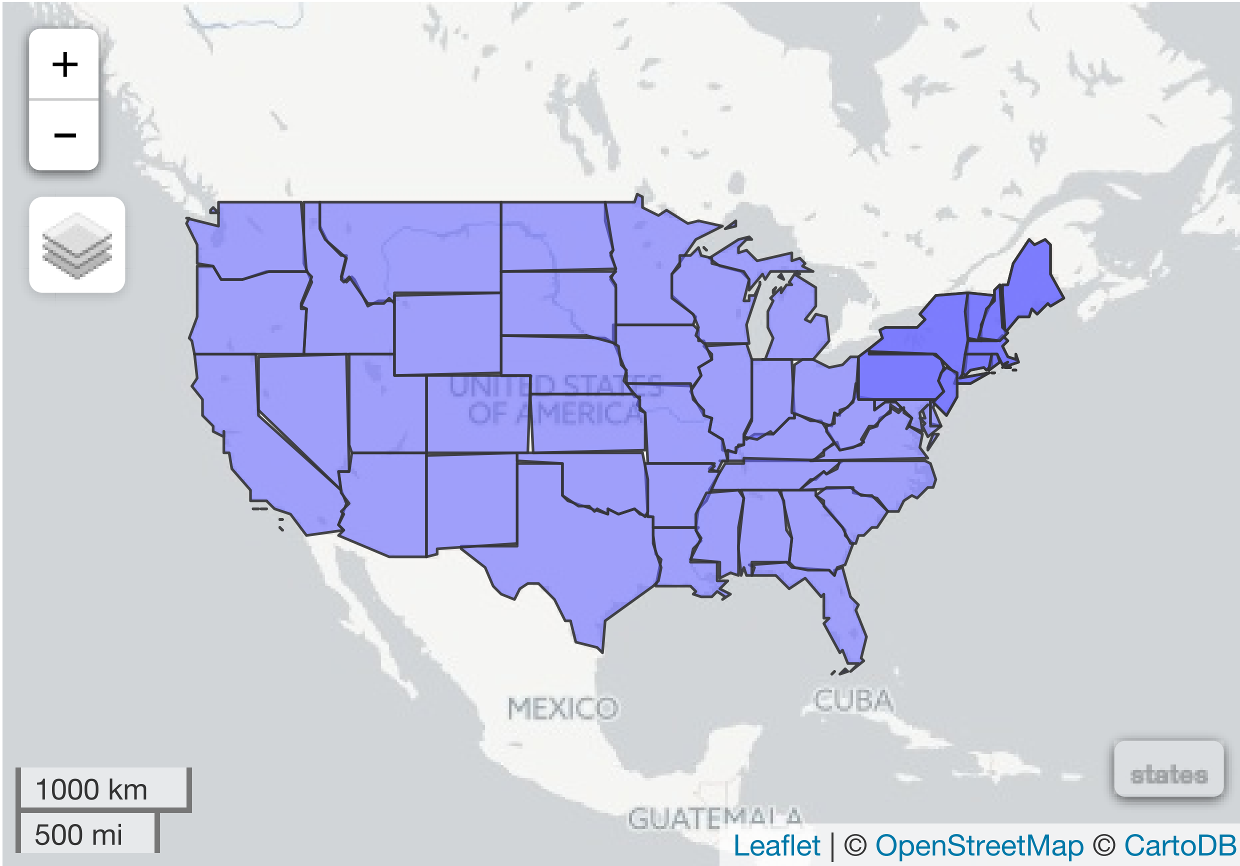
2.5 States: plotly

```r
library(plotly)

ggplotly(g)
```

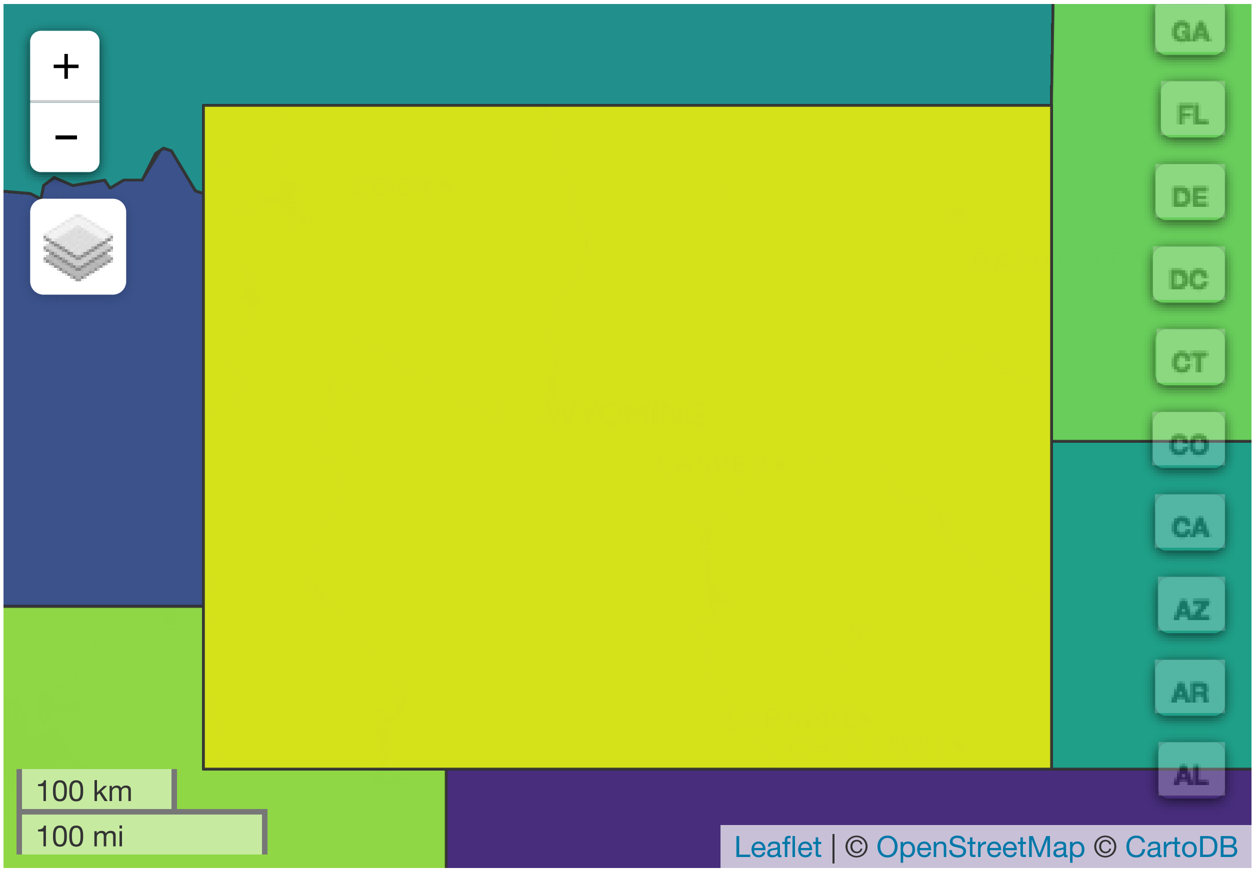


2.6 States: mapview

```
library(mapview)

mapview(states)
```

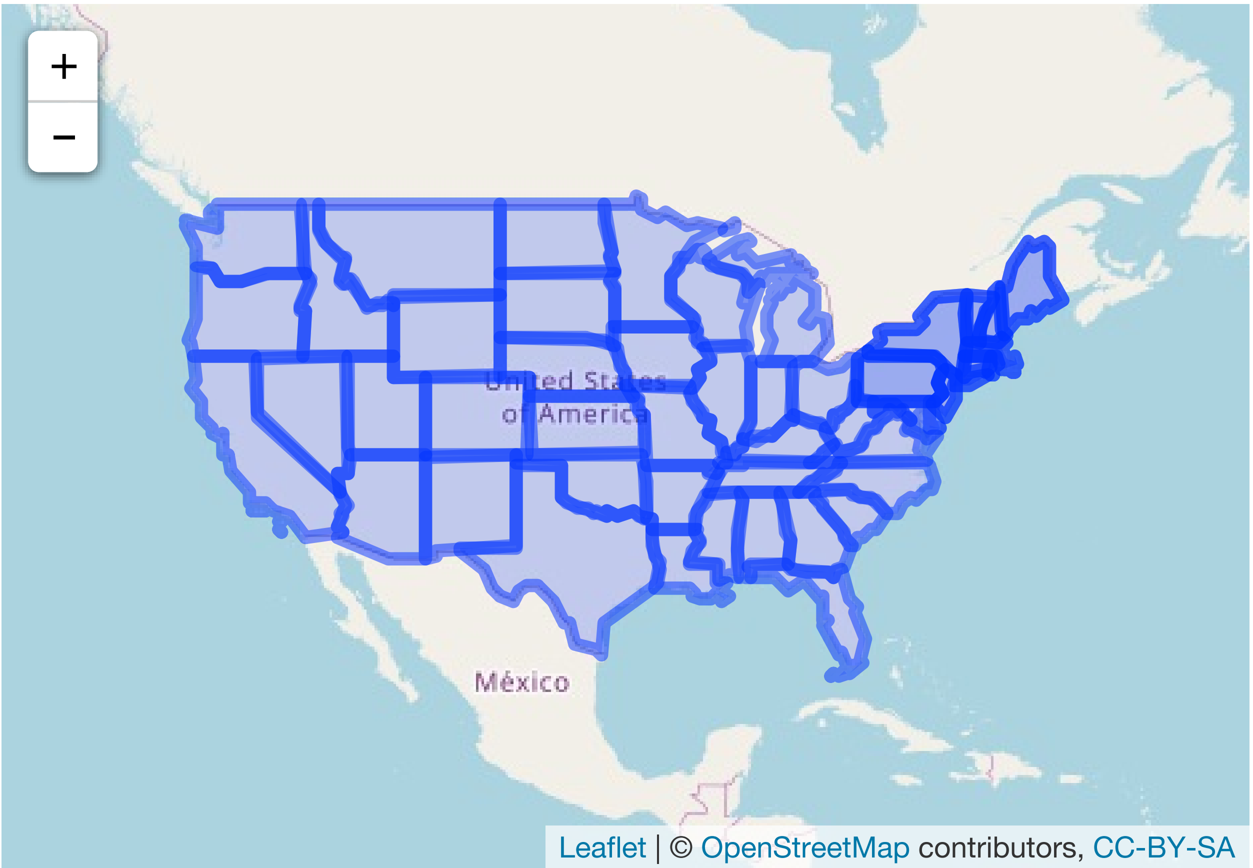



```
mapview(states, zcol='water_km2', burst='STUSPS') # , burst = TRUE, hide = TRUE) # , burst=T)
```



2.7 States: leaflet

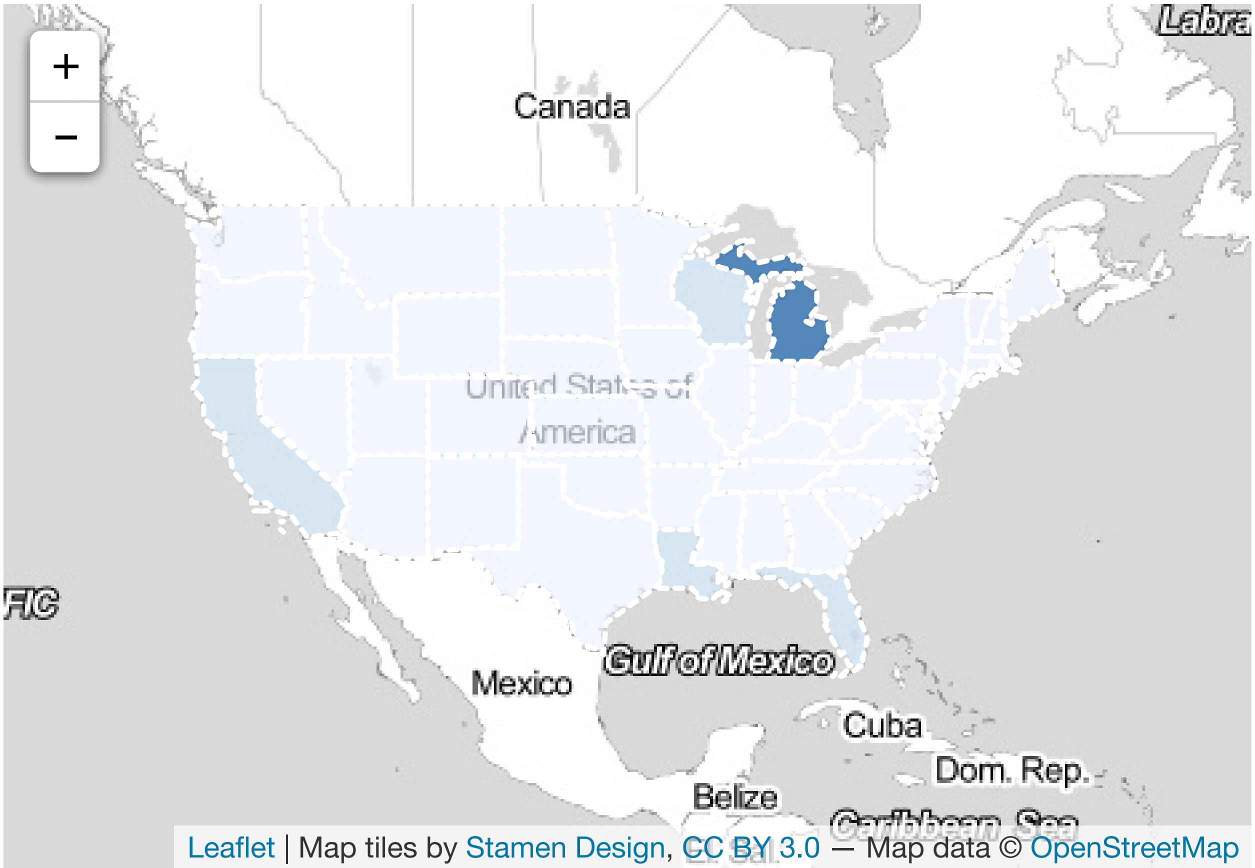
```
library(leaflet)

leaflet(states) %>%
  addTiles() %>%
  addPolygons()
```



2.7.1 Choropleth

- Leaflet for R - Choropleths

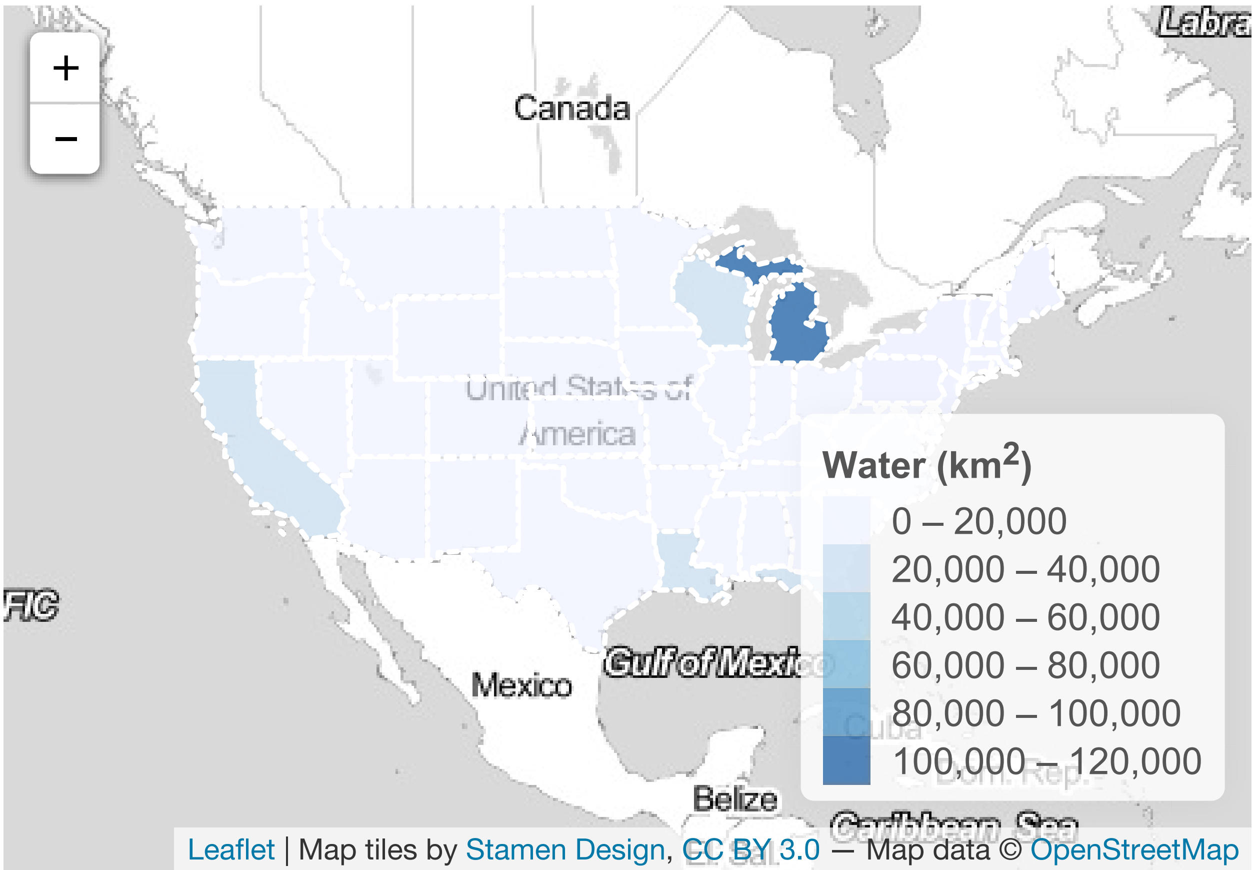
```
pal <- colorBin("Blues", domain = states$water_km2, bins = 7)


leaflet(states) %>%
  addProviderTiles("Stamen.TonerLite") %>%
  addPolygons(
    # fill
    fillColor    = ~pal(water_km2),
    fillOpacity = 0.7,
    # line
    dashArray    = "3",
    weight       = 2,
    color        = "white",
    opacity      = 1,
    # interaction
    highlight = highlightOptions(
      weight = 5,
      color = "#666",
      dashArray = "",
      fillOpacity = 0.7,
      bringToFront = TRUE))
```



2.7.2 Popups and Legend

```
library(htmltools)
library(scales)

labels <- sprintf(
  "<strong>%s</strong><br/> water: %s km<sup>2</sup>",
  states$NAME, comma(states$water_km2)) %>%
  lapply(HTML)

leaflet(states) %>%
  addProviderTiles("Stamen.TonerLite") %>%
  addPolygons(
    # fill
    fillColor   = ~pal(water_km2),
    fillOpacity = 0.7,
    # line
    dashArray   = "3",
    weight      = 2,
    color       = "white",
    opacity     = 1,
    # interaction
    highlight = highlightOptions(
      weight = 5,
```

```
      color = "#666",
      dashArray = "",
      fillOpacity = 0.7,
      bringToFront = TRUE),
  label = labels,
  labelOptions = labelOptions(
    style = list("font-weight" = "normal", padding = "3px 8px"),
    textsize = "15px",
    direction = "auto")) %>%
  addLegend(
    pal = pal, values = ~water_km2, opacity = 0.7, title = HTML("Water (km<sup>2</sup>)"),
    position = "bottomright")
```



2.8 Pipe Operator

- Help > Keyboard Shortcuts Help.

2.9 Challenge: Project States and Calculate Area

Use `st_transform()` USA Contiguous Albers Equal Area Conic: ESRI Projection – Spatial Reference.

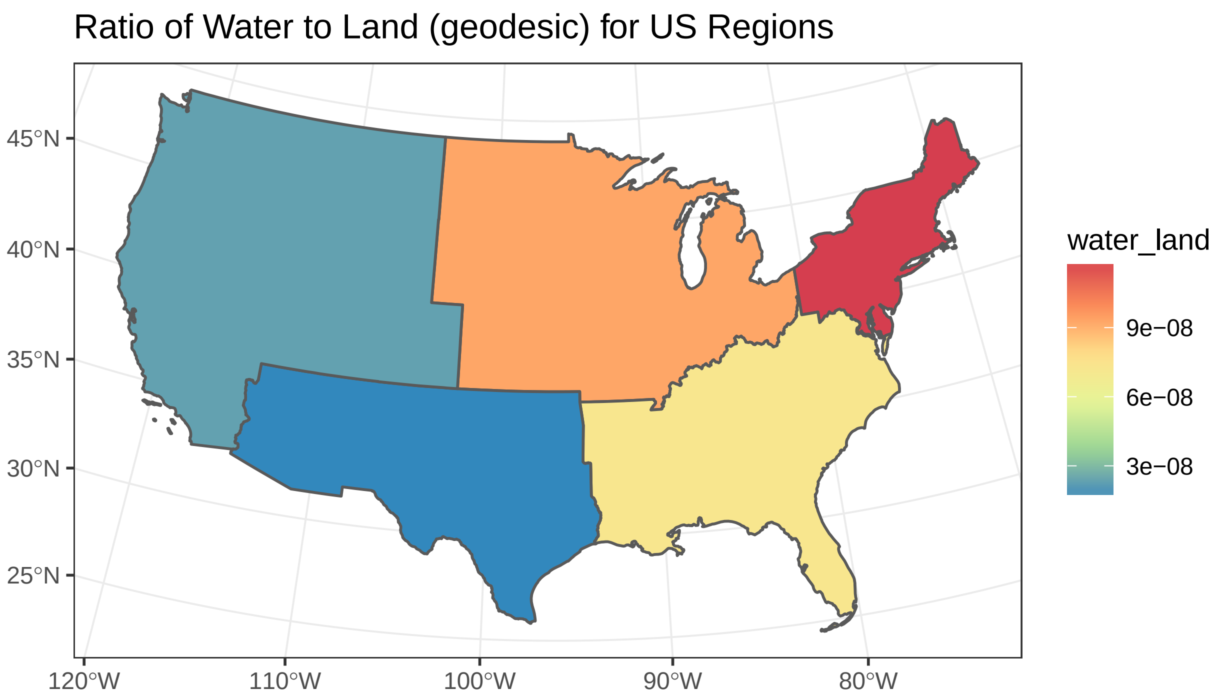
2.9.1 Answers

- ESRI:102003

```
library(geosphere)
library(units)

# Proj4 of http://spatialreference.org/ref/esri/usa-contiguous-albers-equal-area-conic/
crs_usa = '+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=37.5 +lon_0=-96 +x_0=0 +y_0=0 +ellps=GRS80 +datum=NA

regions = states %>%
  st_transform(crs_usa) %>%
  mutate(
    water_m2 = water_km2 %>% set_units(m^2),
    land_m2  = geometry %>% st_zm() %>% st_area()) %>%
  group_by(region) %>%
  summarize(
    water_m2 = sum(water_m2),
    land_m2  = sum(land_m2)) %>%
  mutate(
    water_land = water_m2 / land_m2)

# table
regions %>%
  st_set_geometry(NULL) %>%
  arrange(desc(water_land))
```

```
## # A tibble: 5 x 4
##      region     water_m2          land_m2     water_land
##       <chr>      <units>          <units>        <units>
## 1 Northeast 108922.9 m^2 9.117031e+11 m^2 1.194719e-07 1
## 2   Midwest 184383.2 m^2 1.987266e+12 m^2 9.278237e-08 1
## 3 Southeast 103876.6 m^2 1.427078e+12 m^2 7.278970e-08 1
## 4      West  57568.0 m^2 2.467167e+12 m^2 2.333365e-08 1
## 5 Southwest  24217.7 m^2 1.483758e+12 m^2 1.632186e-08 1
```

```
# plot, ggplot
ggplot(regions) +
  geom_sf(aes(fill = as.numeric(water_land))) +
  scale_fill_distiller("water_land", palette = "Spectral") +
  theme_bw() +
  ggtitle("Ratio of Water to Land (geodesic) for US Regions ")
```



2.10 Key Points

- Area can be calculated a variety of ways. Geodesic is preferred if starting with geographic coordinates (vs projected).

2.11 Issues

- `sf::st_is_valid()`