

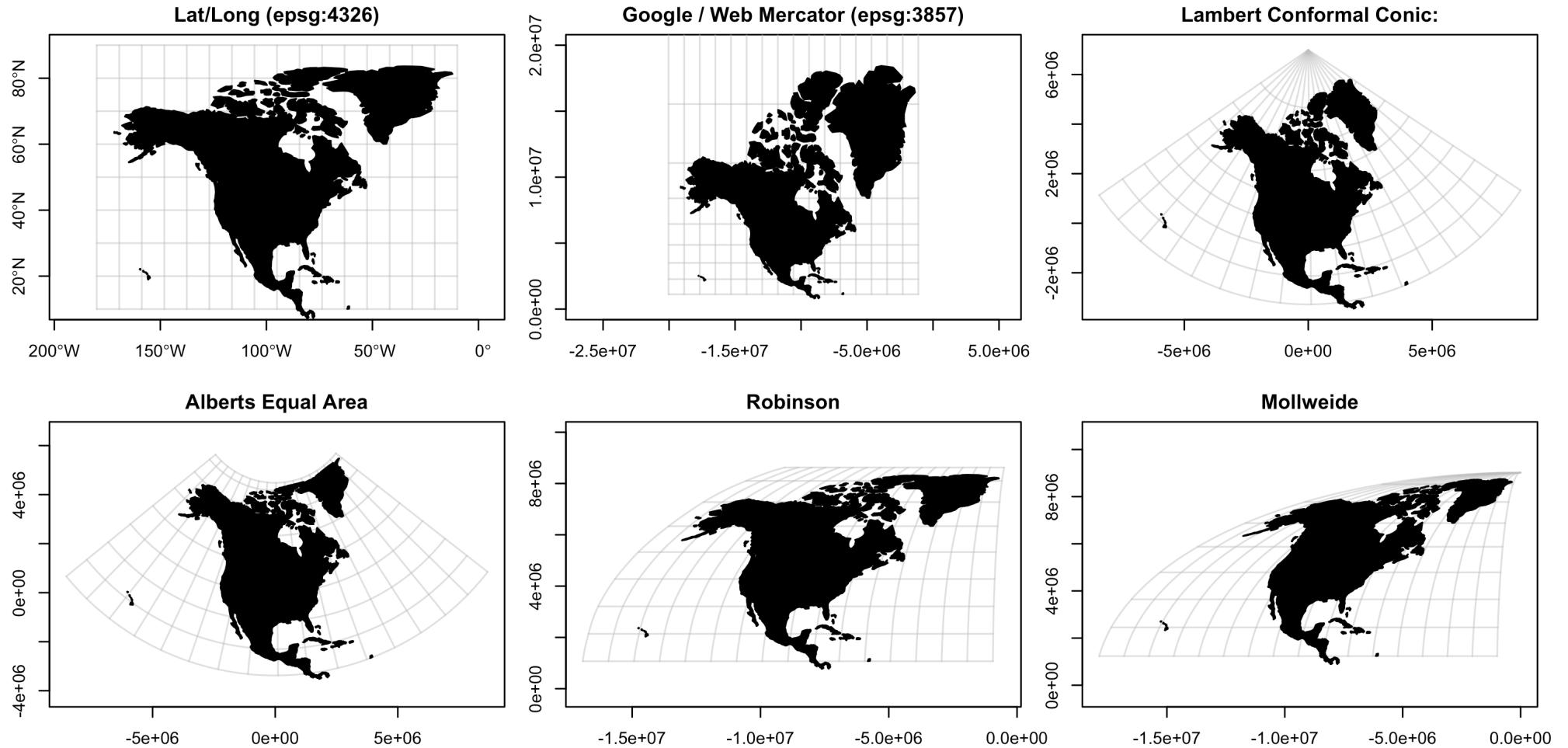
Spatial data & GIS tools

Lecture 22

Dr. Colin Rundel

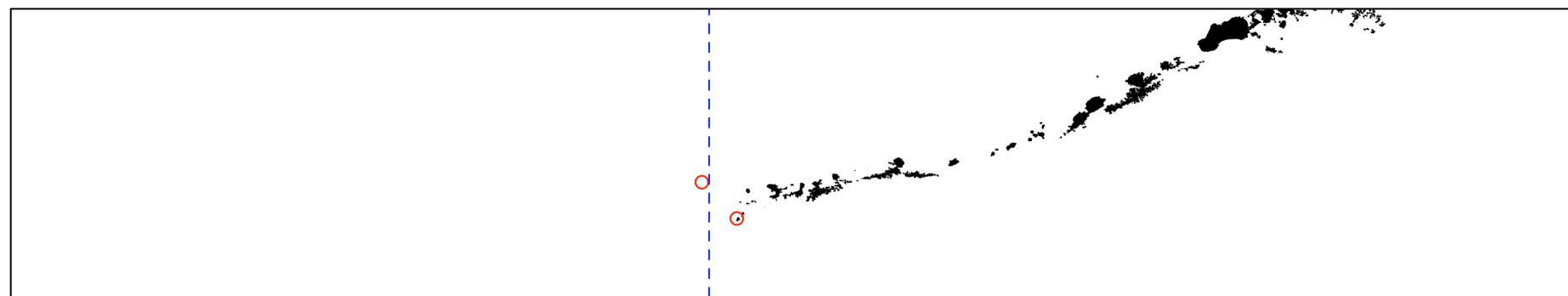
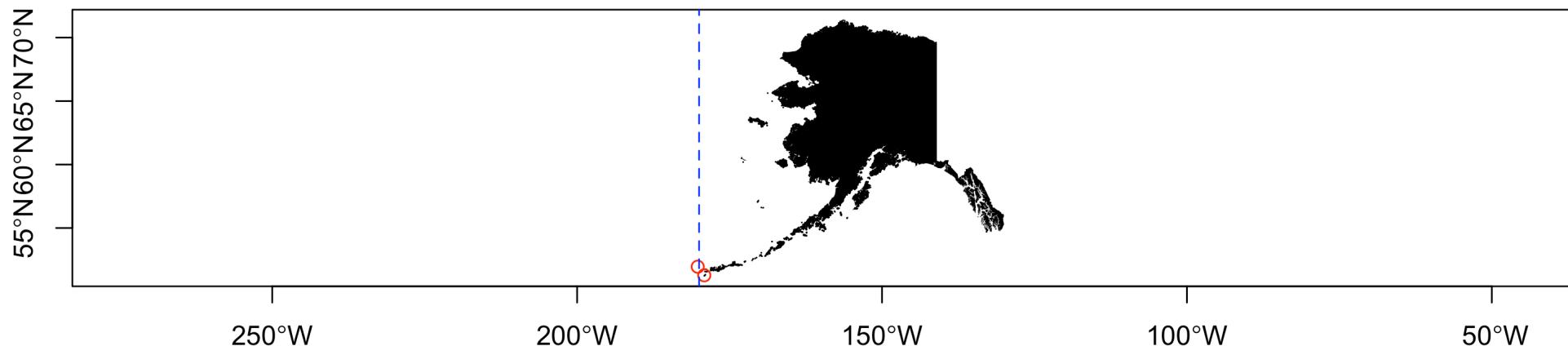
Geospatial stuff is hard

Projections



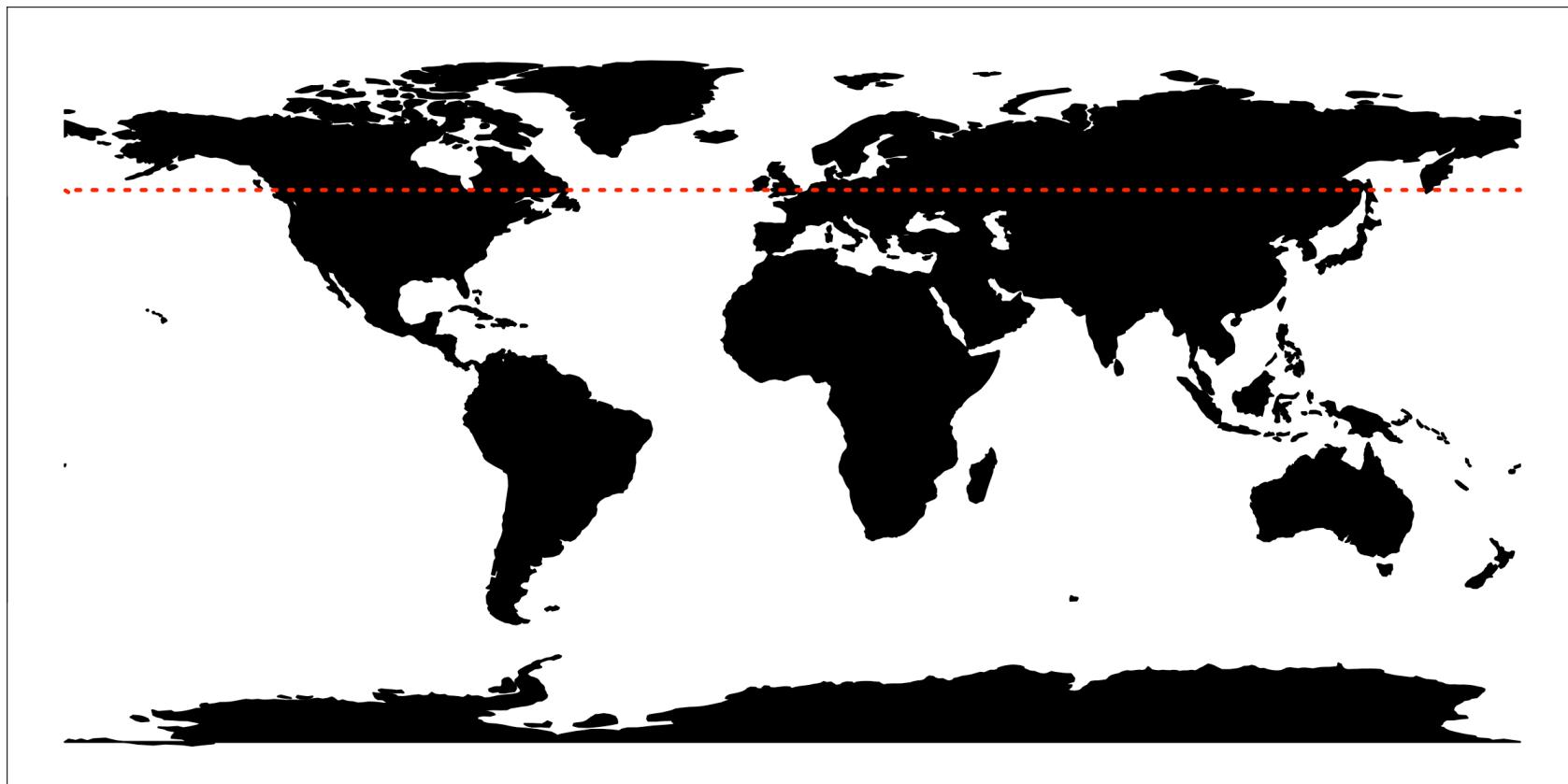
Dateline

How long is the flight between the Western most and the Eastern most points in the US?



Great circle distance

```
1 par(mar=c(0,0,0,0))
2 ak1 = c(179.776, 51.952)
3 ak2 = c(-179.146, 51.273)
4 inter = geosphere::gcIntermediate(ak1, ak2, n=50, addStartEnd=TRUE)
5 plot(st_geometry(world), col="black", ylim=c(-90,90), axes=TRUE)
6 lines(inter,col='red',lwd=2,lty=3)
```



Relationships

Geospatial Data and R

Packages for geospatial data in R

R has a rich package ecosystem for read/writing, manipulating, and analyzing geospatial data. Some core packages:

- [sp](#) - ~~core classes for handling spatial data, additional utility functions~~ - **Deprecated**
- [rgdal](#) - ~~R interface to [gdal](#) (Geospatial Data Abstraction Library) for reading and writing spatial data~~ - **Deprecated**
- [rgeos](#) - ~~R interface to [geos](#) (Geometry Engine Open Source) library for querying and manipulating spatial data. Reading and writing WKT~~ - **Deprecated**
- [sf](#) - Combines the functionality of [sp](#), [rgdal](#), and [rgeos](#) into a single package based on tidy simple features.
- [raster](#) - classes and tools for handling spatial raster data.
- [terra](#) - Methods for spatial data analysis with vector (points, lines, polygons) and raster (grid) data. Replaces [raster](#).
- [stars](#) - Reading, manipulating, writing and plotting spatiotemporal arrays (rasters)

See more - [Spatial task view](#)

Installing sf

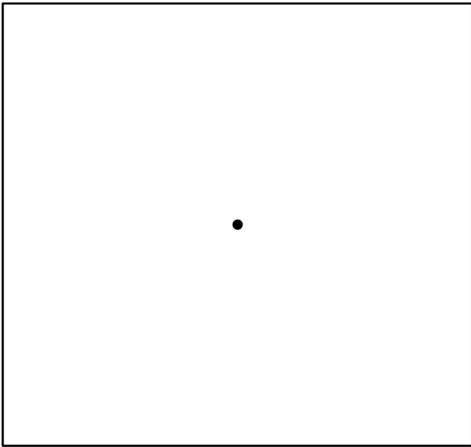
This is the hardest part of using the `sf` package, difficulty comes from its dependence on several external libraries (`geos`, `gdal`, and `proj`).

- *Windows* - installing from source works when Rtools is installed (system requirements are downloaded from rwinlib)
- *MacOS* - install dependencies via homebrew: `gdal`, `geos`, `proj`, `udunits`.
- *Linux* - Install development packages for GDAL ($\geq 2.0.0$), GEOS ($\geq 3.3.0$), Proj4 ($\geq 4.8.0$), udunits2 from your package manager of choice.

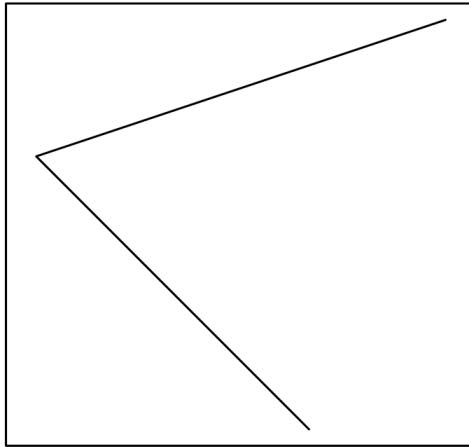
More specific details are included in the repo [README](#) on github.

Simple Features

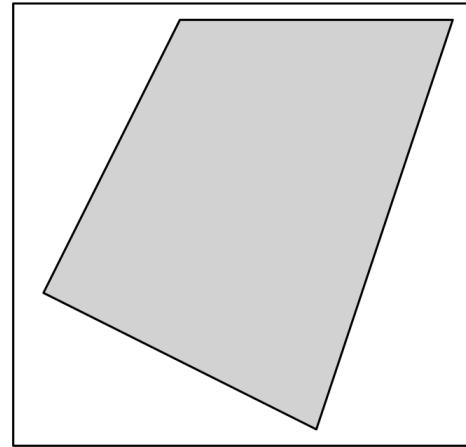
Point



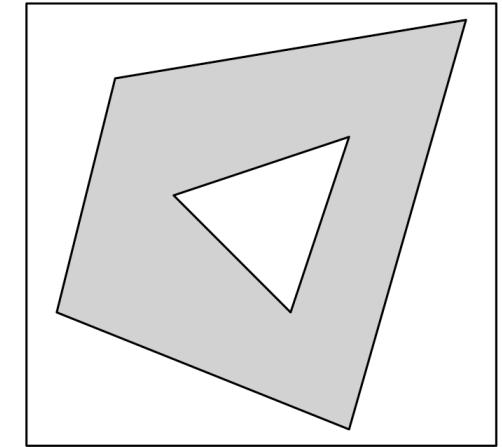
Linestring



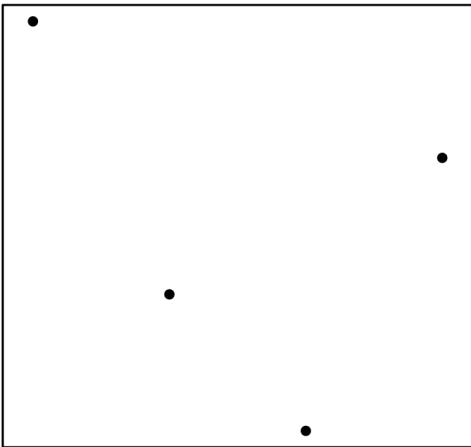
Polygon



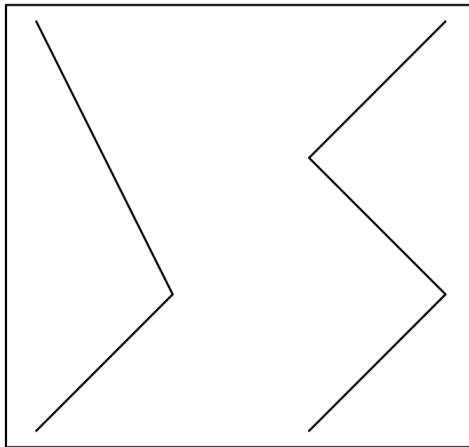
Polygon w/ Hole(s)



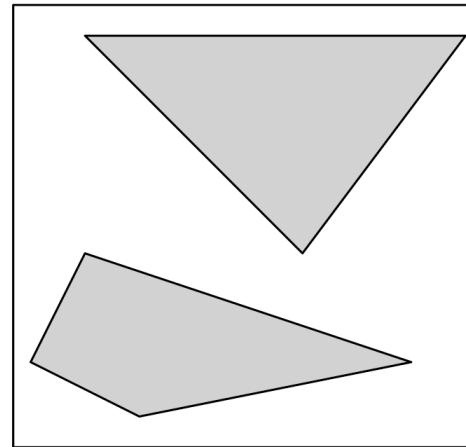
Multipoint



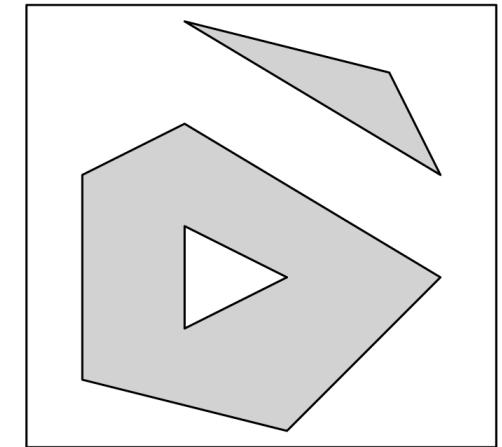
Multilinestring



Multipolygon



Multipolygon w/ Hole(s)



Reading, writing, and converting simple features

- `st_read / st_write` - Shapefile, GeoJSON, KML, ...
- `read_sf / write_sf` - Same, supports tibbles ...
- `st_as_sfc / st_as_wkt` - WKT
- `st_as_sfc / st_as_binary` - WKB
- `st_as_sfc / as(x, "Spatial")` - sp

Shapefiles

```
1 fs:::dir_info("data/gis/nc_counties/") |> select(path:size)
```

```
# A tibble: 4 × 3
  path                      type     size
  <fs:::path>                <fct>   <fs:::b>
1 ...a/gis/nc_counties/nc_counties.dbf file    41K
2 ...a/gis/nc_counties/nc_counties.prj file    165
3 ...a/gis/nc_counties/nc_counties.shp file   1.41M
4 ...a/gis/nc_counties/nc_counties.shx file    900
```

NC Counties - sf + data.frame

```
1 st_read("data/gis/nc_counties/", quiet=TRUE))
```

Simple feature collection with 100 features and 8 fields

Geometry type: MULTIPOLYGON

Dimension: XY

Bounding box: xmin: -84.32186 ymin: 33.84175 xmax: -75.46003 ymax: 36.58815

Geodetic CRS: NAD83

First 10 features:

	AREA	PERIMETER	COUNTY	TYPE	P010	STATE	COUNTY	FIPS	STATE_FIPS
1	0.11175964	1.610396	Ashe	County	1994	NC	37009	37	
2	0.06159483	1.354829	Alleghany	County	1996	NC	37005	37	
3	0.14023009	1.769388	Surry	County	1998	NC	37171	37	
4	0.08912401	1.425249	Gates	County	1999	NC	37073	37	
5	0.06865730	4.428217	Currituck	County	2000	NC	37053	37	
6	0.11859434	1.404309	Stokes	County	2001	NC	37169	37	
7	0.06259671	2.106357	Camden	County	2002	NC	37029	37	
8	0.11542955	1.462524	Warren	County	2003	NC	37185	37	
9	0.14328609	2.397293	Northampton	County	2004	NC	37131	37	
10	0.09245561	1.810778	Hertford	County	2005	NC	37091	37	

NC Counties - sf + tbl_df

```
1 (nc = read_sf("data/gis/nc_counties/"))
```

Simple feature collection with 100 features and 8 fields

Geometry type: MULTIPOLYGON

Dimension: XY

Bounding box: xmin: -84.32186 ymin: 33.84175 xmax: -75.46003 ymax: 36.58815

Geodetic CRS: NAD83

A tibble: 100 × 9

	AREA	PERIMETER	COUNTYP010	STATE	COUNTY	FIPS	STATE_FIPS	SQUARE_MIL
	<dbl>	<dbl>	<dbl>	<chr>	<chr>	<chr>	<chr>	<dbl>
1	0.112	1.61	1994	NC	Ashe County	37009	37	429.
2	0.0616	1.35	1996	NC	Alleghany County	37005	37	236.
3	0.140	1.77	1998	NC	Surry County	37171	37	539.
4	0.0891	1.43	1999	NC	Gates County	37073	37	342.
5	0.0687	4.43	2000	NC	Currituck County	37053	37	264.
6	0.119	1.40	2001	NC	Stokes County	37169	37	456.
7	0.0626	2.11	2002	NC	Camden County	37029	37	241.
8	0.115	1.46	2003	NC	Warren County	37185	37	444.
9	0.143	2.40	2004	NC	Northampton Count.v	37131	37	551.

sf classes

```
1 str(nc, max.level=1)
```

```
sf [100 × 9] (S3: sf/tbl_df/tbl/data.frame)
- attr(*, "sf_column")= chr "geometry"
- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",...: NA NA NA NA NA
NA NA NA
...- attr(*, "names")= chr [1:8] "AREA" "PERIMETER" "COUNTYP010" "STATE" ...
```

```
1 class(nc)
```

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

```
1 class(nc$geometry)
```

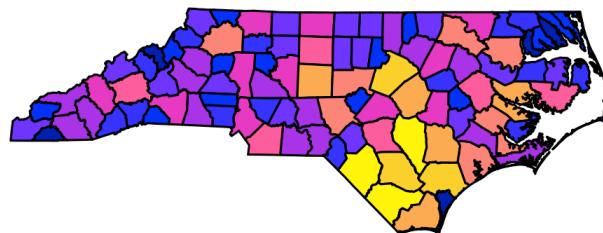
```
1 class(nc$geometry[[1]])
```

```
[1] "XY"           "MULTIPOLYGON" "sfg"
```

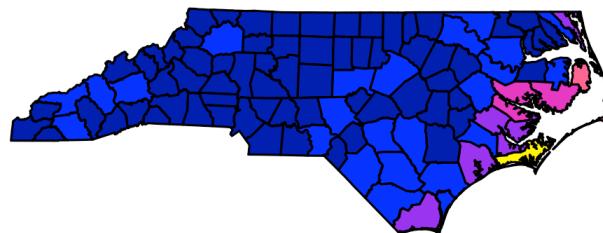
Plotting

```
1 plot(nc)
```

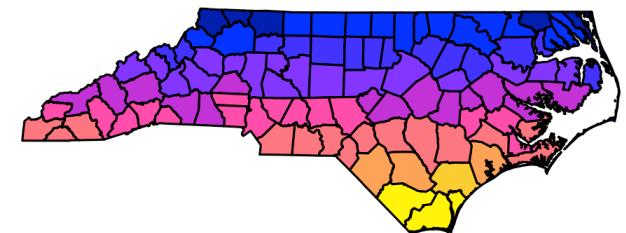
AREA



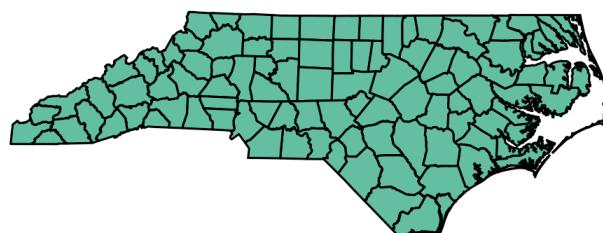
PERIMETER



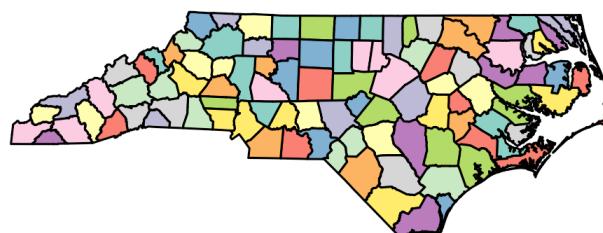
COUNTYP010



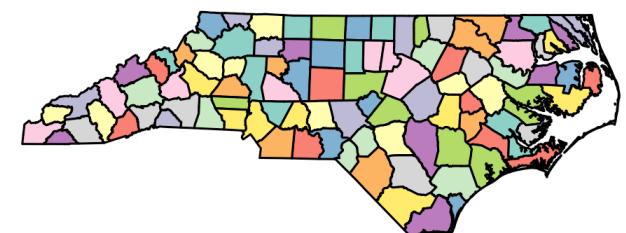
STATE



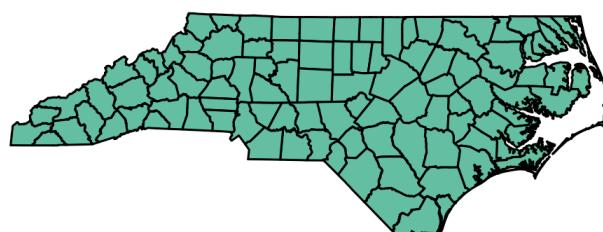
COUNTY



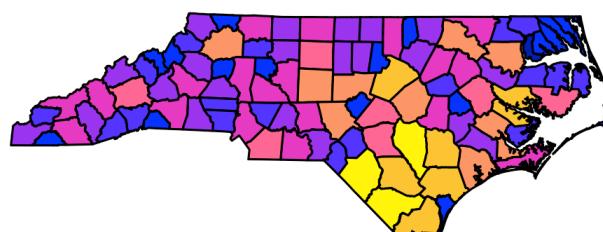
FIPS



STATE_FIPS

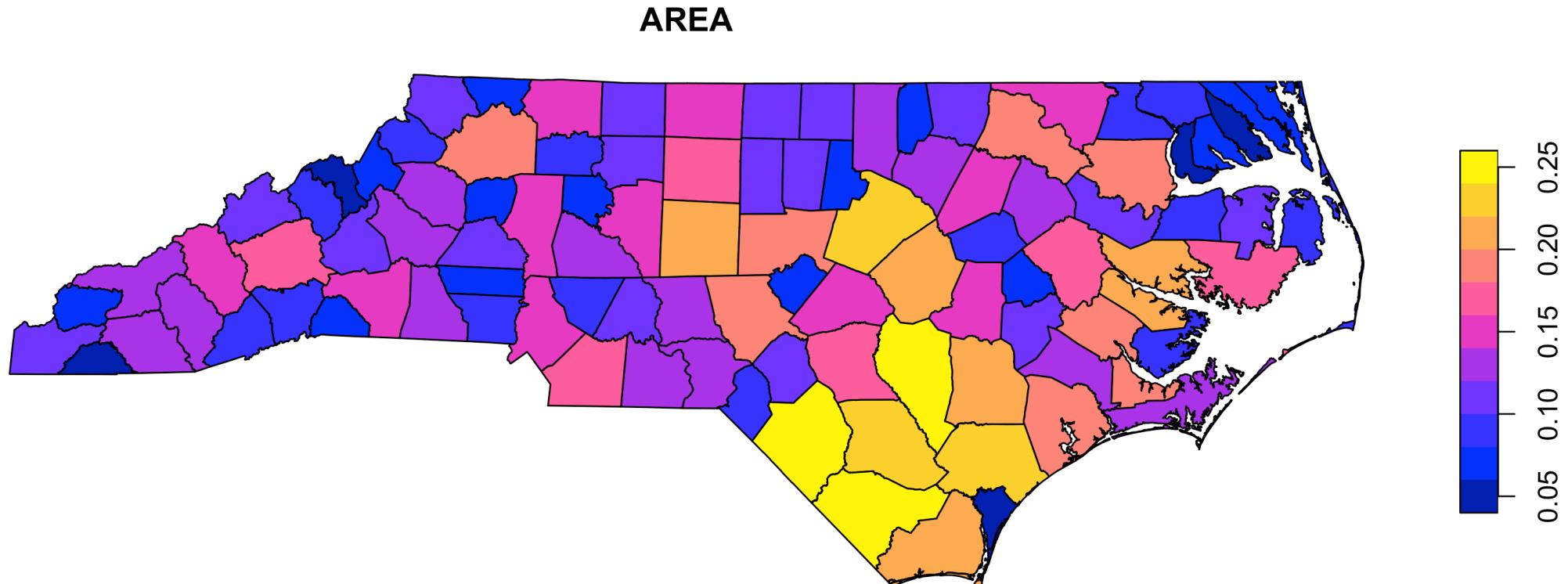


SQUARE_MIL



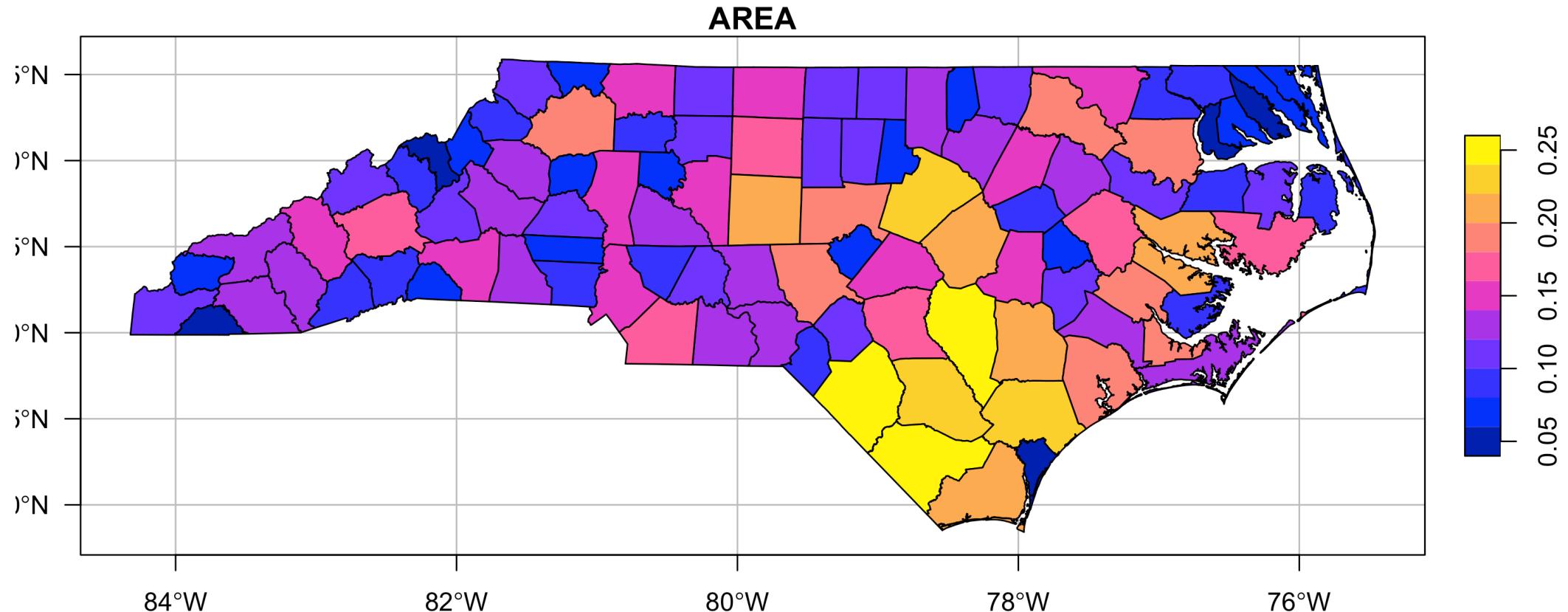
More Plotting

```
1 plot(nc[ "AREA" ])
```



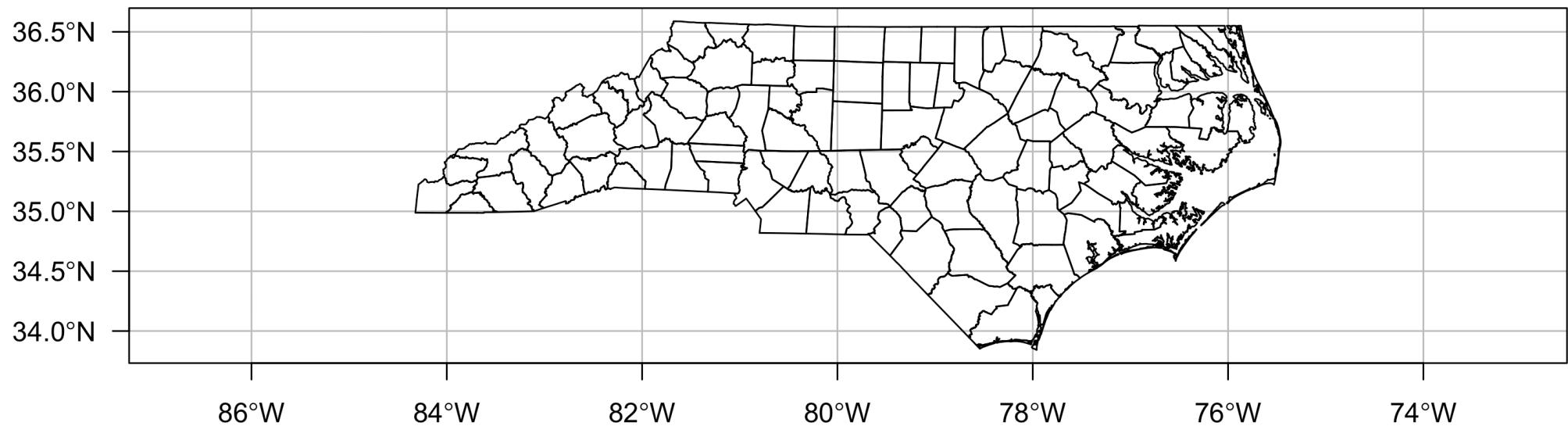
Graticules

```
1 plot(nc[ "AREA" ], graticule=TRUE, axes=TRUE, las=1)
```



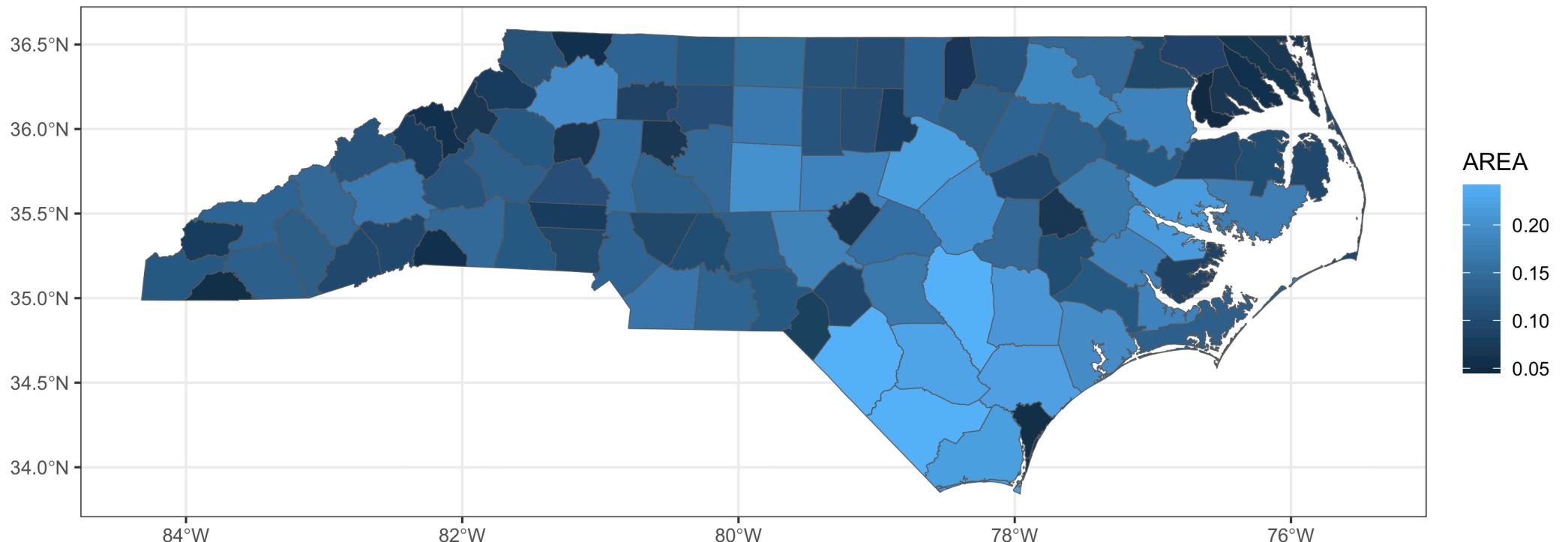
Geometries

```
1 plot(st_geometry(nc), graticule=TRUE, axes=TRUE, las=1)
```



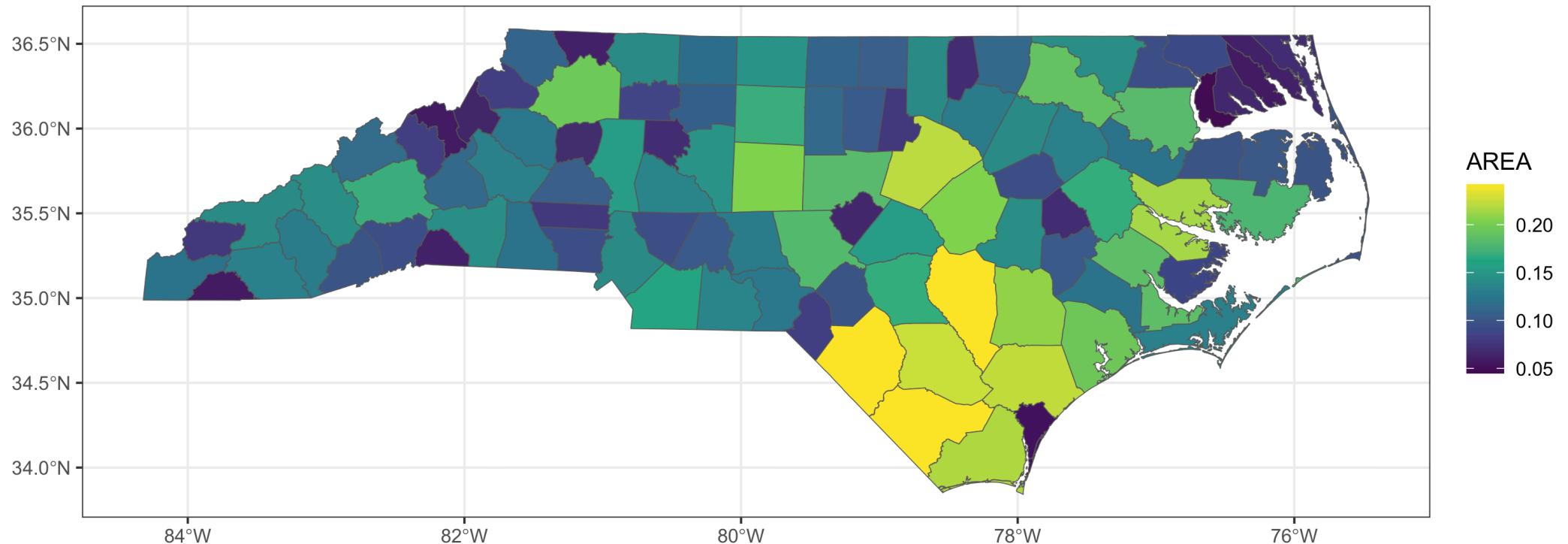
ggplot2

```
1 ggplot(nc, aes(fill=AREA)) +  
2   geom_sf()
```



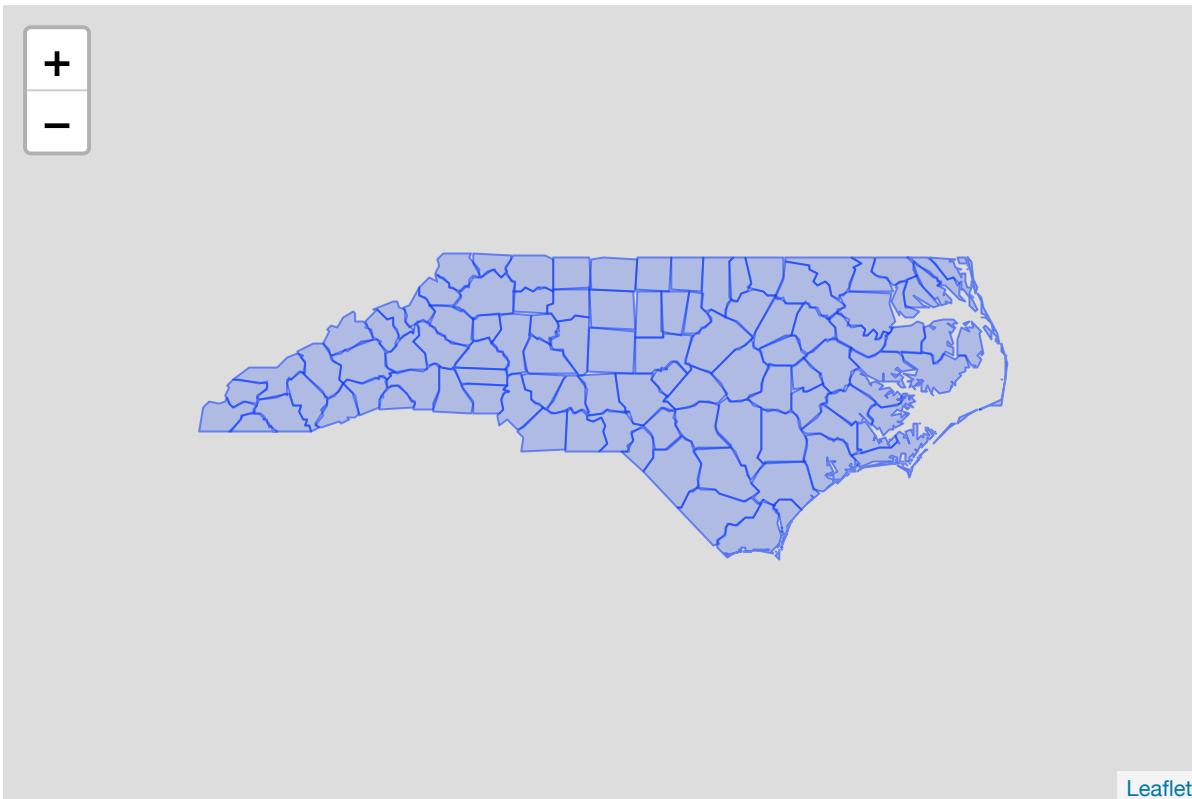
ggplot2 + palettes

```
1 ggplot(nc, aes(fill=AREA)) +  
2   geom_sf() +  
3   scale_fill_viridis_c()
```



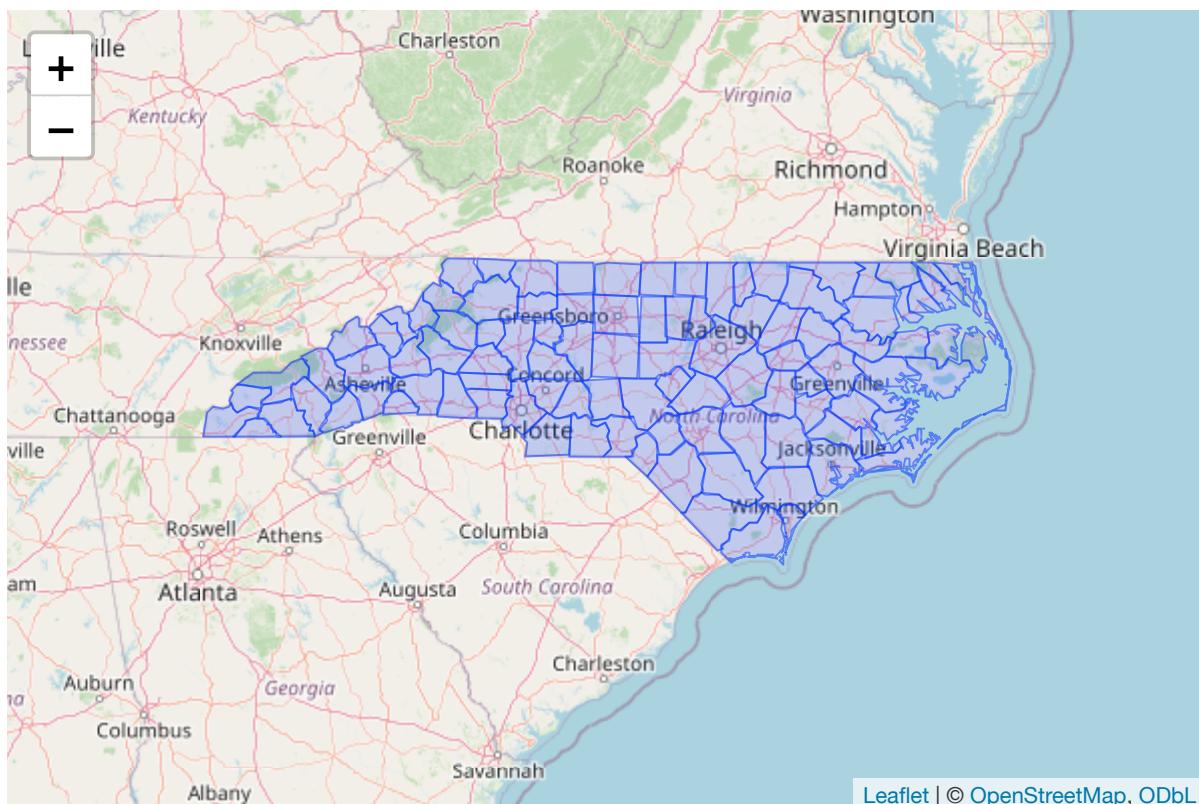
leaflet

```
1 st_transform(nc, "+proj=longlat +datum=WGS84") |>
2 leaflet::leaflet(width = 600, height = 400) |>
3 leaflet::addPolygons(
4   weight = 1, popup = ~COUNTY,
5   highlightOptions = leaflet::highlightOptions(color = "red", weight = 2, bringToFront = TRUE)
6 )
```



leaflet + tiles

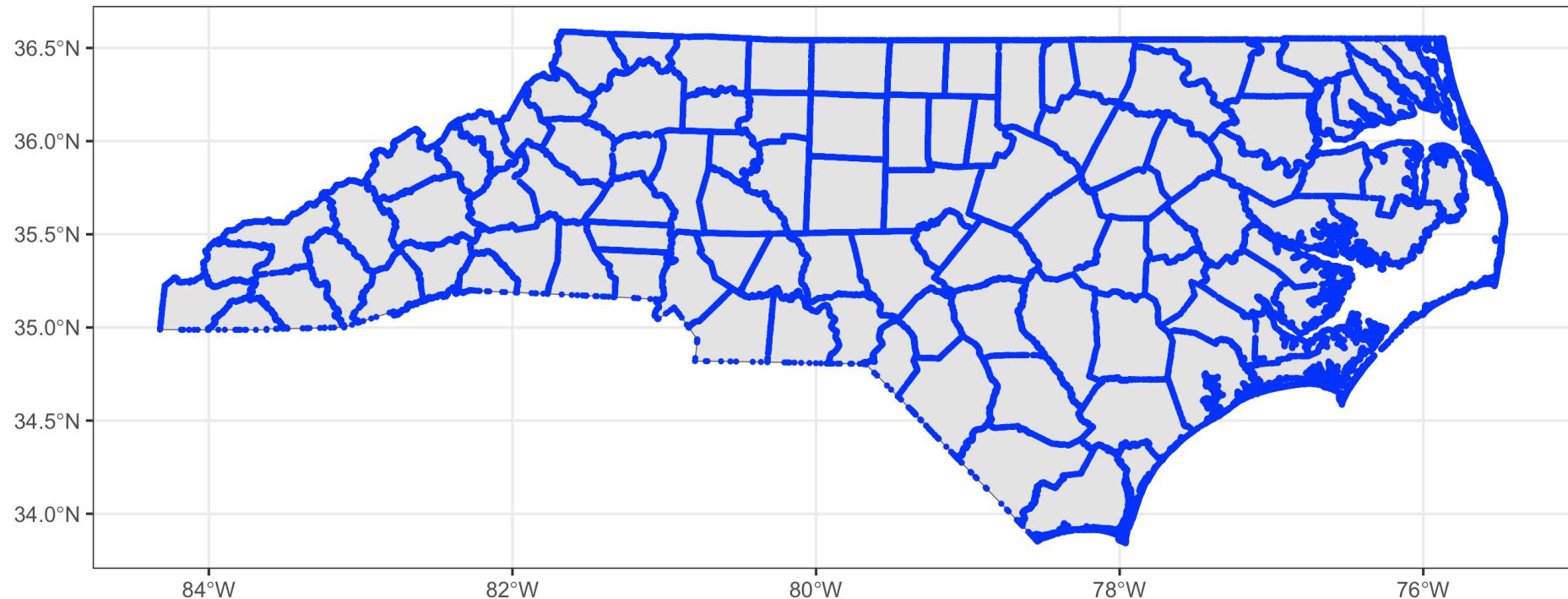
```
1 st_transform(nc, "+proj=longlat +datum=WGS84") |>
2 leaflet::leaflet(width = 600, height = 400) |>
3 leaflet::addPolygons(
4   weight = 1,
5   popup = ~COUNTY,
6   highlightOptions = leaflet::highlightOptions(color = "red", weight = 2, bringToFront = TRUE)
7 ) |>
8 leaflet::addTiles()
```



GIS in R

Geometry casting

```
1 nc_pts = st_cast(nc, "MULTIPOINT")
2 ggplot() +
3   geom_sf(data=nc) +
4   geom_sf(data=nc_pts, size=0.5, color="blue")
```

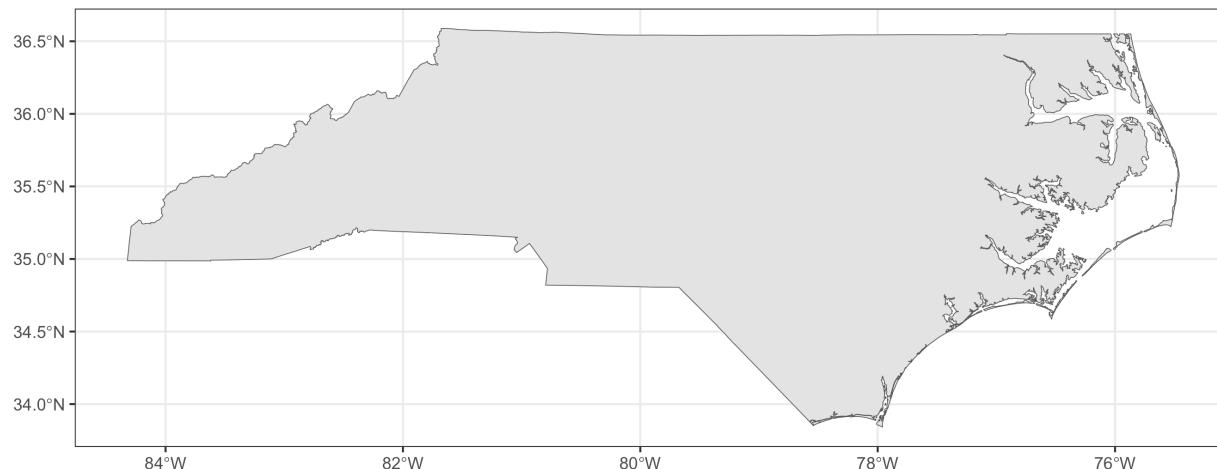


Joining

```
1 nc_state = st_union(nc))
```

Geometry set for 1 feature
Geometry type: MULTIPOLYGON
Dimension: XY
Bounding box: xmin: -84.32186 ymin: 33.84175 xmax: -75.46003 ymax: 36.58815
Geodetic CRS: NAD83
MULTIPOLYGON (((-75.82791 36.19327, -75.82931 3...

```
1 ggplot() + geom_sf(data=nc_state)
```

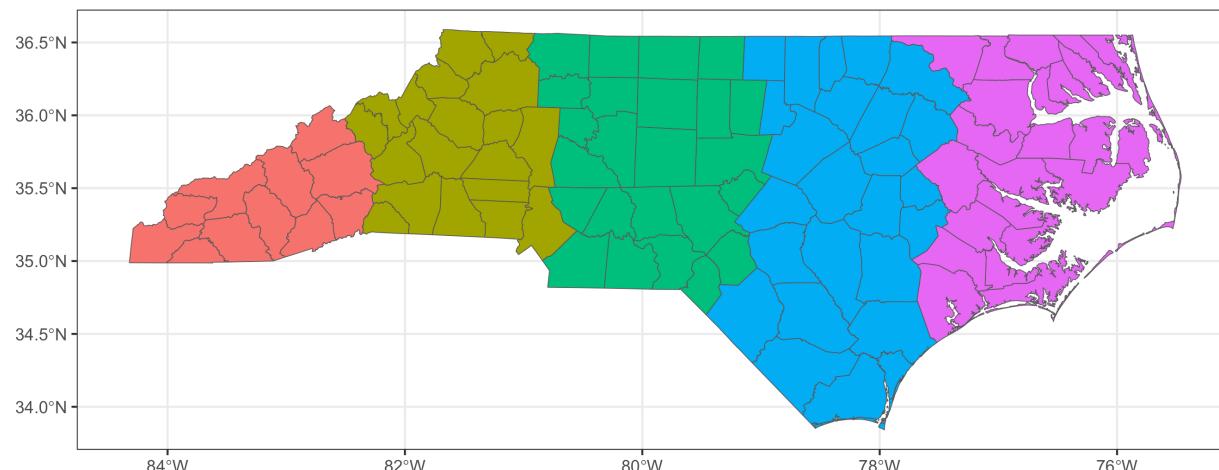


sf & dplyr

```
1 nc_cut = nc |>
2   mutate(
3     ctr_x = st_centroid(nc) |> st_coordinates() |> (\(x) x[,1])(),
4     region = cut(ctr_x, breaks = 5)
5   )
```

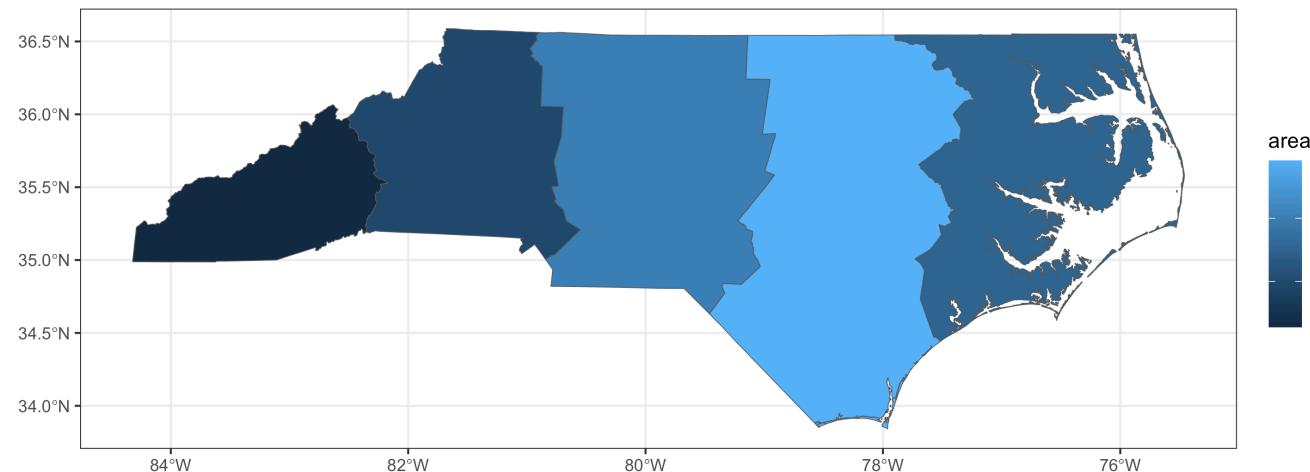
Warning: There was 1 warning in `stopifnot()`.
i In argument: `ctr_x = (function(x) x[, 1])(st_coordinates(st_centroid(nc)))`.
Caused by warning:
! st_centroid assumes attributes are constant over geometries

```
1 ggplot(nc_cut) +
2   geom_sf(aes(fill=region)) +
3   guides(fill = "none")
```



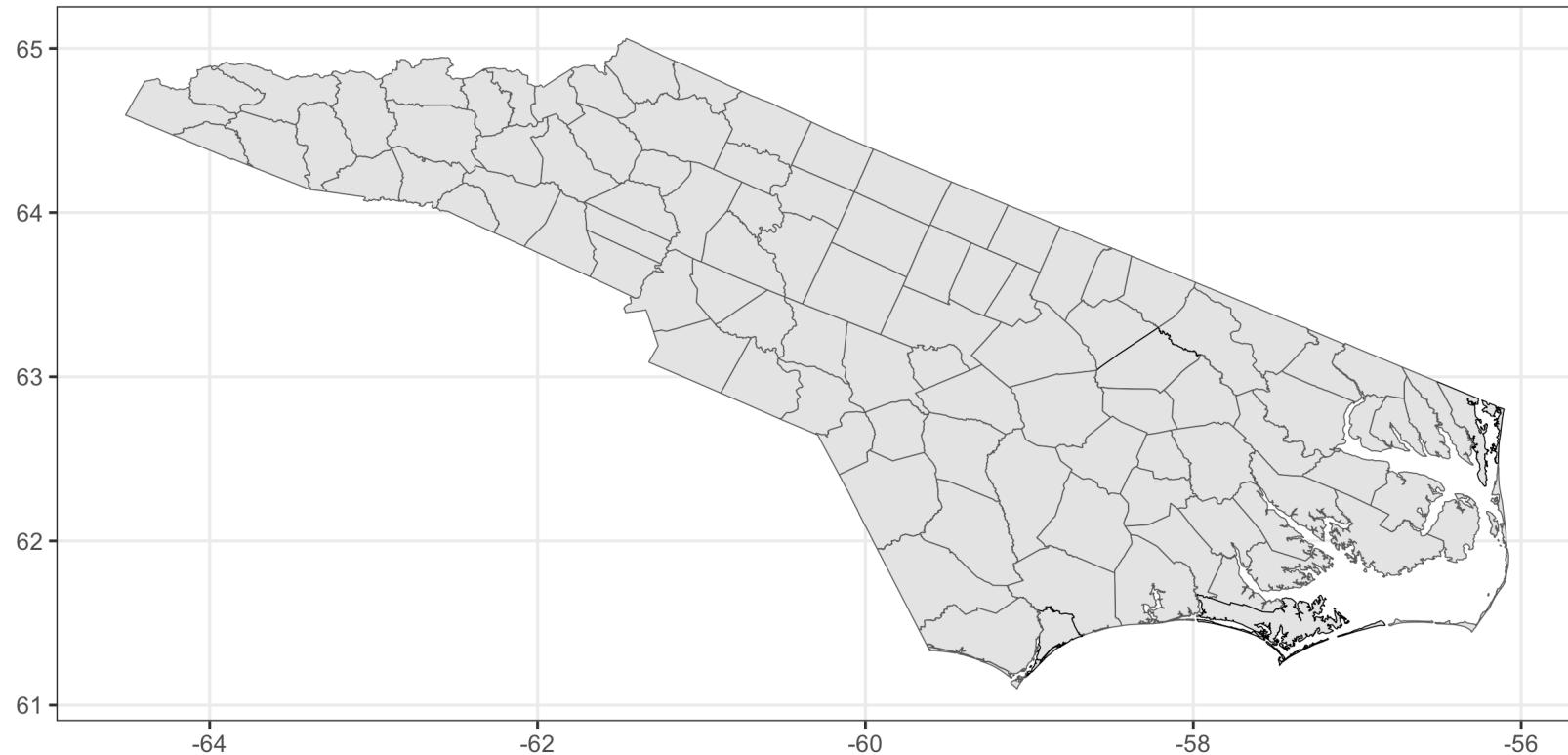
sf & dplyr (cont.)

```
1 nc_cut2 = nc_cut |>  
2   group_by(region) |>  
3   summarize(  
4     area = sum(AREA)  
5   )  
6  
7 ggplot() + geom_sf(data=nc_cut2, aes(fill=area))
```



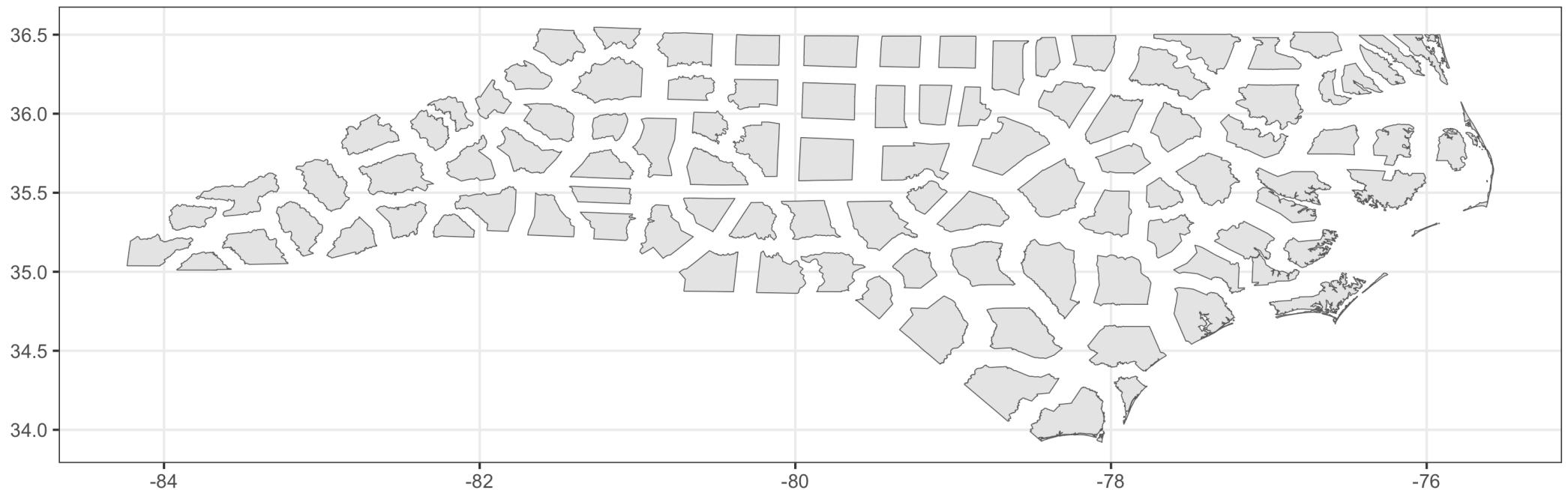
Affine Transformations

```
1 rotate = function(a) matrix(c(cos(a), sin(a), -sin(a), cos(a)), 2, 2)
2
3 state_rotate = (st_geometry(nc) * rotate(pi/8)) |> lwgeom::lwgeom_make_valid()
4 ggplot() + geom_sf(data=state_rotate)
```



Scaling + Translations

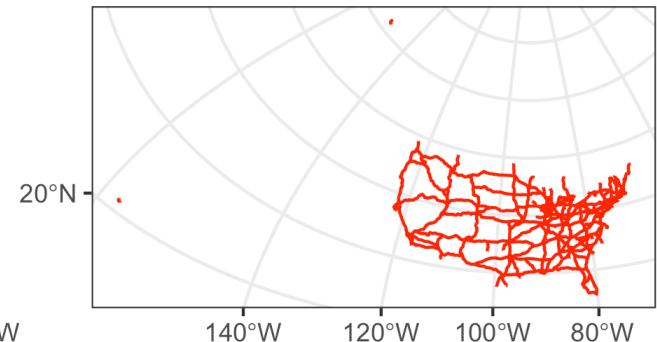
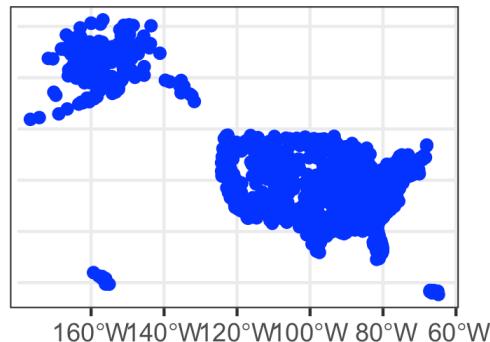
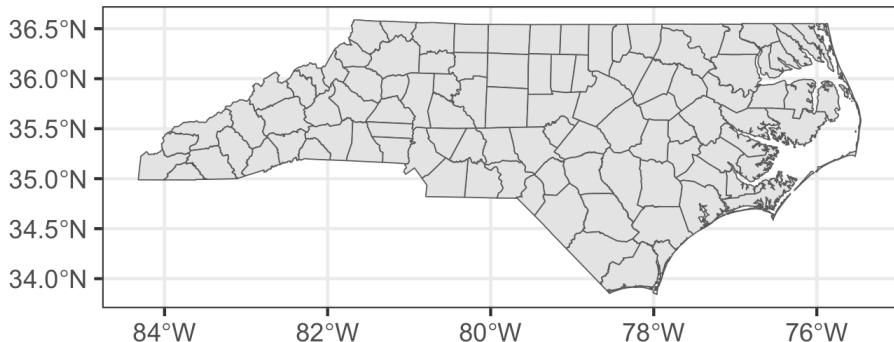
```
1 ctrd = st_centroid(st_geometry(nc))
2 nc_scaled = (st_geometry(nc) - ctrd) * 0.66 + ctrd
3
4 ggplot() + geom_sf(data=nc_scaled)
```



Some other data

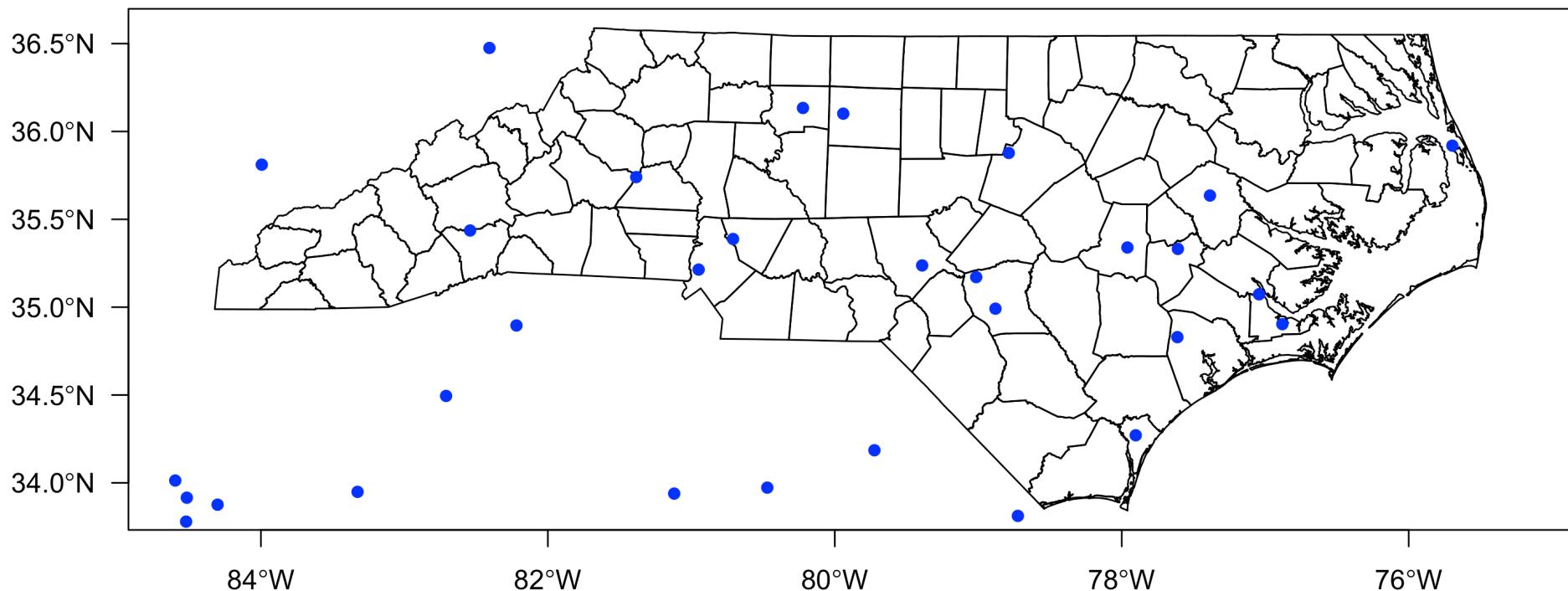
```
1 air = read_sf("data/gis/airports/", quiet=TRUE)
2 hwy = read_sf("data/gis/us_interstates/", quiet=TRUE)
```

```
1 (ggplot(nc) + geom_sf()) +
2 (ggplot(air) + geom_sf(color = "blue")) +
3 (ggplot(hwy) + geom_sf(color = "red"))
```



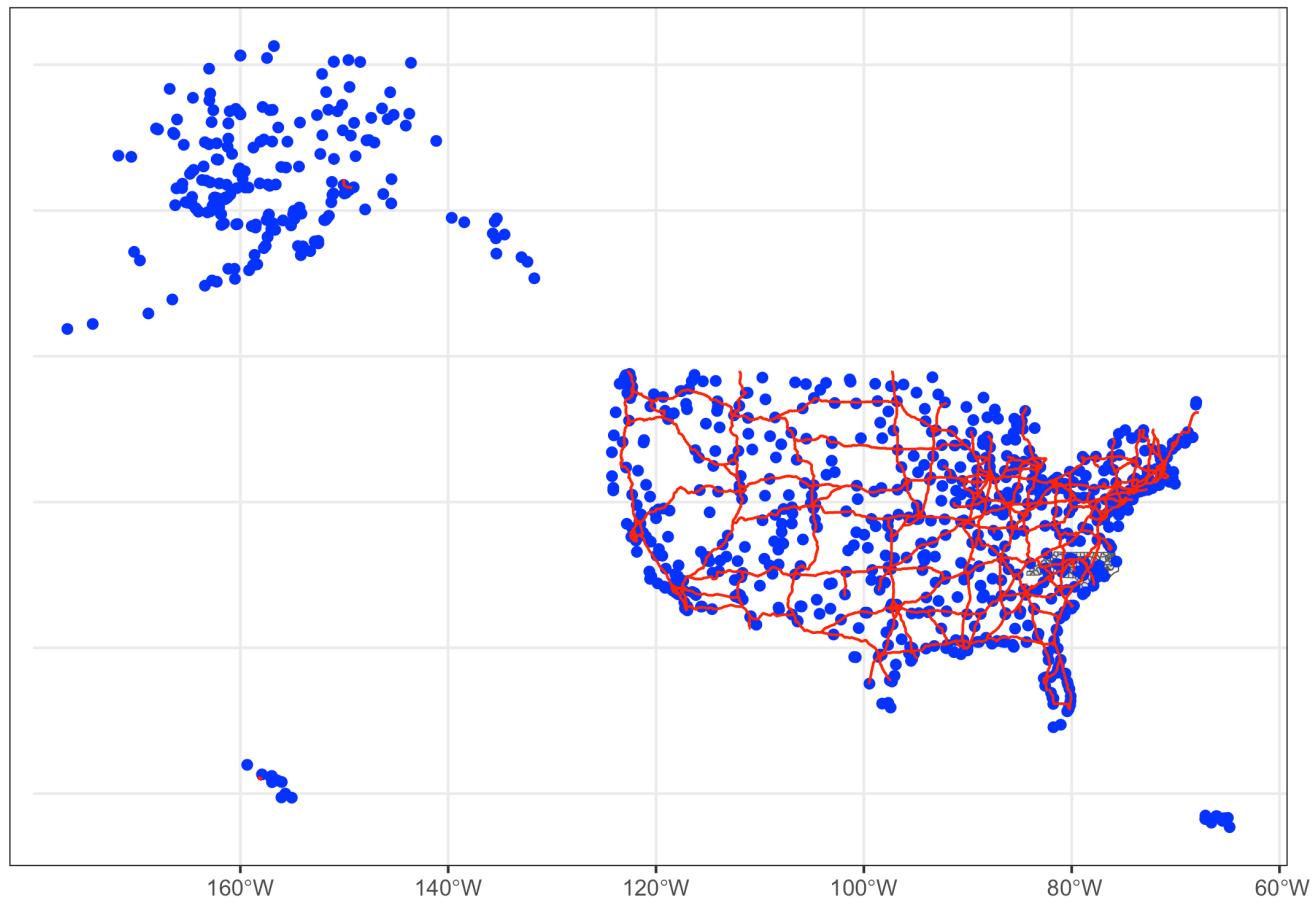
Overlays w/ base plots

```
1 plot(st_geometry(nc), axes=TRUE, las=1)
2 plot(st_geometry(air), axes=TRUE, pch=16, col="blue", main="air", add=TRUE)
3 plot(st_geometry(hwy), axes=TRUE, col="red", add=TRUE)
```



Overlays w/ ggplot

```
1 ggplot() +  
2   geom_sf(data=nc) +  
3   geom_sf(data=air, color="blue") +  
4   geom_sf(data=hwy, color="red")
```



Projections

```
1 st_crs(nc)
```

Coordinate Reference System:

User input: NAD83

wkt:

```
GEOGCRS["NAD83",
    DATUM["North American Datum 1983",
        ELLIPSOID["GRS 1980",6378137,298.257222101,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2,
        AXIS["latitude",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["longitude",east,
            ORDER[2],
```

```
1 st_crs(hwy)
```

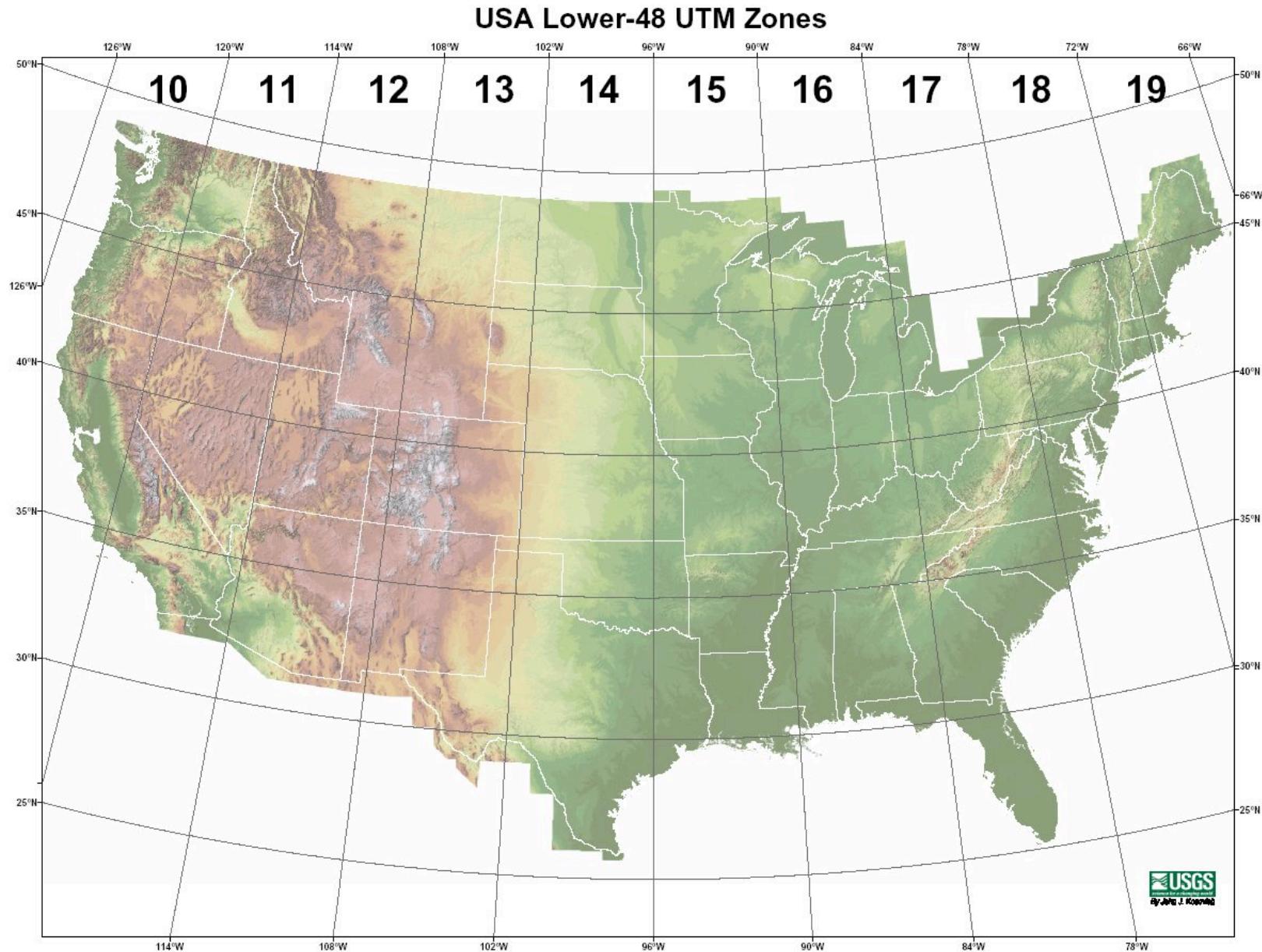
Coordinate Reference System:

User input: NAD83 / UTM zone 15N

wkt:

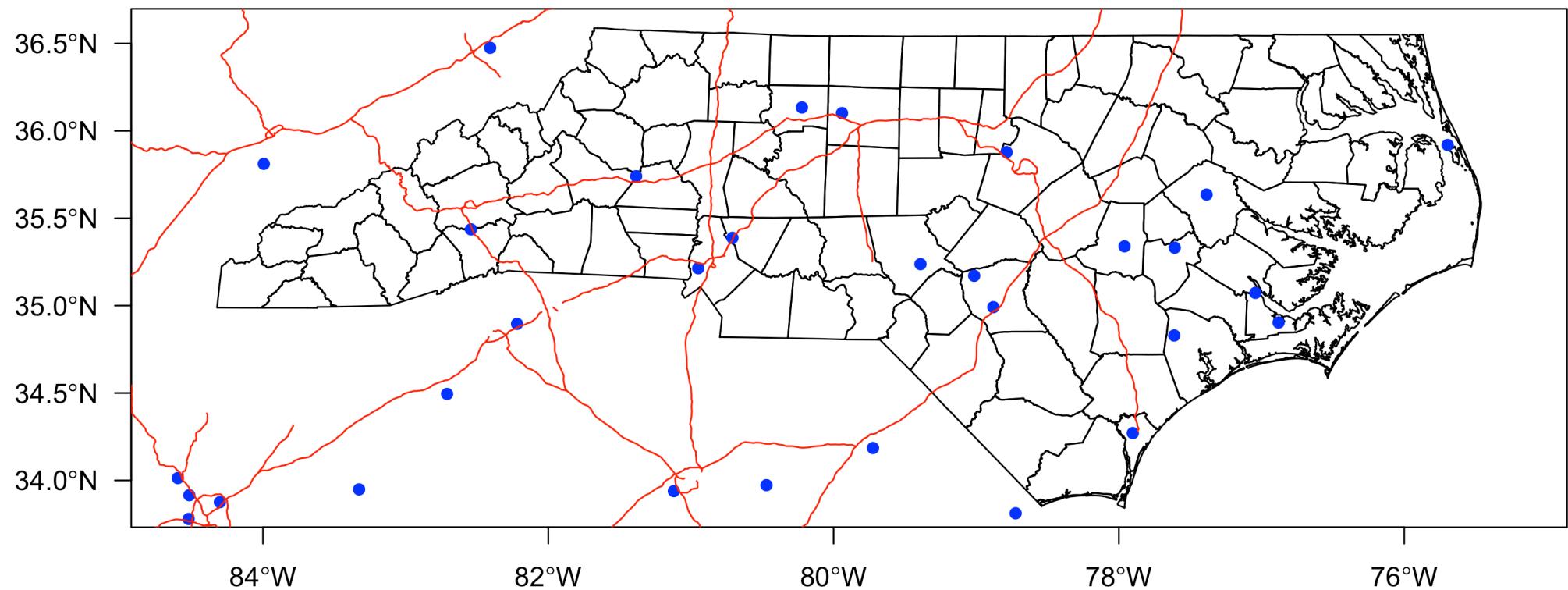
```
PROJCRS["NAD83 / UTM zone 15N",
    BASEGEOGCRS["NAD83",
        DATUM["North American Datum 1983",
            ELLIPSOID["GRS
1980",6378137,298.257222101,
            LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4269]],
    CONVERSION["UTM zone 15N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0.
```

Aside - UTM Zones



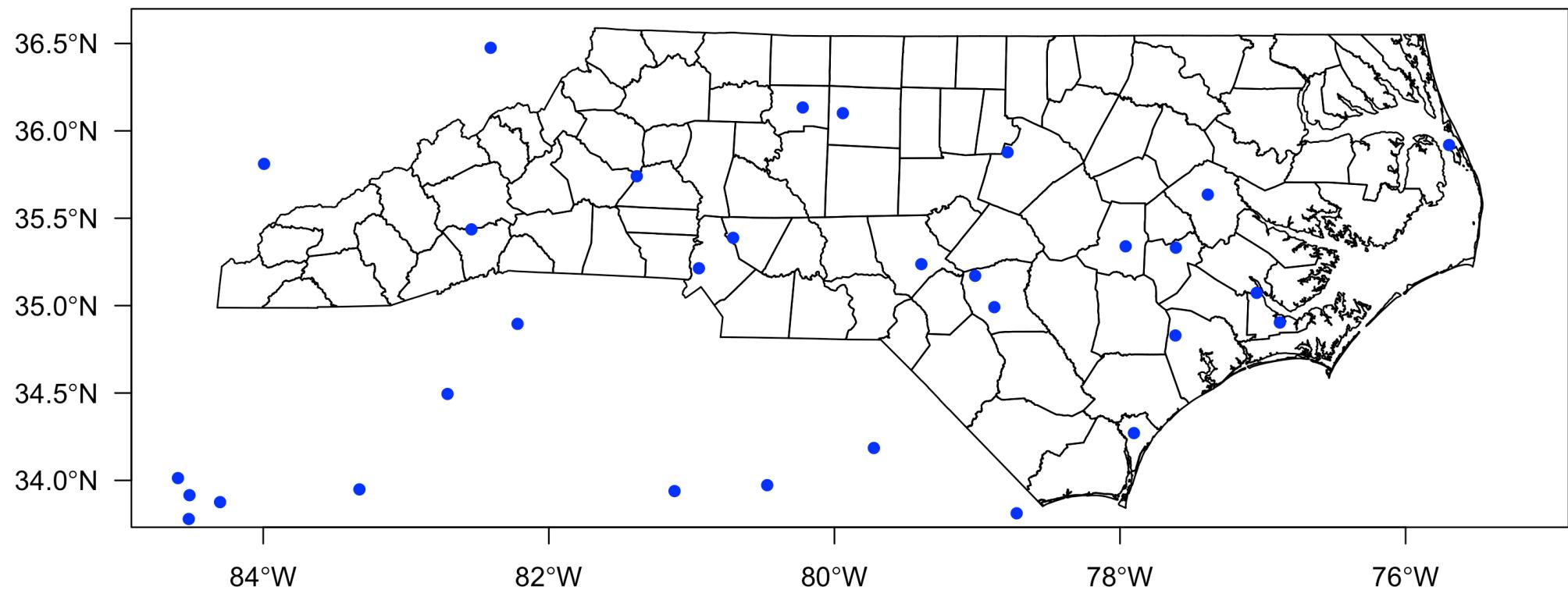
hwy -> Lat/Long

```
1 hwy = st_transform(hwy, st_crs(nc))
```



Airport Example

NC Airports



Sparse Insections

```
1 st_intersects(nc[20:30,], air) |> str()
```

```
List of 11
$ : int(0)
$ : int 268
$ : int 717
$ : int(0)
$ : int(0)
$ : int(0)
$ : int(0)
- attr(*, "predicate")= chr "intersects"
- attr(*, "region.id")= chr [1:11] "1" "2" "3" "4" ...
- attr(*, "remove_self")= logi FALSE
- attr(*, "retain_unique")= logi FALSE
- attr(*, "ncol")= int 940
```

Dense Insections

```
1 st_intersects(nc, air, sparse=FALSE) |> str()  
  
logi [1:100, 1:940] FALSE FALSE FALSE FALSE FALSE FALSE ...  
  
1 st_intersects(nc, air, sparse=FALSE) |> (\(x) x[20:30, 260:270])()  
  
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]  
[1,] FALSE  
[2,] FALSE  
[3,] FALSE  
[4,] FALSE  
[5,] FALSE  
[6,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE  
[7,] FALSE  
[8,] FALSE  
[9,] FALSE  
[10,] FALSE  
[11,] FALSE FALSE
```

Which counties have airports?

```
1 nc_air = nc |>
2   mutate(
3     n_air = map_int(
4       st_intersects(nc, air),
5       length
6     )
7   ) |>
8   filter(n_air > 0)
9
10 nc_air |> pull(COUNTY)
```

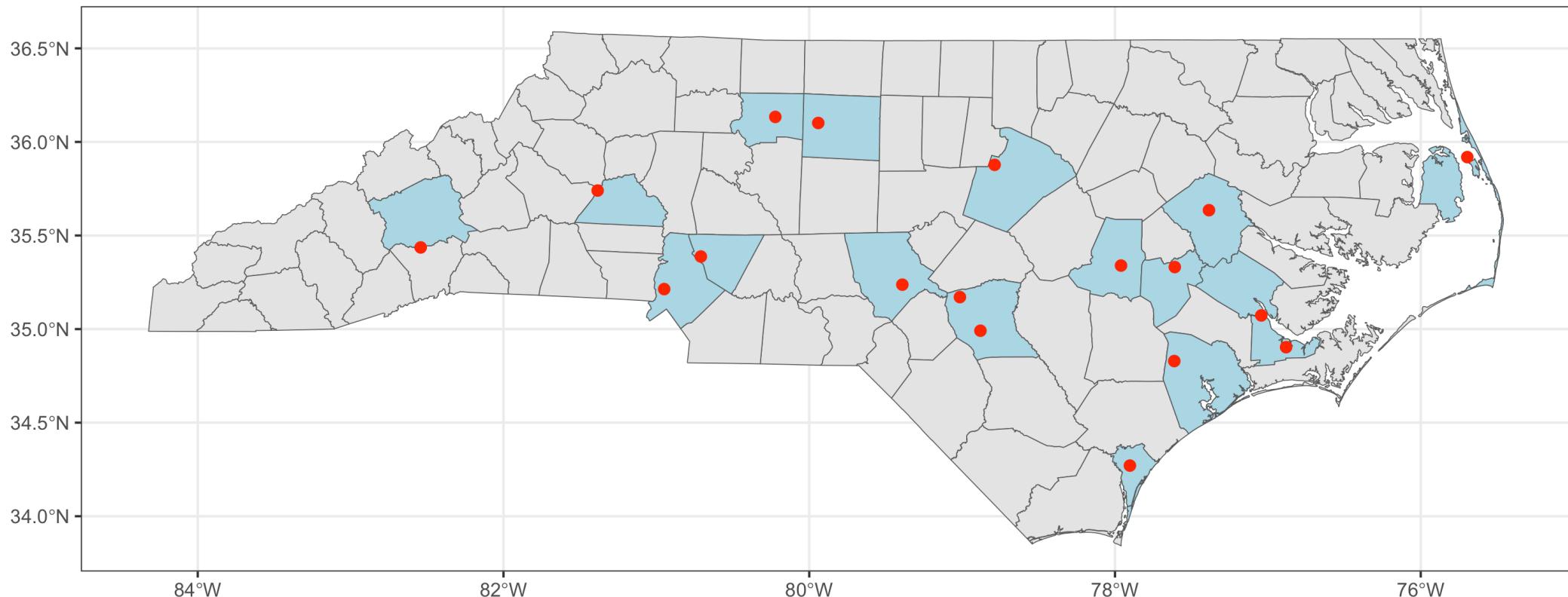
```
[1] "Forsyth County"      "Guilford County"
[3] "Dare County"         "Wake County"
[5] "Pitt County"          "Catawba County"
[7] "Buncombe County"      "Wayne County"
[9] "Mecklenburg County"    "Moore County"
[11] "Cabarrus County"      "Lenoir County"
[13] "Craven County"        "Cumberland County"
[15] "Onslow County"        "New Hanover County"
```

```
1 air_nc = air |>
2   slice(
3     st_intersects(nc, air) |>
4       unlist() |>
5       unique()
6   )
7 air_nc |> pull(AIRPT_NAME)
```

```
[1] "SMITH REYNOLDS AIRPORT"
[2] "PIEDMONT TRIAD INTERNATIONAL AIRPORT"
[3] "DARE COUNTY REGIONAL AIRPORT"
[4] "RALEIGH-DURHAM INTERNATIONAL AIRPORT"
[5] "PITT-GREENVILLE AIRPORT"
[6] "HICKORY REGIONAL AIRPORT"
[7] "ASHEVILLE REGIONAL AIRPORT"
[8] "SEYMORE JOHNSON AIR FORCE BASE"
[9] "CHARLOTTE/DOUGLAS INTERNATIONAL AIRPORT"
[10] "MOORE COUNTY AIRPORT"
[11] "CONCORD REGIONAL AIRPORT"
[12] "KINSTON REGIONAL JETPORT AT STALLINGS FIELD"
[13] "CHERRY POINT MARINE CORPS AIR STATION
/CUNNINGHAM FIELD/"
[14] "COASTAL CAROLINA REGIONAL AIRPORT"
[15] "POPE AIR FORCE BASE"
[16] "FAYETTEVILLE REGIONAL/GRANNIS FIELD"
```

Results

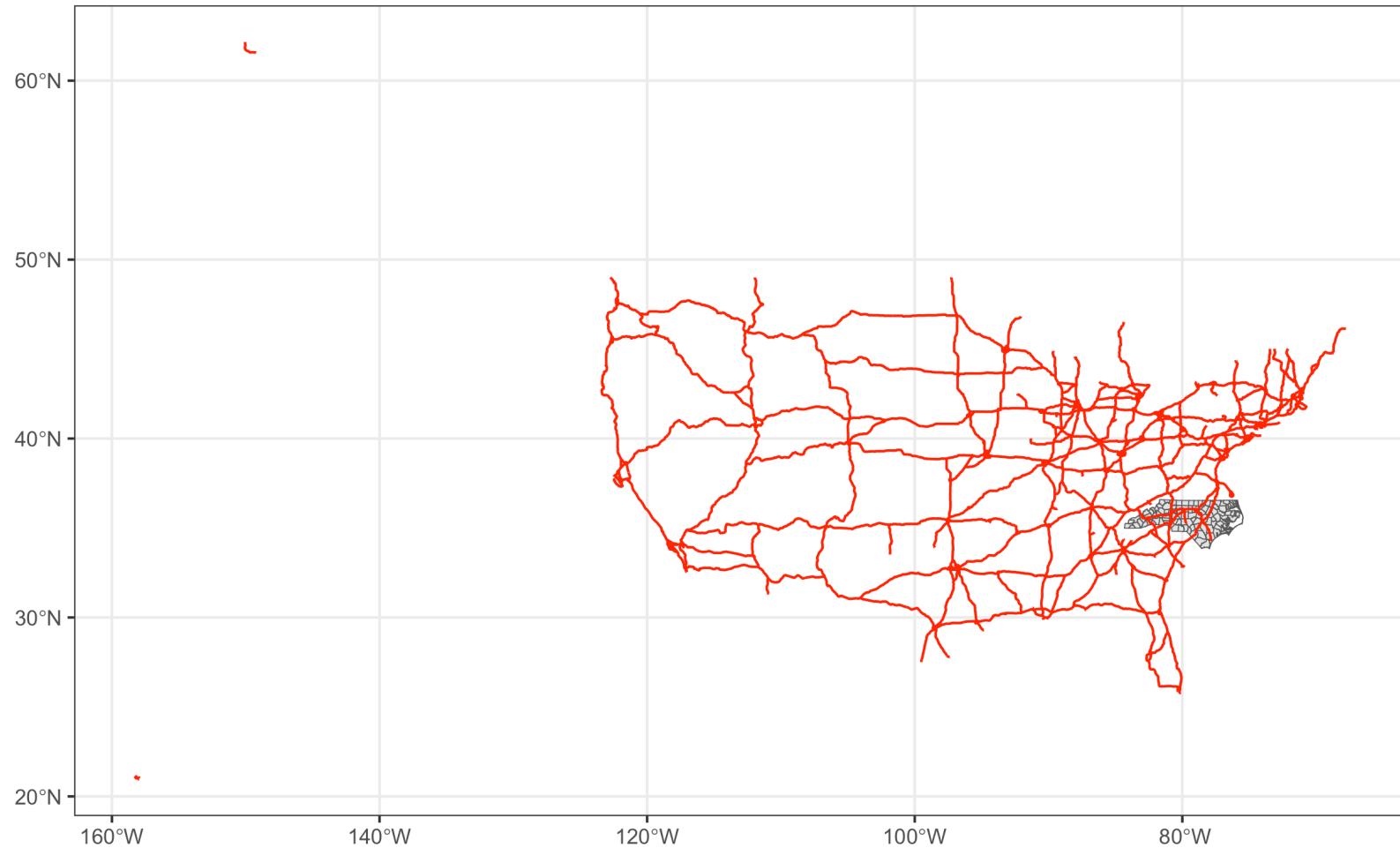
```
1 ggplot() +  
2   geom_sf(data=nc) +  
3   geom_sf(data = nc_air, fill = "lightblue") +  
4   geom_sf(data = air_nc, color = "red", size=2)
```



Highway Example

Highways

```
1 ggplot() +  
2   geom_sf(data=nc) +  
3   geom_sf(data=hwy, col='red')
```

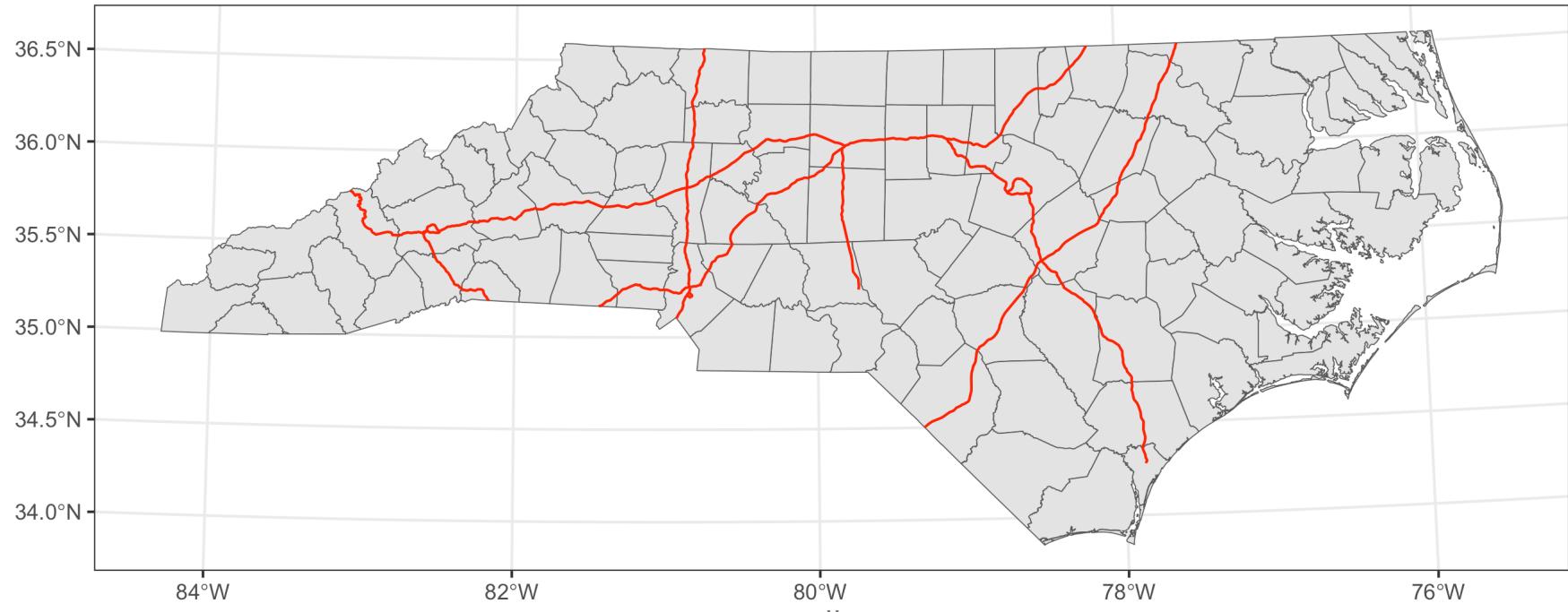


Intersection

```
1 nc_utm = st_transform(nc, "+proj=utm +zone=17 +datum=NAD83 +units=m +no_defs")
2 hwy_utm = st_transform(hwy, "+proj=utm +zone=17 +datum=NAD83 +units=m +no_defs")
3
4 hwy_nc = st_intersection(hwy_utm, nc_utm)
```

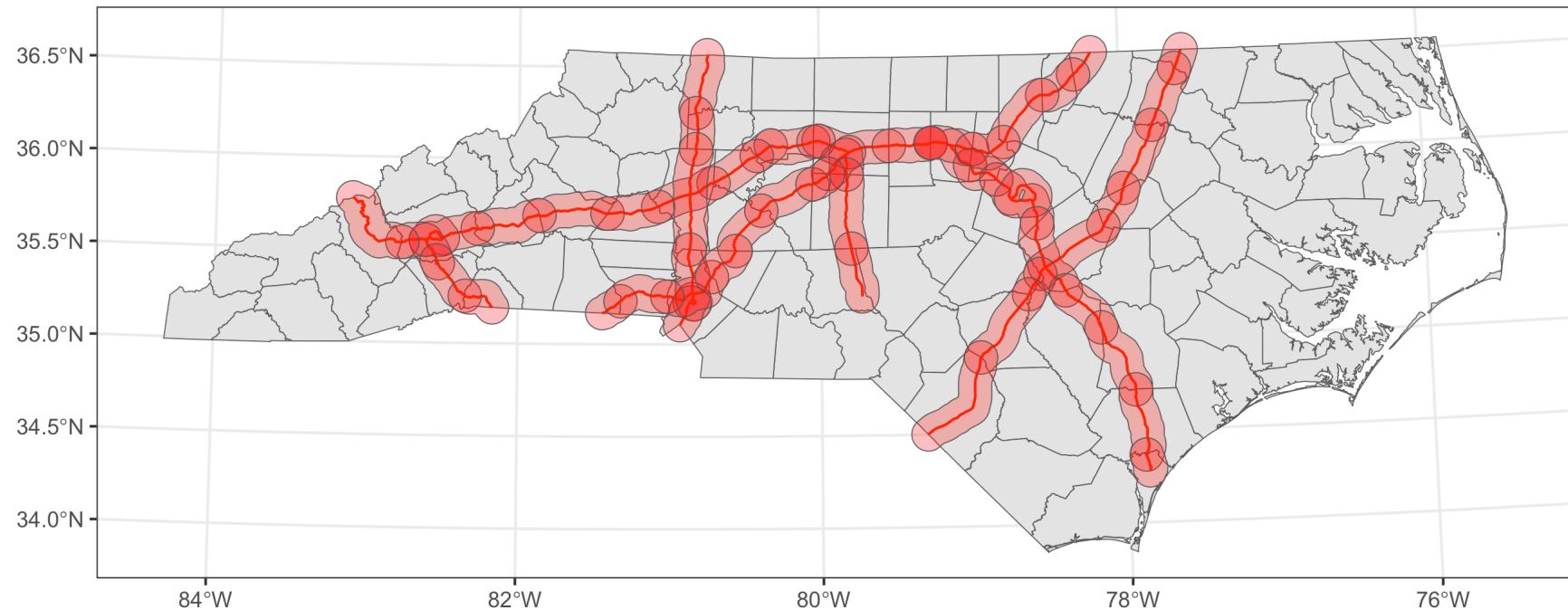
Warning: attribute variables are assumed to be spatially constant throughout all geometries

```
1 ggplot() +
2   geom_sf(data=nc_utm) +
3   geom_sf(data=hwy_nc, col='red')
```



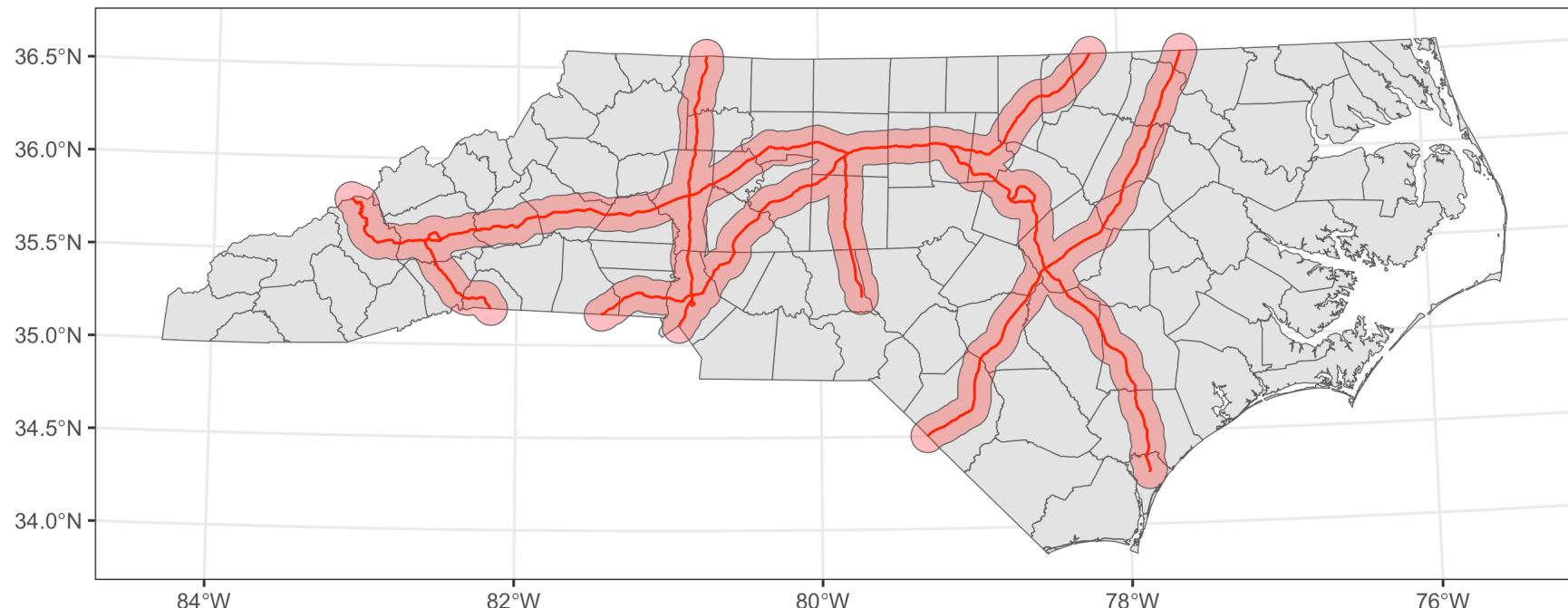
Buffering

```
1 hwy_nc_buffer = hwy_nc |>  
2   st_buffer(10000)  
3  
4 ggplot() +  
5   geom_sf(data=nc_utm) +  
6   geom_sf(data=hwy_nc, color='red') +  
7   geom_sf(data=hwy_nc_buffer, fill='red', alpha=0.3)
```



Buffering + Union

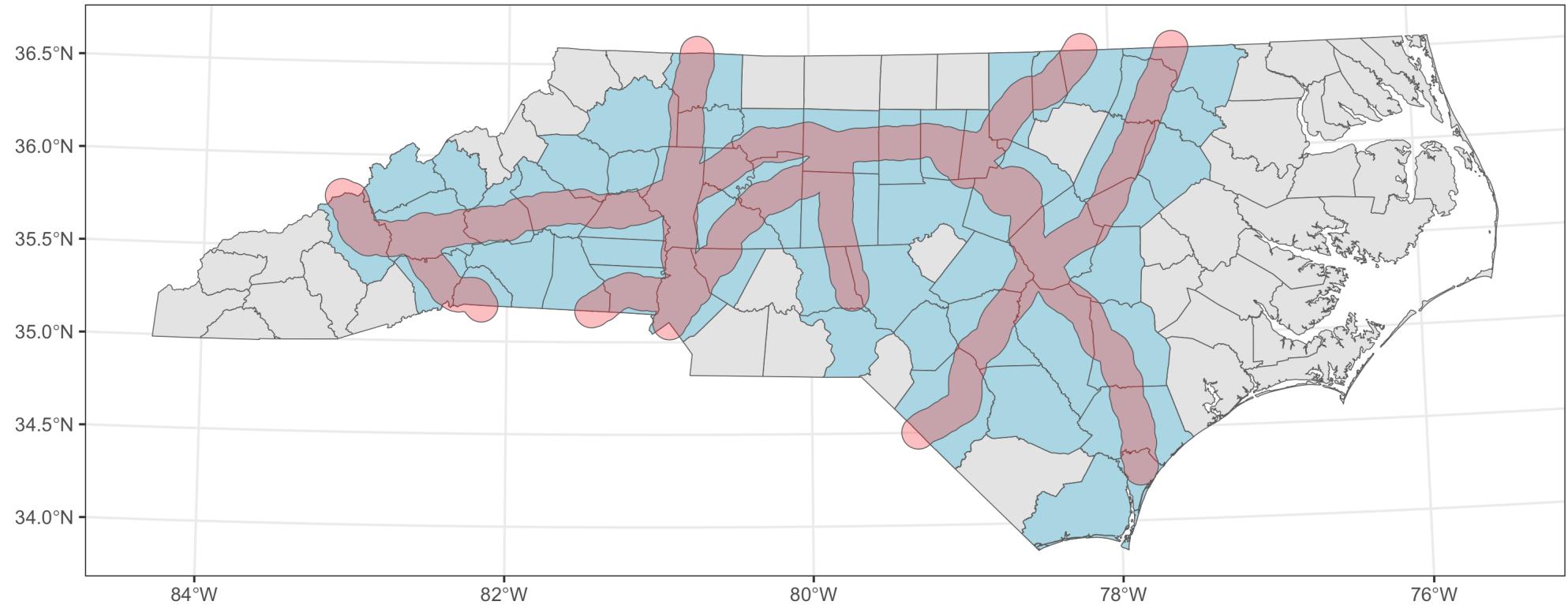
```
1 hwy_nc_buffer = hwy_nc |>
2   st_buffer(10000) |>
3   st_union() |>
4   st_sf()
5
6 ggplot() +
7   geom_sf(data=nc_utm) +
8   geom_sf(data=hwy_nc, color='red') +
9   geom_sf(data=hwy_nc_buffer, fill='red', alpha=0.3)
```



Counties near the interstate (Buffering + Union)

```
1 hwy_nc_buffer = hwy_nc |>
2   st_buffer(10000) |>
3   st_union() |>
4   st_sf()
5
6 hwy_cty = nc_utm |>
7   slice(
8     st_intersects(hwy_nc_buffer, nc_utm) |>
9       unlist() |>
10      unique()
11    )
12
13 ggplot() +
14   geom_sf(data=nc_utm) +
15   geom_sf(data=hwy_nc, color='red') +
16   geom_sf(data=hwy_cty, fill='lightblue') +
17   geom_sf(data=hwy_nc_buffer, fill='red', alpha=0.3)
```

Counties near the interstate (Buffering + Union)



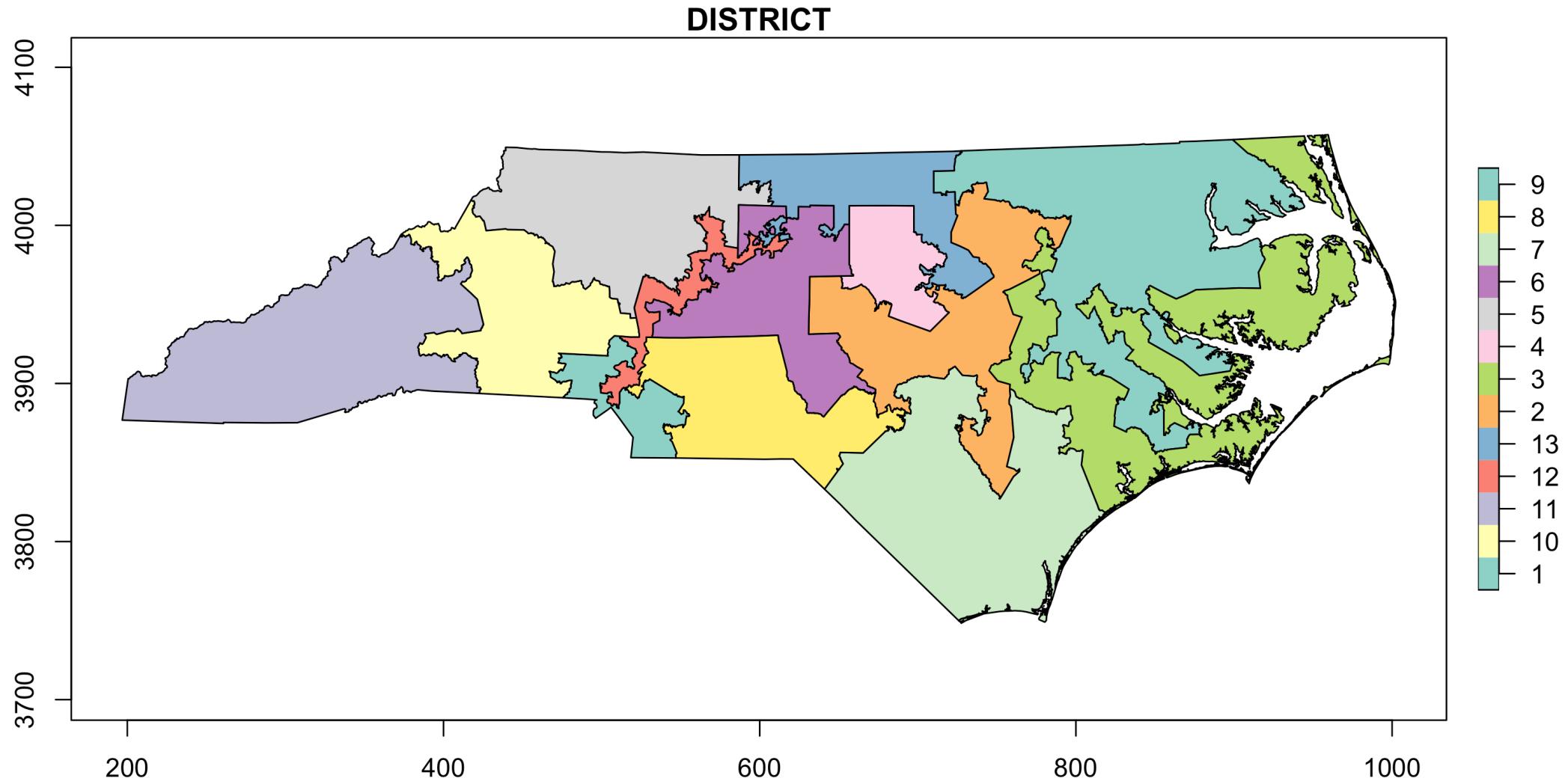
Gerrymandering Example

NC House Districts - 112th Congress

```
1 nc_house = read_sf("data/gis/nc_districts112.gpkg", quiet = TRUE) |>
  2 select(ID, DISTRICT)
```

```
Simple feature collection with 13 features and 2 fields
Geometry type: MULTIPOLYGON
Dimension:      XY
Bounding box:   xmin: -84.32187 ymin: 33.84452 xmax: -75.45998 ymax: 36.58812
Geodetic CRS:   WGS 84
# A tibble: 13 × 3
  ID       DISTRICT                         geom
  <chr>    <chr>                          <MULTIPOLYGON [°]>
1 037108112001 1 ((((-77.32845 35.35031, -77.35398 35.32799, -...
2 037108112002 2 ((((-78.89928 35.12619, -78.89763 35.12859, -...
3 037108112003 3 ((((-75.68266 35.23291, -75.68113 35.23237, -...
4 037108112004 4 ((((-78.77926 35.78568, -78.77947 35.77568, -...
5 037108112005 5 ((((-79.8968 36.38075, -79.89213 36.37108, -7...
6 037108112006 6 ((((-80.4201 35.68953, -80.41483 35.68918, -8...
```

```
1 nc_house = nc_house |>  
2   st_transform("+proj=utm +zone=17 +datum=NAD83 +units=km +no_defs")  
3 plot(nc_house[, "DISTRICT"], axes=TRUE)
```



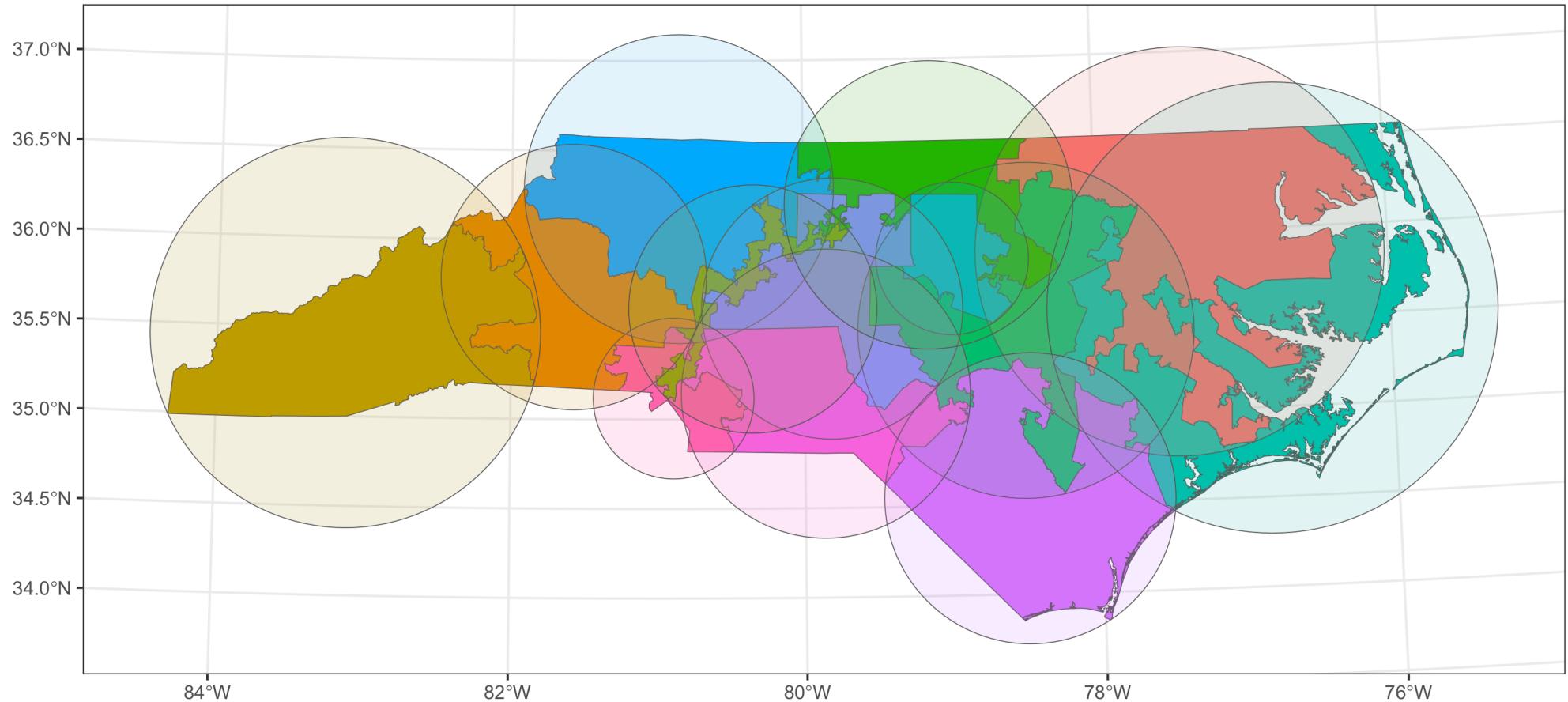
Measuring Compactness - Reock Score

The Reock score is a measure of compactness that is calculated as the ratio of a shape's area to the area of its minimum bounding circle.

```
1 circs = nc_house |>
2   lwgeom::st_minimum_bounding_circle()
3 plot(circs |> filter(DISTRICT == 1) |> st_geometry())
4 plot(nc_house |> select(DISTRICT) |> filter(DISTRICT == 1), add=TRUE)
```

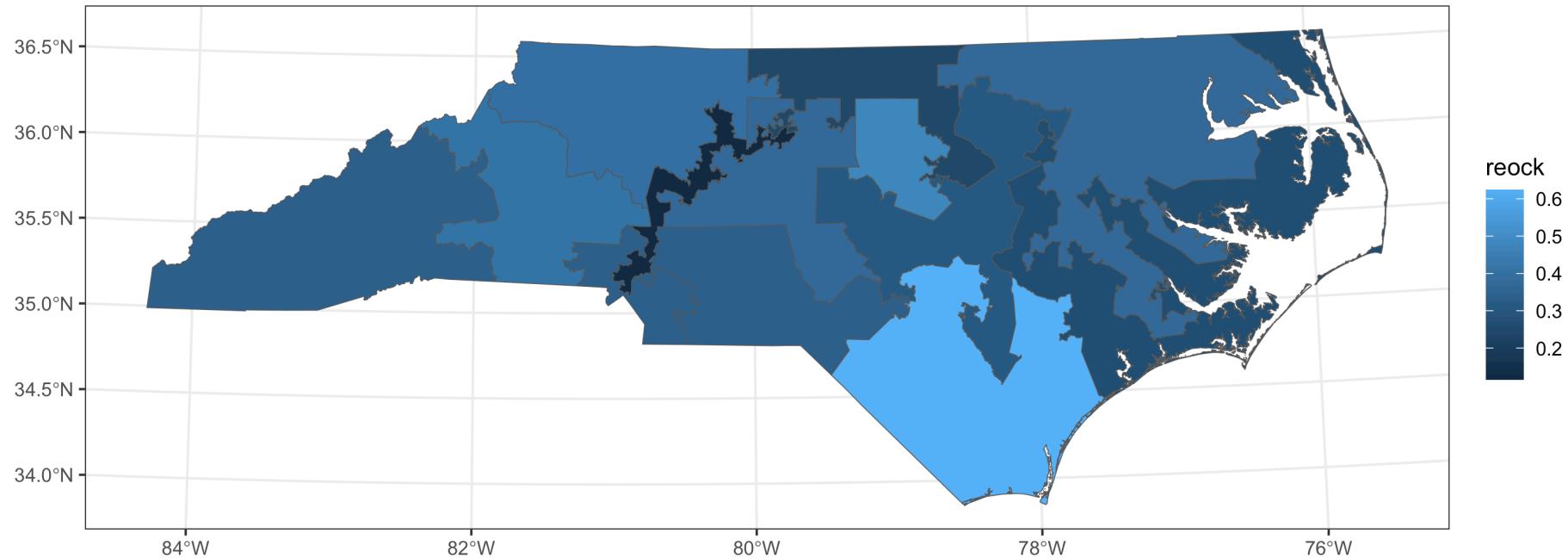


```
1 ggplot(mapping = aes(fill=DISTRICT)) +  
2   geom_sf(data=nc_house) +  
3   geom_sf(data=circs, alpha=0.15) +  
4   guides(color="none", fill="none")
```



Calculating Reock

```
1 nc_house |>  
2   mutate(reock = (st_area(nc_house) / st_area(circs)) |> as.numeric()) |>  
3   ggplot(aes(fill = reock)) +  
4     geom_sf()
```



```
1 nc_house |>
2   mutate(reock = st_area(nc_house) / st_area(circs)) |>
3   arrange(reock) |>
4   as_tibble() |>
5   select(-geom) |>
6   print(n=Inf)
```

```
# A tibble: 13 × 3
  ID      DISTRICT  reock
  <chr>    <chr>     [1]
1 037108112012 12     0.116
2 037108112013 13     0.237
3 037108112003 3      0.266
4 037108112002 2      0.303
5 037108112009 9      0.339
6 037108112008 8      0.342
7 037108112011 11     0.344
8 037108112006 6      0.378
9 037108112001 1      0.378
10 037108112005 5     0.399
11 037108112010 10     0.411
12 037108112004 4     0.480
13 037108112007 7     0.624
```