

# **Lec 22 - Spatial data**

## **Statistical Programming**

**Sta 323 | Spring 2022**

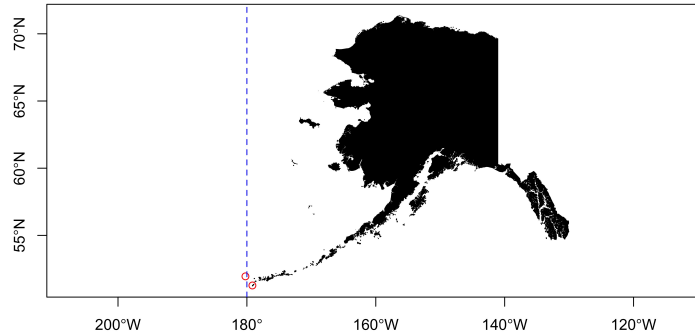
**Dr. Colin Rundel**

**Geospatial stuff is hard**

# Projections

# Dateline

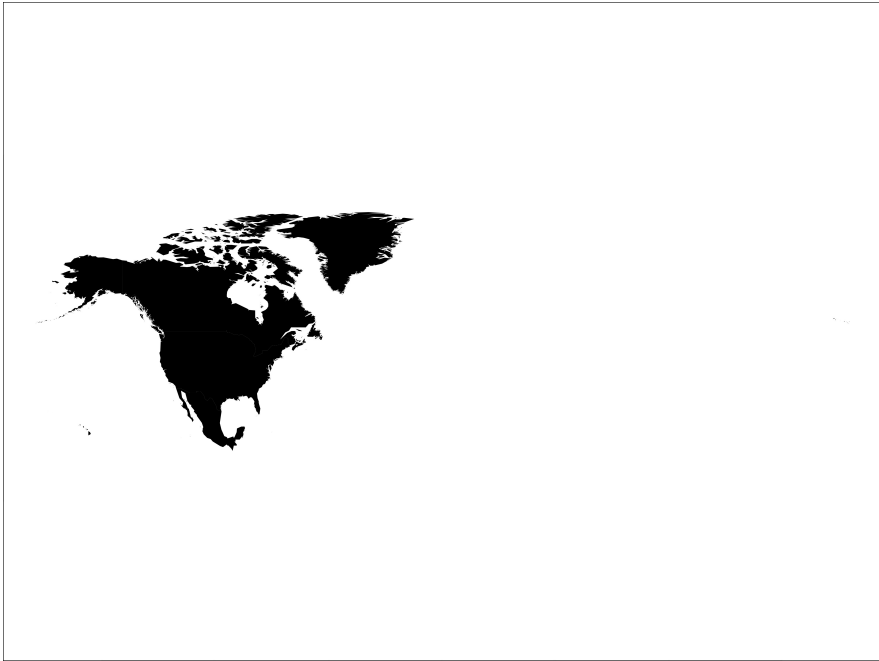
How long is the flight between the Western most and the Eastern most points in the US?



# Great circle distance

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

# Plotting is hard



# 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.
- `stars` - Reading, manipulating, writing and plotting spatiotemporal arrays (rasters)

# Installing `sf`

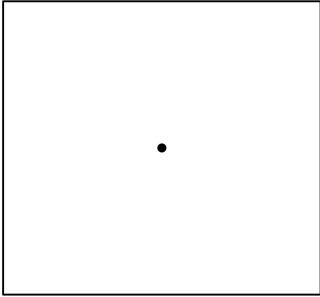
This is the hardest part of using the `sf` package, difficulty comes from its dependence on several external libraries (`geos`, `gdal`, and `proj`).

- 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.

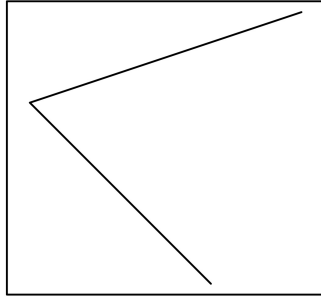
More specific details are included in the repo [README](#) on github.

# Simple Features

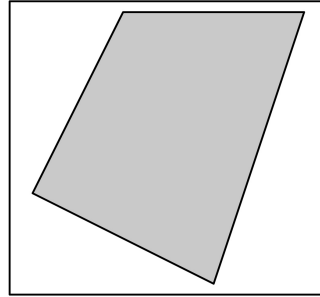
Point



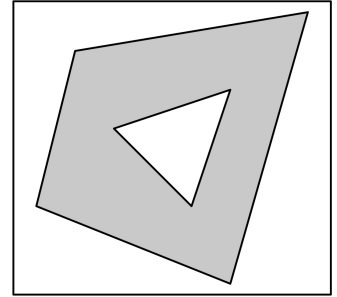
Linestring



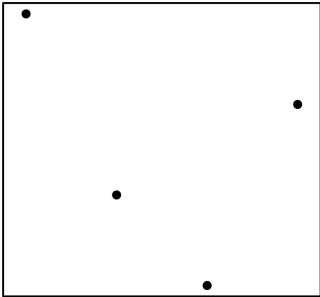
Polygon



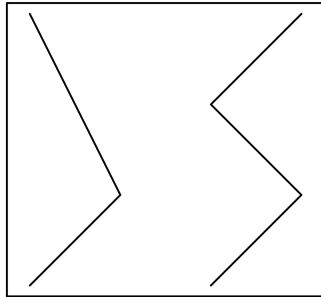
Polygon w/ Hole(s)



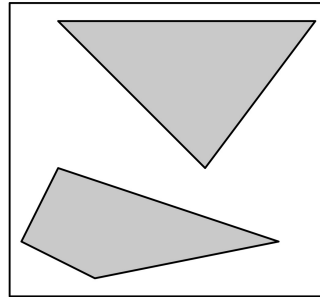
Multipoint



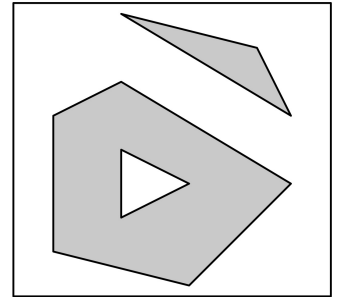
Multilinestring



Multipolygon



Multipolygon w/ Hole(s)



# Reading, writing, and converting simple features

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

# Shapefiles

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

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

# NC Counties

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

```
## Reading layer `nc_counties' from data source
##   `/Users/rundel/Desktop/Sta323-Sp22/website/static/slides/data/gis/nc_counties'
##   using driver `ESRI Shapefile'
## 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

## 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 COUNTYP010 STATE COUNTY FIPS STATE_FIPS
## 1  0.11175964  1.610396      1994    NC    Ashe County 37009      37
## 2  0.06159483  1.354829      1996    NC    Alleghany County 37005      37
## 3  0.14023009  1.769388      1998    NC    Surry County 37171      37
## 4  0.08912401  1.425249      1999    NC    Gates County 37073      37
## 5  0.06865730  4.428217      2000    NC    Currituck County 37053      37
## 6  0.11859434  1.404309      2001    NC    Stokes County 37169      37
```

# sf tibbles

```
(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 Coun... 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 Coun... 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 Co... 37131 37 551.
## 10 0.0925 1.81 2005 NC Hertford County 37091 37 356.
```

```
## # ... with 90 more rows, and 1 more variable: geometry <MULTIPOLYGON [°]>
```

# sf classes

```
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" ...
```

```
class(nc)
```

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

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

```
## [1] "sfc_MULTIPOLYGON" "sfc"
```

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

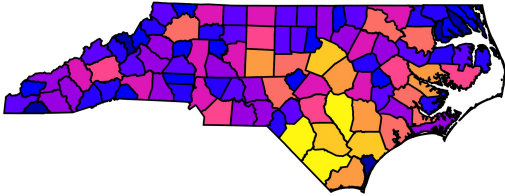
```
## [1] "XY"          "MULTIPOLYGON" "sfg"
```



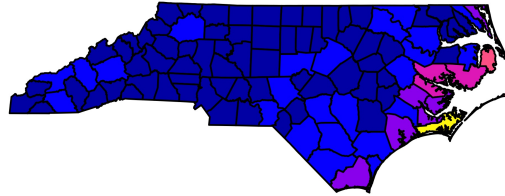
# Plotting

```
plot(nc)
```

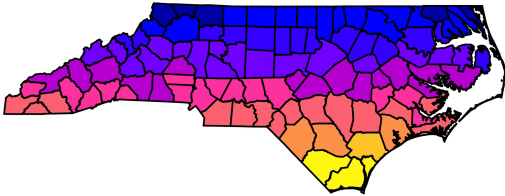
AREA



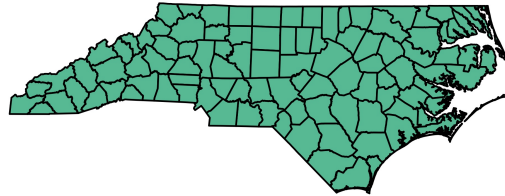
PERIMETER



COUNTYP010



STATE



COUNTY



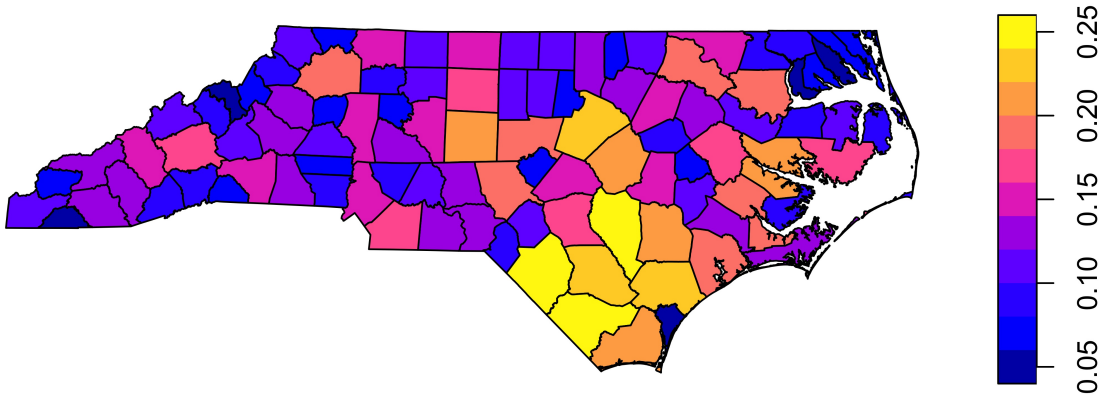
FIPS



# More Plotting

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

AREA



# Graticules

```
par(oma=c(0,2,0,0))  
plot(nc["AREA"], graticule=TRUE, axes=TRUE, las=1)
```

# Geometries

```
par(oma=c(0,2,0,0))  
plot(st_geometry(nc), graticule=TRUE, axes=TRUE, las=1)
```

# ggplot2

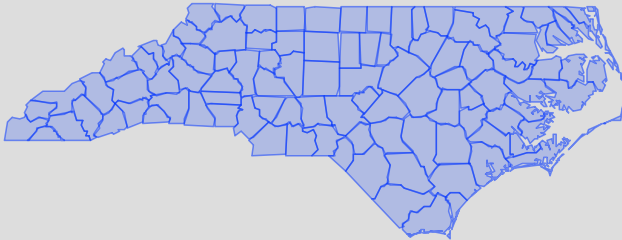
```
ggplot(nc, aes(fill=AREA)) +  
  geom_sf()
```

# ggplot2 + palettes

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

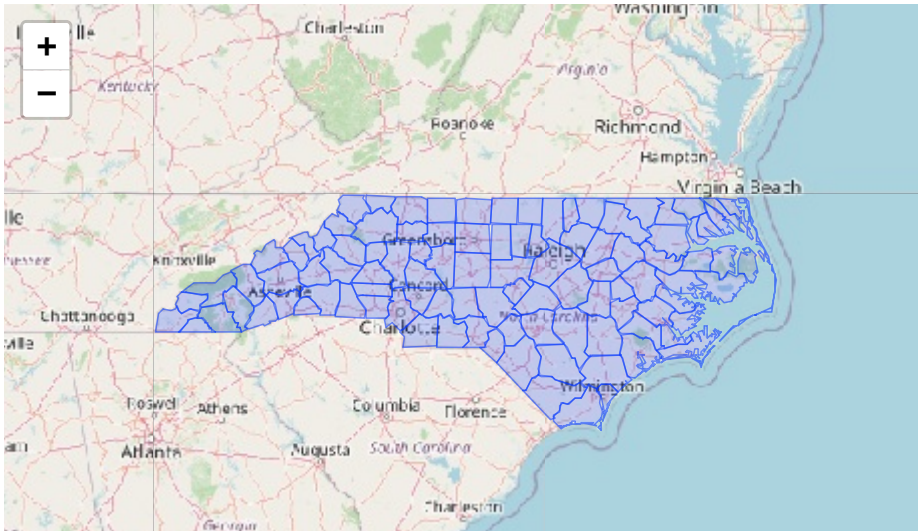
# leaflet

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



# leaflet + tiles

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

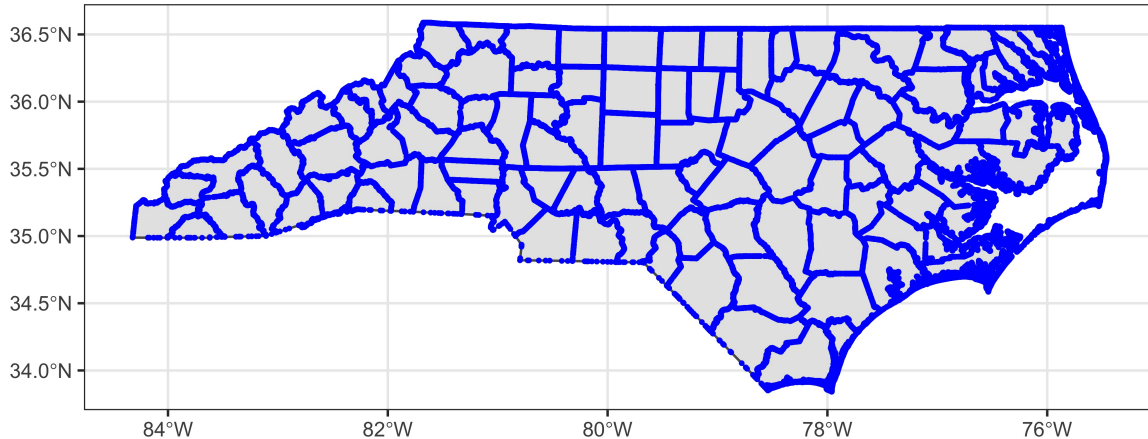




# GIS in R

# Geometry casting

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



# Grouping

```
(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...
```

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

# sf & dplyr

```
nc_cut = nc %>%  
  mutate(  
    ctr_x = st_centroid(nc) %>% st_coordinates() %>% .[,1],  
    region = cut(ctr_x, breaks = 5)  
  )
```

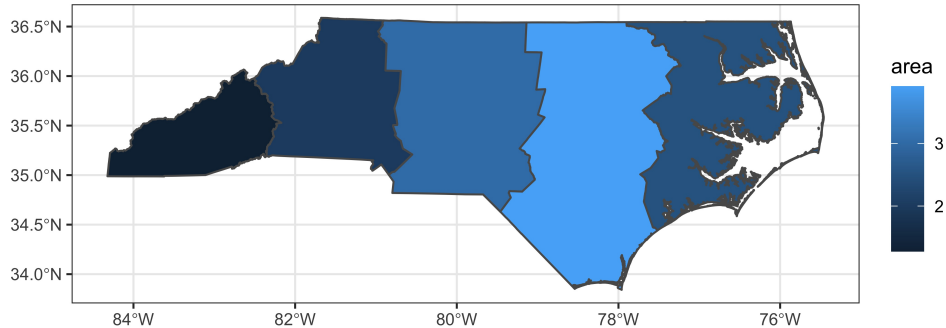
```
## Warning in st_centroid.sf(nc): st_centroid assumes attributes are constant over  
## geometries of x
```

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

# sf & dplyr (cont.)

```
nc_cut2 = nc_cut %>%  
  group_by(region) %>%  
  summarize(  
    area = sum(AREA)  
  )
```

```
ggplot() + geom_sf(data=nc_cut2, aes(fill=area))
```



# Affine Transformations

```
rotate = function(a) matrix(c(cos(a), sin(a), -sin(a), cos(a)), 2, 2)
```

```
(ggplot() + geom_sf(data=(nc_state) * rotate(-pi/4))) +  
(ggplot() + geom_sf(data=(nc_state) * rotate(pi/6)))
```

# Scaling + Translations

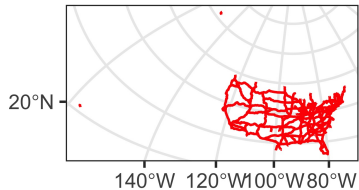
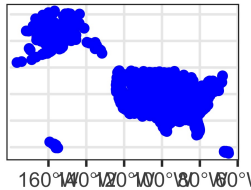
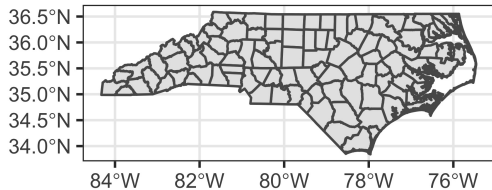
```
ctrd = st_centroid(st_geometry(nc))  
nc_scaled = (st_geometry(nc) - ctrd) * 0.66 + ctrd  
  
ggplot() + geom_sf(data=nc_scaled)
```



# Some other data

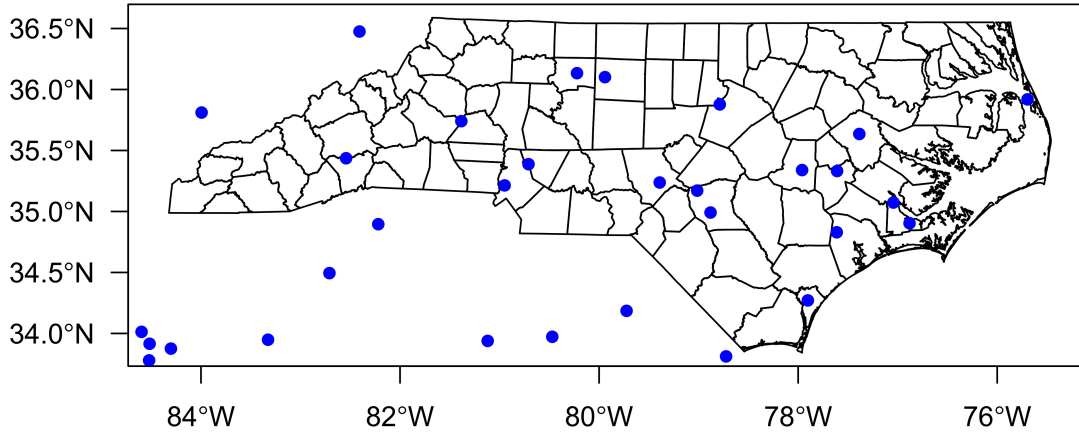
```
air = read_sf("data/gis/airports/", quiet=TRUE)
hwy = read_sf("data/gis/us_interstates/", quiet=TRUE)
```

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

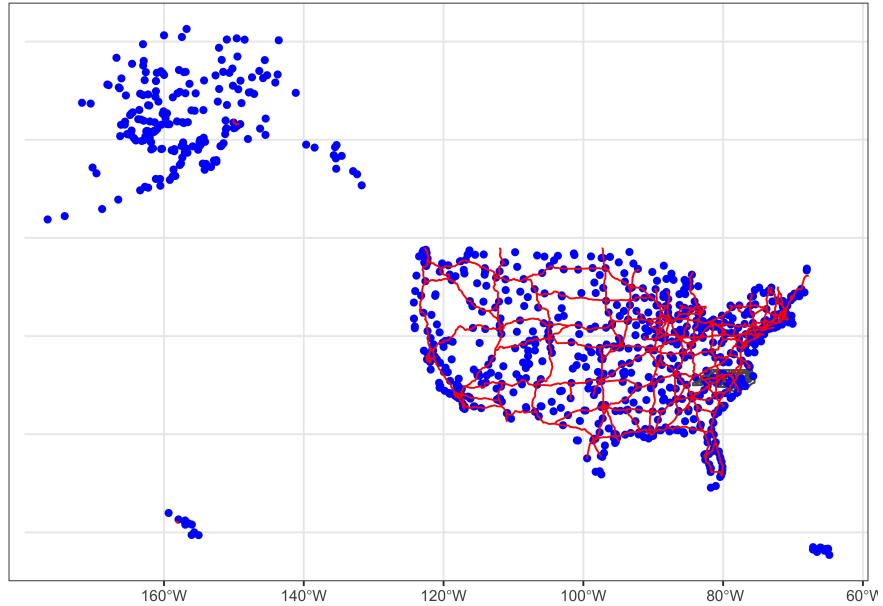




# Overlays?



# Overlays? (ggplot)



# Projections

```
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],
##     ANGLEUNIT["degree",0.0174532925199433]],
##   ID["EPSG",4269]]
```

```
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]]],
```

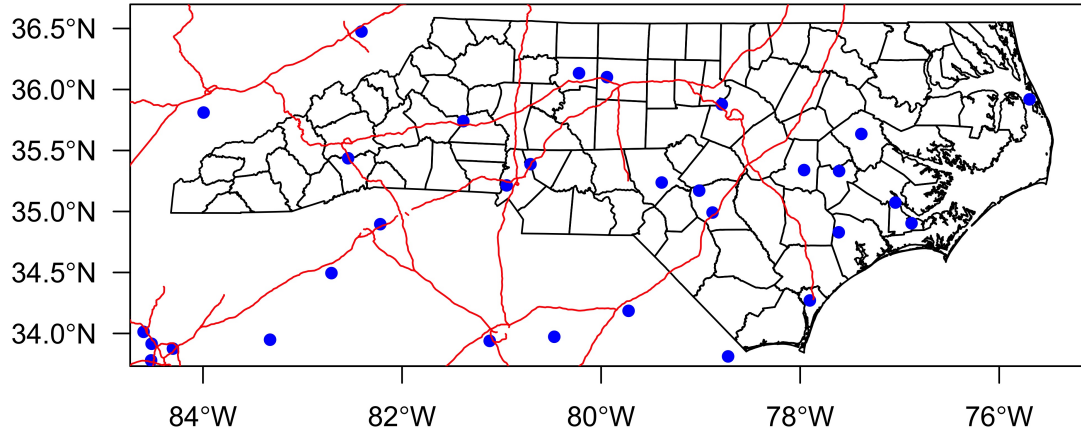
```
st_crs(air)
```

```
## 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],
##     ANGLEUNIT["degree",0.0174532925199433]],
##   ID["EPSG",4269]]
```

# Aside - UTM Zones

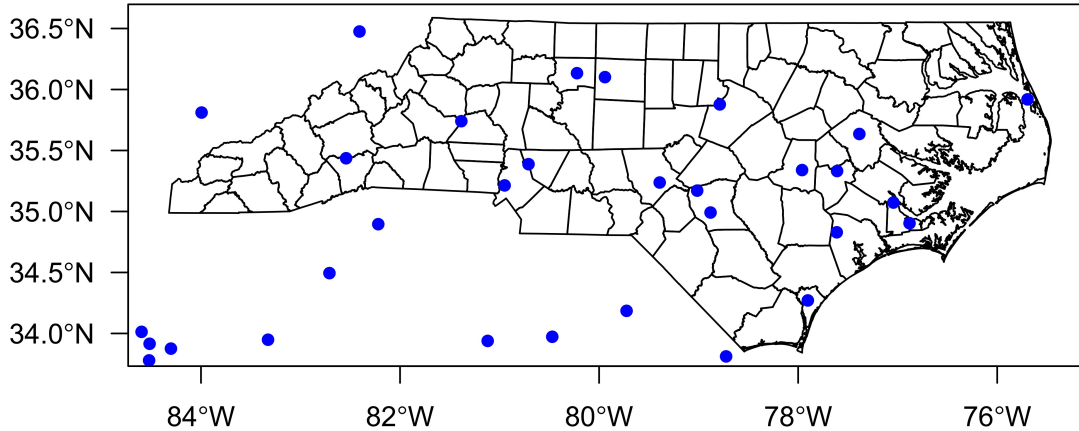
# Lat/Long

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



# Airport Example

# NC Airports



# Sparse Insections

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

```
## List of 11
## $ : int(0)
## $ : int(0)
## $ : int(0)
## $ : int(0)
## $ : int(0)
## $ : 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
## - attr(*, "class")= chr [1:2] "sgbp" "list"
```



# Dense Insections

```
st_intersects(nc, air, sparse=FALSE) %>% str()
```

```
## logi [1:100, 1:940] FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
st_intersects(nc, air, sparse=FALSE) %>% .[20:30, 260:270]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
## [1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [3,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [4,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [5,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [6,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
## [7,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [8,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [9,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [10,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [11,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

# Which counties have airports?

```
nc_air = nc %>%  
  mutate(  
    n_air = map_int(st_intersects(nc, air), length) %>%  
  ) %>%  
  filter(n_air > 0)  
  
nc_air %>% pull(COUNTY)
```

```
## [1] "Forsyth County"      "Guilford County"  
## [4] "Wake County"         "Pitt County"  
## [7] "Buncombe County"    "Wayne County"  
## [10] "Moore County"       "Cabarrus County"  
## [13] "Craven County"      "Cumberland County"  
## [16] "New Hanover County"
```

```
air_nc = air %>%  
  slice(  
    st_intersects(nc, air) %>% unlist() %>% unique  
  )  
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] "SEYMOUR 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"  
## [14] "COASTAL CAROLINA REGIONAL AIRPORT"  
## [15] "POPE AIR FORCE BASE"  
## [16] "FAYETTEVILLE REGIONAL/GRANNIS FIELD" 42 / 50
```

# Results

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

# Highway Example

# Highways

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

# NC Interstate Highways

```
hwy_nc = st_intersection(hwy, nc)
```

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

```
ggplot() +  
  geom_sf(data=nc) +  
  geom_sf(data=hwy_nc, col='red')
```

# Counties near the interstate (Projection)

```
nc_utm = st_transform(nc, "+proj=utm +zone=17 +datum=NAD83 +units=m +no_defs")
hwy_utm = st_transform(hwy, "+proj=utm +zone=17 +datum=NAD83 +units=m +no_defs")

hwy_nc = st_intersection(hwy_utm, nc_utm)
```

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

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

# Counties near the interstate (Buffering)

```
hwy_nc_buffer = hwy_nc %>%  
  st_buffer(10000)  
  
ggplot() +  
  geom_sf(data=nc_utm) +  
  geom_sf(data=hwy_nc, color='red') +  
  geom_sf(data=hwy_nc_buffer, fill='red', alpha=0.3)
```



# Counties near the interstate (Buffering + Union)

```
hwy_nc_buffer = hwy_nc %>%  
  st_buffer(10000) %>%  
  st_union() %>%  
  st_sf()  
  
ggplot() +  
  geom_sf(data=nc_utm) +  
  geom_sf(data=hwy_nc, color='red') +  
  geom_sf(data=hwy_nc_buffer, fill='red', alpha=0.3)
```

# Example

How many counties in North Carolina are within 5, 10, 20, or 50 km of an interstate highway?