R Spatial Basics Agenda

Follow-along example with NJ Bus Stop data

1. Loading data from file
    a. CSV format
    b. Shapefile format
2. Geographic data basics
3. Basic maps
4. Gridded aggregations

TACC
TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

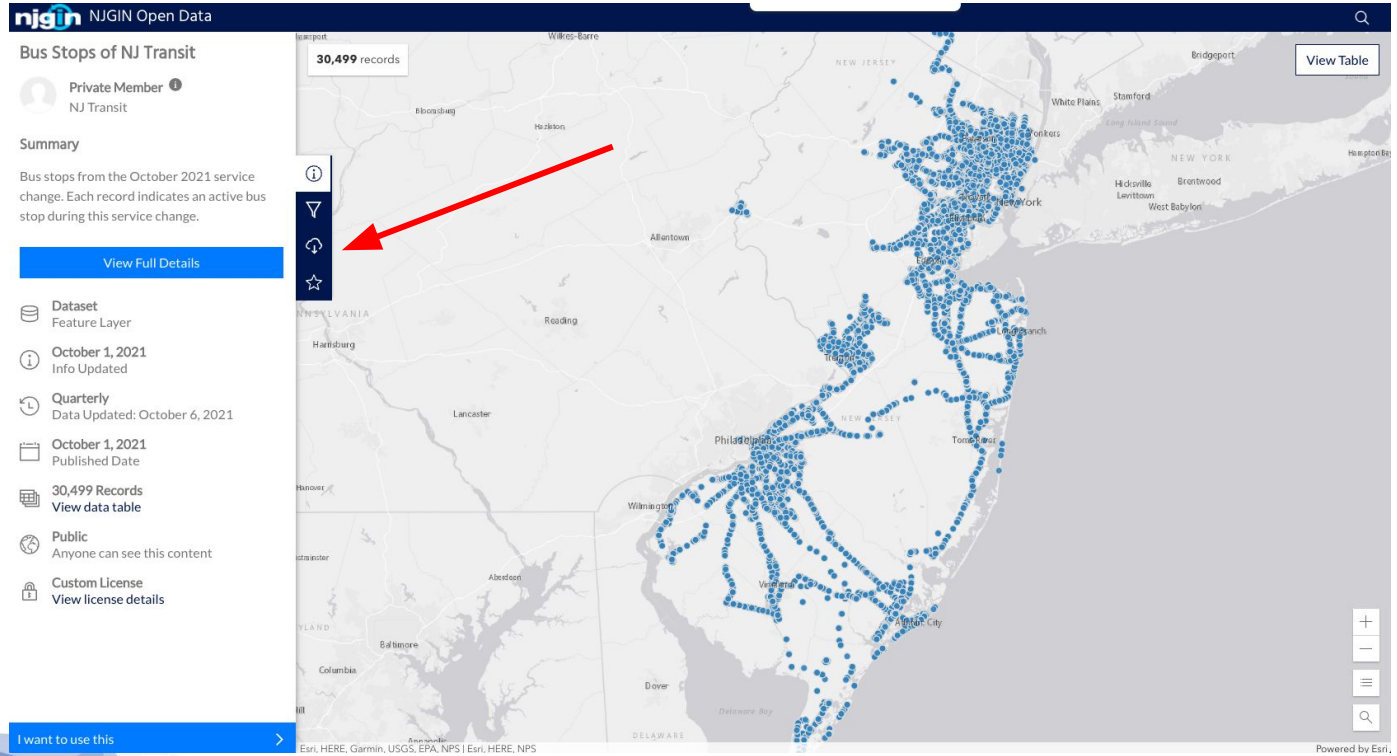
TEXAS
The University of Texas at Austin

R Spatial Basics

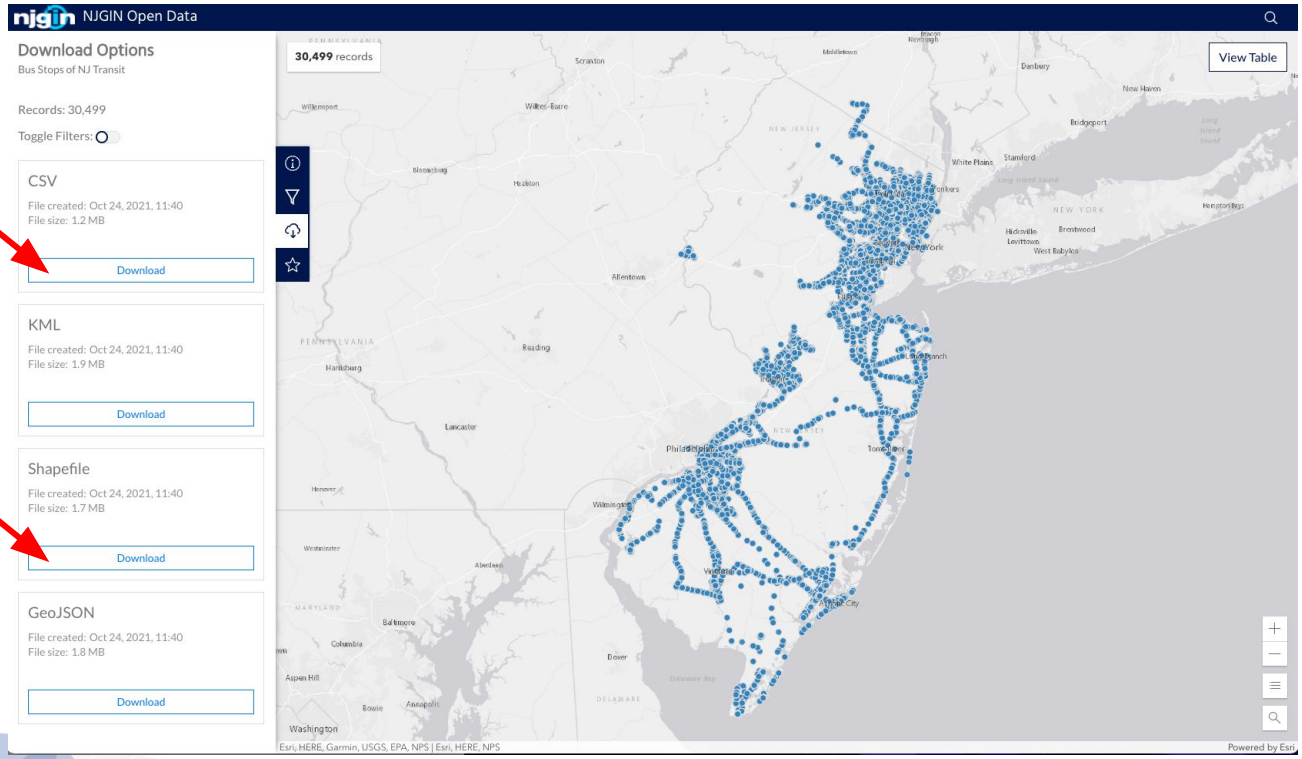
Kelly Pierce, Scalable Computational Intelligence
kpierce@tacc.utexas.edu

With thanks to David Walling and Chris Ramos for contributing materials to these lessons

Data download



Download Shapefile and CSV file



Geographic data basics

- Geographic coordinate systems (GCS): where on the ellipsoid did this happen?
- Projected coordinate systems (PCS) are flattened-out
- R handles projection math for us

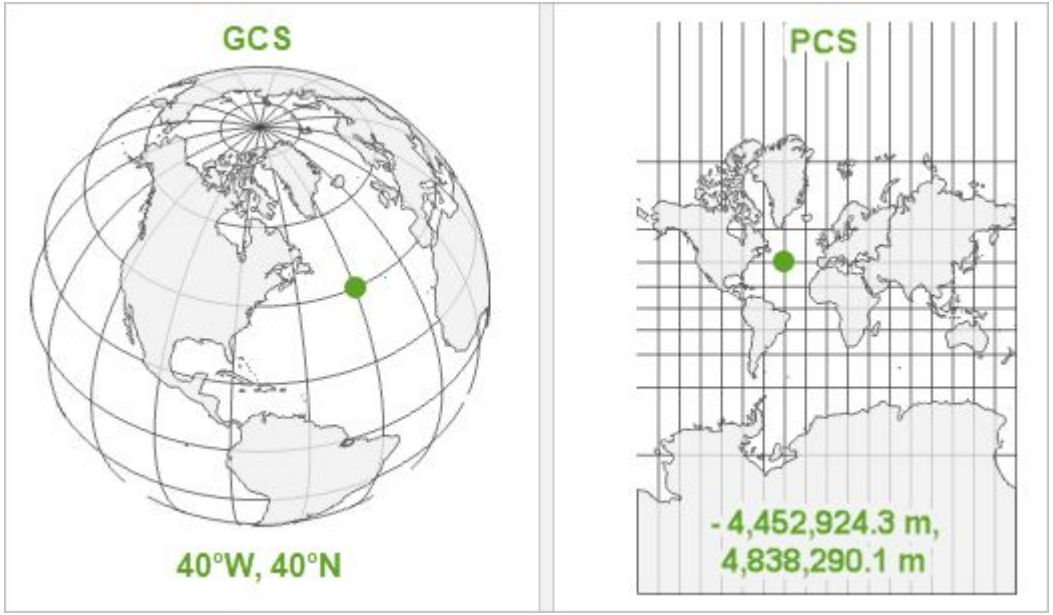


image source: https://www.esri.com/arcgis-blog/products/arcgis-pro/mapping/gcs_vs_pcs/

Loading data from file

- comma-separated values
  - single file in text format
  - columns are separated by commas
  - rows are separated by newlines

```
column1,column2
value1,value2
```

- shapefiles
  - collection of files
  - contains spatial metadata

Creating simple features from a CSV

- "Simple features" refers to a spatial data standard used by ArcGIS, the R **`sf`** library, and other geospatial software

```
bus_csv <- read.csv('data/Bus_Stops_of_NJ_Transit.csv')
head(bus_csv)
```

```
##     FID COUNTY DLAT_GIS DLONG_GIS LINE STOP_NUM STOP_TYPE STREET_DIR
## 1 30448  Essex 40.72093 -74.22520   37    17819        NS          N
## 2 30449  Essex 40.72093 -74.22520  107    17819        NS          N
## 3 30450  Essex 40.72115 -74.22518   90    17841        NS          S
## 4 30451  Essex 40.79417 -74.23053   97    19442        NS          N
## 5 30452  Essex 40.79419 -74.23067   97    19445        FS          S
## 6 30453  Essex 40.77684 -74.17063   99    18659        NS          E
```

Creating simple features from a CSV

- After loading the CSV formatted data, we convert to a simple features format with the function **`st_as_sf()`** from library **`sf`**
- We don't know the CRS, so we guess a common one: WGS84 (not recommended in real life)

```
library(sf)
```

```
## Linking to GEOS 3.8.1, GDAL 3.2.1, PROJ 7.2.1; sf_use_s2() is TRUE
```

```
bus_sf <- bus_csv %>% st_as_sf(.,coords=c("DLONG_GIS","DLAT_GIS"), crs=WGS84)
```

Creating simple features from a CSV

metadata

data

```
## Simple feature collection with 6 features and 10 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -74.23067 ymin: 40.72093 xmax: -74.17063 ymax: 40.79419
## Geodetic CRS:  WGS 84
##     FID COUNTY LINE STOP_NUM STOP_TYPE STREET_DIR        DESCRIPTION_BSL
## 1 30448  Essex   37    17819        NS          N GROVE ST AT HERPERS ST
## 2 30449  Essex  107    17819        NS          N GROVE ST AT HERPERS ST
## 3 30450  Essex   90    17841        NS          S GROVE ST AT HERPERS ST
## 4 30451  Essex   97    19442        NS          N HARRISON AVE AT ELM ST
## 5 30452  Essex   97    19445        FS          S HARRISON AVE AT ELM ST
## 6 30453  Essex   99    18659        NS          E HELLER PKWY AT LAKE ST
##   DIRECTION_OP    MUNICIPALITY                              GlobalID
## 1           Ou   IRVINGTON TWP 9dd47d93-3ac8-4167-85bd-20fefa063e33
## 2           Ou   IRVINGTON TWP f3cde248-5286-4a0d-8710-94c38b3bafd8
## 3           Ou   IRVINGTON TWP 8fc8919f-8a25-4eec-aa65-f5c9d6d793dd
## 4           In WEST ORANGE TWP 65a3bde8-d8f0-475d-9f49-ff86a81d387f
## 5           Ou WEST ORANGE TWP 4a3fa65f-a13a-4f07-bfcd-977f7562c5c4
## 6           In          NEWARK 70298ad9-9b33-419d-b883-1d2434264126
##                    geometry
## 1  POINT (-74.2252 40.72093)
## 2  POINT (-74.2252 40.72093)
## 3 POINT (-74.22518 40.72115)
## 4 POINT (-74.23053 40.79417)
## 5 POINT (-74.23067 40.79419)
## 6 POINT (-74.17063 40.77684)
```

Shapefile basics

- Shapefiles are really collections of files

dBASE table file

Main file

Index file

Metadata file

Optional codepage file (identifies character set to use)

Projections definition file

Creating simple features from a shapefile

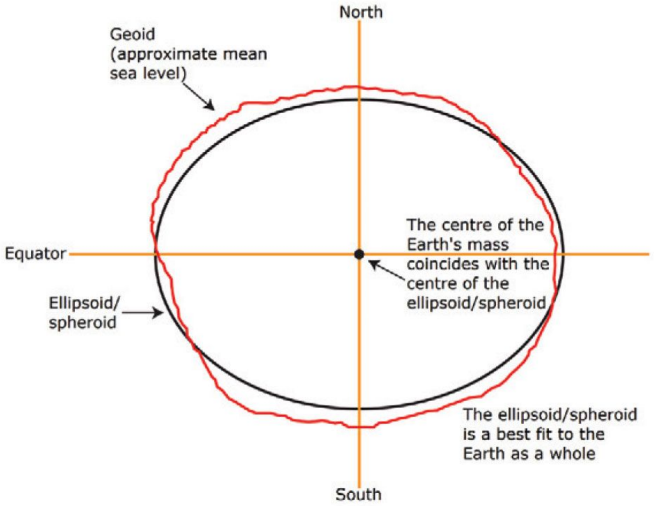
- We can load shapefiles directly with **`st_read()`**
- The CRS will be read from the shapefile **`.prj`** file

```
bus <- st_read('data/Bus_Stops_of_NJ_Transit.shp')
```

```
## Reading layer 'Bus_Stops_of_NJ_Transit' from data source
##   '/Users/kpierce/data_trainings/PAP-TACC-2022/data/Bus_Stops_of_NJ_Transit.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 30499 features and 12 fields
## Geometry type: MULTIPOINT
## Dimension:     XY
## Bounding box:  xmin: -8406962 ymin: 4712207 xmax: -8230860 ymax: 5050000
## Projected CRS: WGS 84 / Pseudo-Mercator
```

Reference frames

Geocentric (WGS84)



Regional (New Jersey state planar)

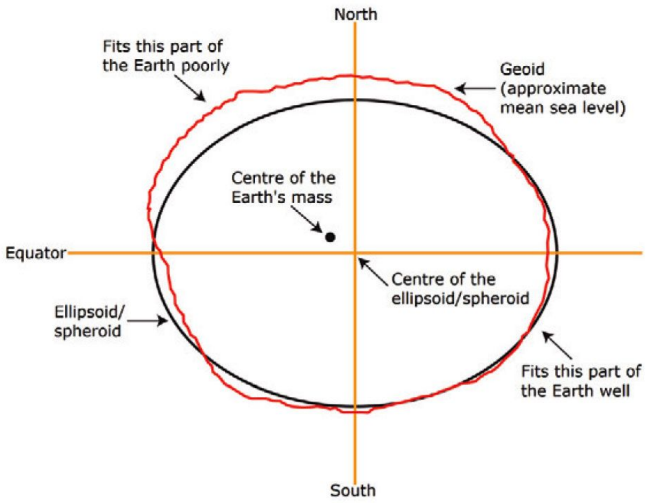


image source:
https://csegrecorder.com/articles/view/datums-projections-and-coordinate-systems

Change coordinate reference systems

- Coordinate reference systems (both geographic and projected) are described by codes
- ESRI maintains a database of codes at https://spatialreference.org/ref/esri/
- We'll define two CRS codes as variables so we can reference them by name, making our script easier-to-read

```
WGS84 = 4326
NJ_PLANAR = 'ESRI:102311'
```

- WGS84 is our geocentric geographic reference system
- NJ_PLANAR is our regional projected reference system

Change coordinate reference systems

```
bus_wgs84 <- st_transform(bus, crs=WGS84)
```

```
head(bus_wgs84)
```

```
## Simple feature collection with 6 features and 12 fields
## Geometry type: MULTIPOINT
## Dimension:     XY
## Bounding box:  xmin: -74.23071 ymin: 40.72092 xmax: -74.17066 ymax: 40.79424
## Geodetic CRS:  WGS 84
```

Change coordinate reference systems

```
bus_nj <- st_transform(bus, crs=NJ_PLANAR)
```

```
head(bus_nj)
```

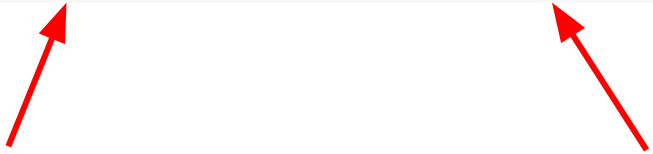
```
## Simple feature collection with 6 features and 12 fields
## Geometry type: MULTIPOINT
## Dimension:     XY
## Bounding box:  xmin: 172725.2 ymin: 209594.1 xmax: 177799.7 ymax: 217734.3
## Projected CRS: NAD_1983_HARN_StatePlane_New_Jersey_FIPS_2900
```

Map the bus stops

- Point locations
- Basemaps
- Choropleth maps

Mapping point locations

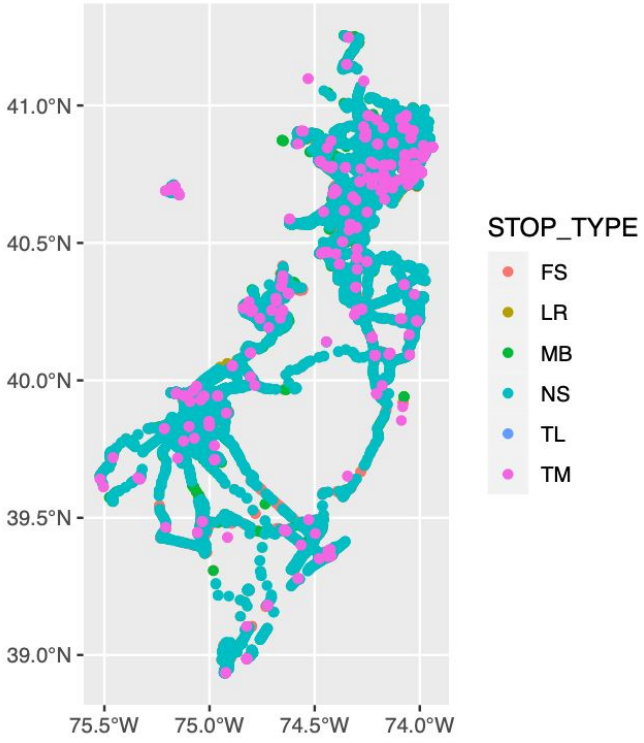
```
nj_bus_map <- ggplot() +
  geom_sf(data=bus_wgs84, aes(color=STOP_TYPE), inherit.aes = FALSE)
```



Use the unprojected WGS84 data; **`geom_sf()`** will project the data onto the map plane

color the points by the value in column **`STOP_TYPE`**

Mapping point locations



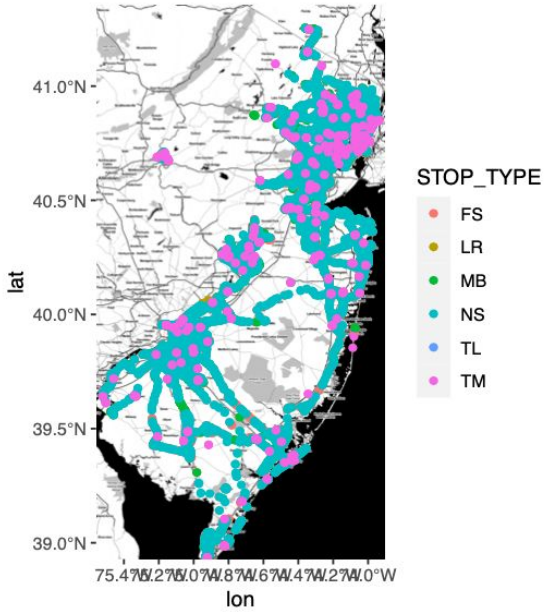
Adding map layers

- Our map needs a background to provide context
- Open Street Maps provides free map tiles
  - download tiles corresponding to bounding box
  - different tile themes available (http://maps.stamen.com/)
- Libraries **`ggplot`** and **`ggmap`** allow us to build layered maps in R

```
nj <- read_sf(
  'https://opendata.arcgis.com/datasets/5f45e1ece6e14ef5866974a7b57d3b95_1.geojson'
  )
nj <- nj %>% st_transform(crs=WGS84)
nj_bbox <- unname(st_bbox(nj))
nj_base_map <- get_stamenmap(bbox=nj_bbox, maptype="toner", force=TRUE)
```

Adding map layers

```
library(ggmap)
nj_bus_map <- ggmap(nj_base_map) +
  geom_sf(data=bus_wgs84, aes(color=STOP_TYPE), inherit.aes = FALSE)
```



Choropleth map of 2010 population

The state boundary map we used for the bounding box contained some additional data:

```
## Simple feature collection with 6 features and 22 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -75.41973 ymin: 38.92852 xmax: -73.90245 ymax: 41.13372
## Geodetic CRS:  WGS 84
## # A tibble: 6 x 23
##   OBJECTID COUNTY     COUNTY_LABEL  CO    GNIS_NAME GNIS  FIPSSTCO FIPSCO  ACRES
##      <int> <chr>      <chr>         <chr> <chr>     <chr> <chr>    <chr>   <dbl>
## 1        1 ATLANTIC   Atlantic Cou~ ATL   County o~ 8822~ 34001    1      3.91e5
## 2        2 BERGEN     Bergen County BER   County o~ 8822~ 34003    3      1.53e5
## 3        3 BURLINGTON Burlington C~ BUR   County o~ 8822~ 34005    5      5.25e5
## 4        4 CAMDEN     Camden County CAM   County o~ 8822~ 34007    7      1.46e5
## 5        5 CAPE MAY   Cape May Cou~ CAP   County o~ 8822~ 34009    9      1.83e5
## 6        6 CUMBERLAND Cumberland C~ CUM   County o~ 8822~ 34011    11     3.21e5
## # ... with 14 more variables: SQ_MILES <dbl>, POP2010 <int>, POP2000 <int>,
## #   POP1990 <int>, POP1980 <int>, POPDEN2010 <int>, POPDEN2000 <int>,
## #   POPDEN1990 <int>, POPDEN1980 <int>, REGION <chr>, GLOBALID <chr>,
## #   Shape_Length <dbl>, Shape_Area <dbl>, geometry <MULTIPOLYGON [°]>
```

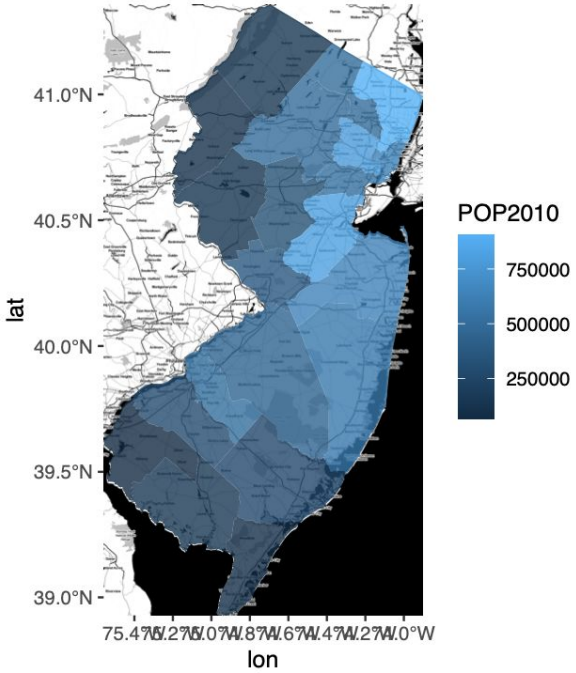
Choropleth map of 2010 population

The state boundary map we used for the bounding box contained some additional data:

```
## Simple feature collection with 6 features and 22 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -75.41973 ymin: 38.92852 xmax: -73.90245 ymax: 41.13372
## Geodetic CRS:  WGS 84
## # A tibble: 6 x 23
##   OBJECTID COUNTY     COUNTY_LABEL  CO    GNIS_NAME GNIS  FIPSSTCO FIPSCO  ACRES
##      <int> <chr>      <chr>         <chr> <chr>     <chr> <chr>    <chr>   <dbl>
## 1        1 ATLANTIC   Atlantic Cou~ ATL   County o~ 8822~ 34001    1      3.91e5
## 2        2 BERGEN     Bergen County BER   County o~ 8822~ 34003    3      1.53e5
## 3        3 BURLINGTON Burlington C~ BUR   County o~ 8822~ 34005    5      5.25e5
## 4        4 CAMDEN     Camden County CAM   County o~ 8822~ 34007    7      1.46e5
## 5        5 CAPE MAY   Cape May Cou~ CAP   County o~ 8822~ 34009    9      1.83e5
## 6        6 CUMBERLAND Cumberland C~ CUM   County o~ 8822~ 34011    11     3.21e5
## # ... with 14 more variables: SQ_MILES <dbl>, POP2010 <int>, POP2000 <int>,
## #   POP1990 <int>, POP1980 <int>, POPDEN2010 <int>, POPDEN2000 <int>,
## #   POPDEN1990 <int>, POPDEN1980 <int>, REGION <chr>, GLOBALID <chr>,
## #   Shape_Length <dbl>, Shape_Area <dbl>, geometry <MULTIPOLYGON [°]>
```

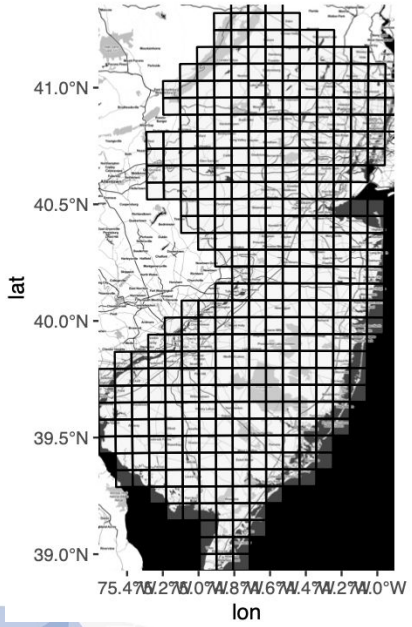
Choropleth map of 2010 population

```
nj_map <- ggmap(nj_base_map) +
  geom_sf(data=nj, aes(fill=POP2010), inherit.aes = FALSE, color = NA, alpha = 0.8)
```

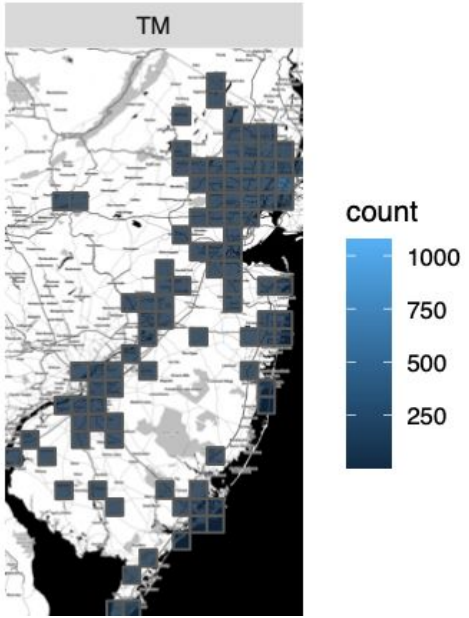


Gridded (“fishnet”) mappings

Overlay grid



Aggregate points into grid cells



Define spatial extent for 5-mi grid

- project the NJ border into the regional planar CRS
- convert 5-mile grid to meters (1,609m/mi)

```
nj_flat <- st_transform(nj, crs=NJ_PLANAR)
fishnet_width <- 1609 * 5
```

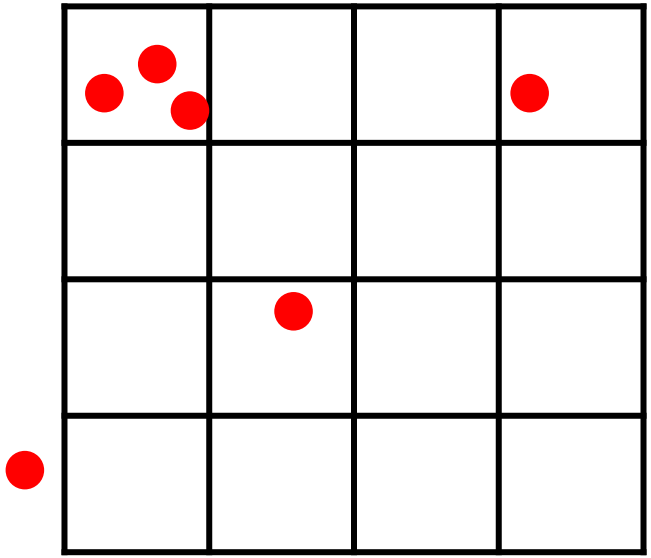
- remove interior county borders from shape

```
nj_border <- st_make_valid(st_union(nj_flat))
```

Create grid and convert to `sf`

```
net <- st_make_grid(
  x=nj_border, cellsize=fishnet_width, what='polygons', square=TRUE, crs=NJ_PLANAR
  )
net_agg <- st_as_sf(net) %>% tibble::rowid_to_column(., "net_id")
net_intersect <- st_intersects(nj_border, net_agg)
fishnet <- net_agg[unique(unlist(net_intersect)),]
```

Point-in-polygon join



Point-in-polygon join

```
bus_net <- st_join(bus_nj, fishnet, join=st_within) %>%
    st_drop_geometry()
```

- point data is left argument
- polygon data is right argument
- polygon geometry is implicitly dropped by st_join, and only point geometry is retained
- we intentionally drop point geometry to retain only the bus stop ID data and the fishnet ID data

Finalize bus stop fishnet

Aggregate on `net_id`

```
bus_net_summary <- bus_net %>%
  group_by(net_id, STOP_TYPE) %>%
  summarise(count=n())
```

Rejoin fishnet geometry

```
bus_net_final <- st_as_sf(
  left_join(
    fishnet,
    bus_net_summary,
    by='net_id'
    ),
  crs=st_crs(fishnet))
```

Map bus stop counts by type

```
bus_type_map <- ggmap(nj_base_map) +
  geom_sf(
    data=st_transform(bus_net_final, crs=WGS84),
    aes(fill=count),
    inherit.aes = FALSE, alpha=0.8
    ) +
  facet_wrap(facets='STOP_TYPE', nrow=3)
bus_type_map
```