

Lecture 15

GPs for GLMs + Spatial Data

3/20/2018

GPs and GLMs

Logistic Regression

A typical logistic regression problem uses the following model,

$$y_i \sim \text{Bern}(p_i)$$

$$\begin{aligned}\text{logit}(p_i) &= \mathbf{X} \boldsymbol{\beta} \\ &= \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}\end{aligned}$$

Logistic Regression

A typical logistic regression problem uses the following model,

$$y_i \sim \text{Bern}(p_i)$$

$$\begin{aligned}\text{logit}(p_i) &= \mathbf{X} \boldsymbol{\beta} \\ &= \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}\end{aligned}$$

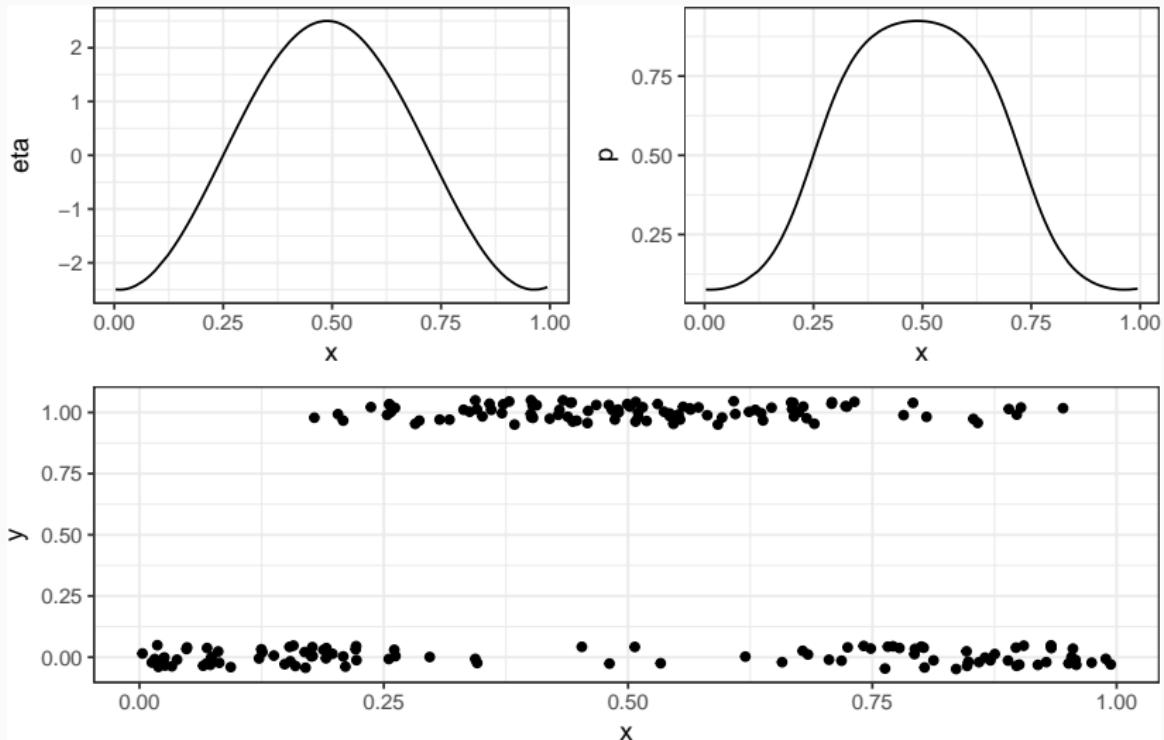
there is no reason that the linear equation above can't contain things like random effects or GPs

$$\begin{aligned}y_i &\sim \text{Bern}(p_i) \\ \text{logit}(p_i) &= \mathbf{X} \boldsymbol{\beta} + w(\mathbf{x})\end{aligned}$$

where

$$w(\mathbf{x}) \sim \mathcal{N}(0, \Sigma)$$

A toy example



Jags Model*

```
logistic_model = "model{
  for(i in 1:N) {
    y[i] ~ dbern(p[i])
    logit(p[i]) = beta0 + eta[i]

    y_hat[i] ~ dbern(p[i])
    loglik_i[i] = y[i] * log(p[i]) + (1-y[i]) * log(1-p[i])
  }
  loglik = sum(loglik_i)

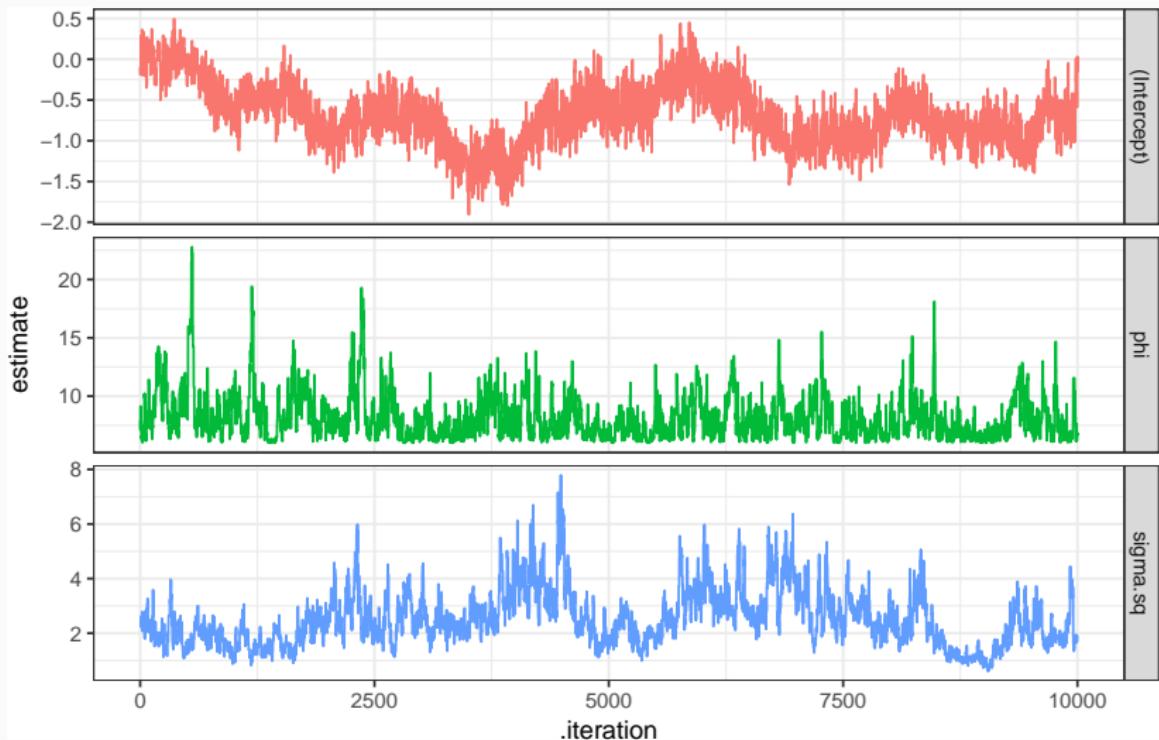
  eta ~ dmnorm(rep(0,N), inverse(Sigma))

  for (i in 1:(length(y)-1)) {
    for (j in (i+1):length(y)) {
      Sigma[i,j] = sigma2 * exp(- l * d[i,j]))
      Sigma[j,i] = Sigma[i,j]
    }
  }

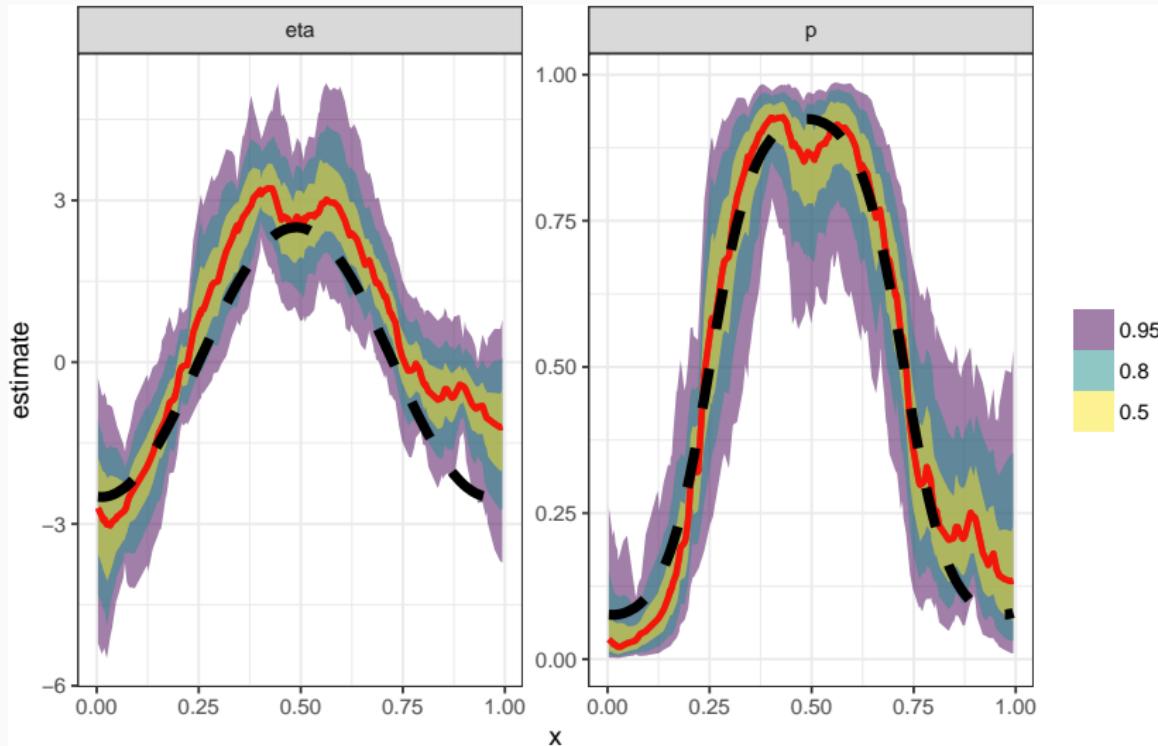
  for (i in 1:length(y)) {
    Sigma[i,i] = sigma2
  }

  beta0 ~ dnorm(0, 1)
  sigma2 = 1/tau
  tau ~ dgamma(1, 2)
  l ~ dunif(3/0.5, 3/0.01)
}"
```

Model Results - Diagnostics

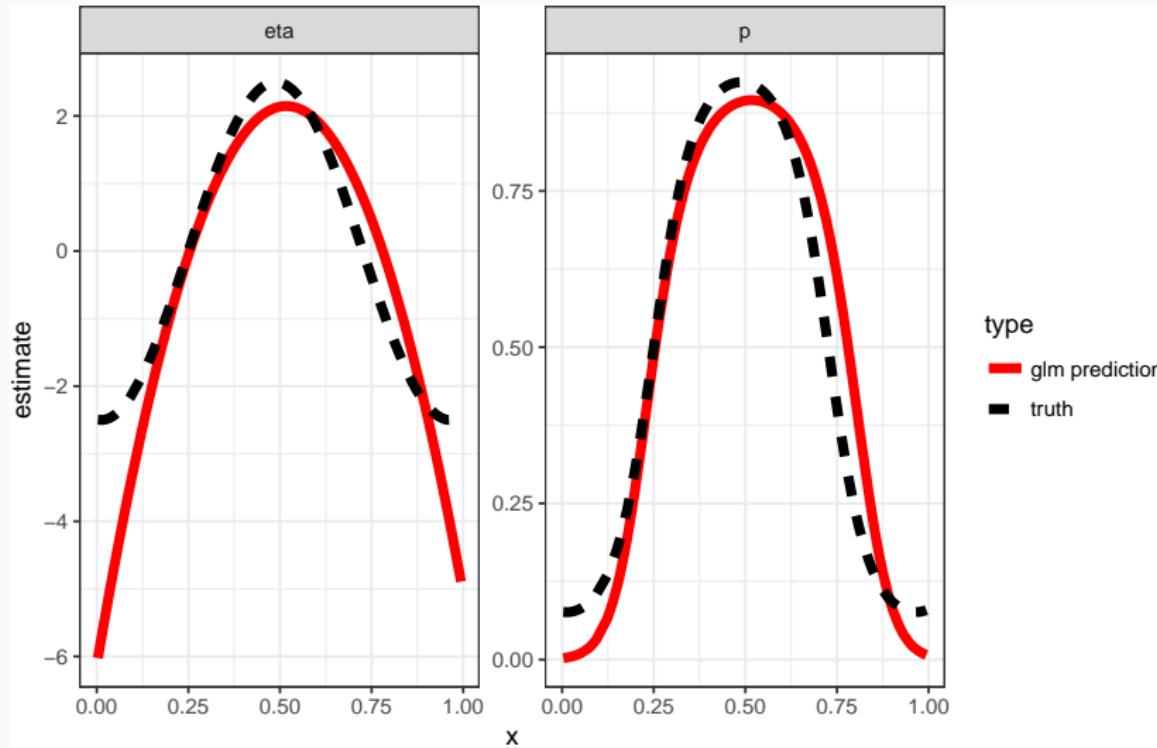


Model Results - Fit



Model vs glm

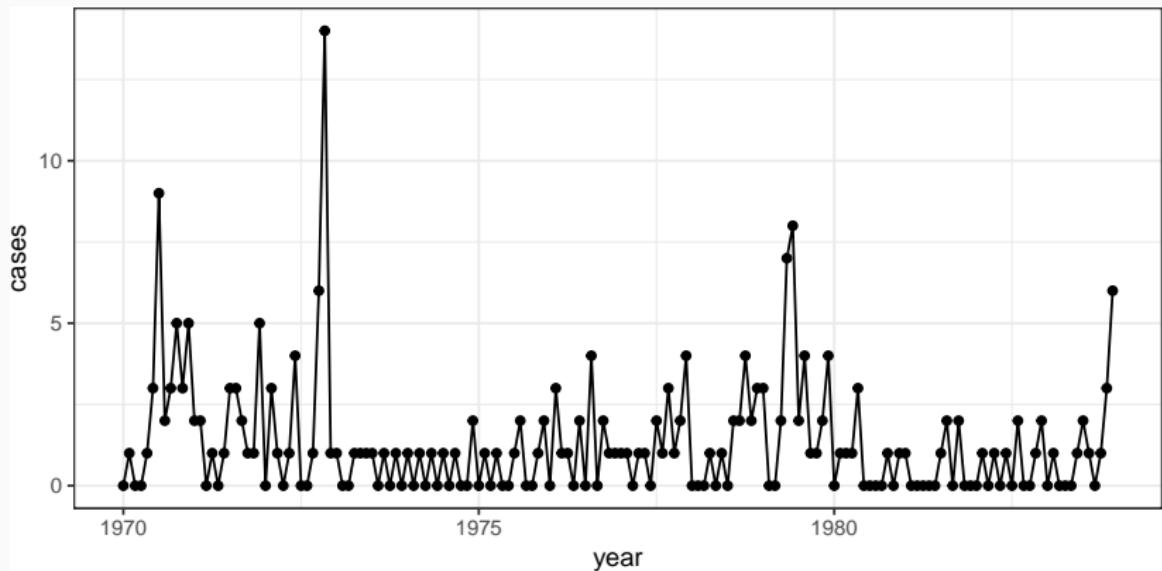
```
g = glm(y~poly(x,2), data=d, family="binomial")
```



Count data - Polio cases

Polio from the glarma package.

This data set gives the monthly number of cases of poliomyelitis in the U.S. for the years 1970–1983 as reported by the Center for Disease Control.



Polio Model

Model:

$$y_i \sim \text{Pois}(\lambda_i)$$
$$\log(\lambda_i) = \beta_0 + w(\mathbf{t})$$

where

$$w(\mathbf{t}) \sim \mathcal{N}(0, \Sigma)$$

$$\{\Sigma\}_{ij} = \sigma^2 \exp(-|\phi d_{ij}|)$$

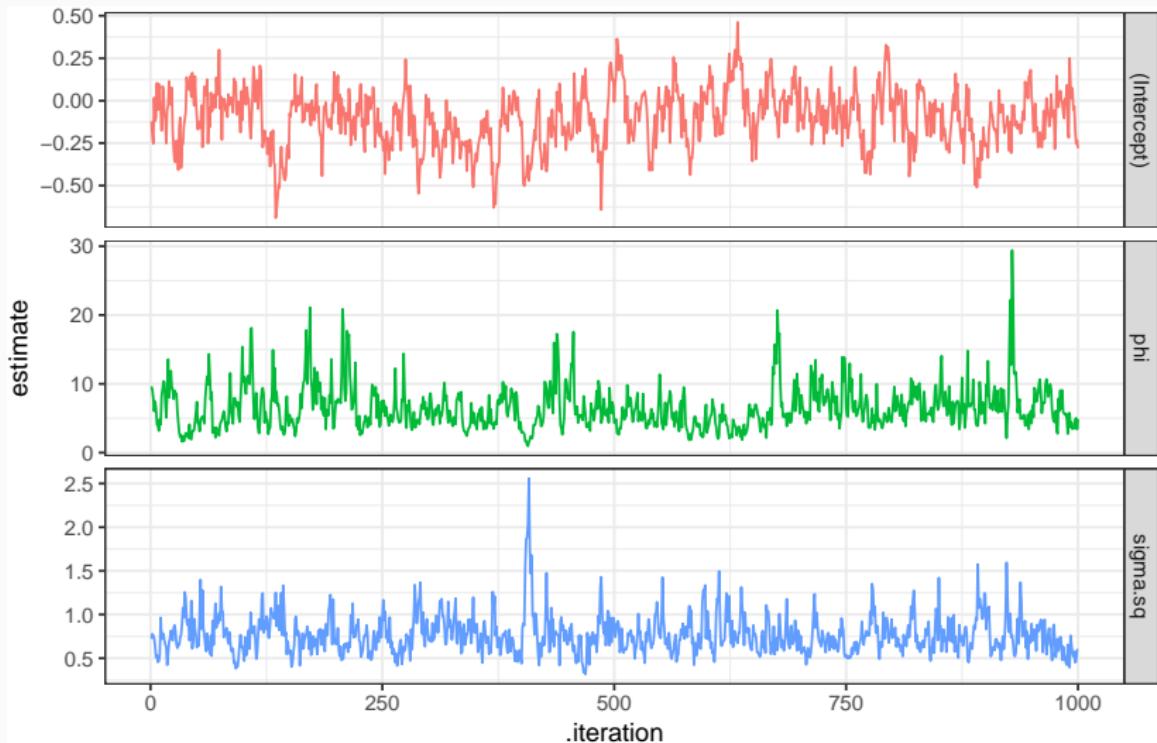
Priors:

$$\beta_0 \sim \mathcal{N}(0, 1)$$

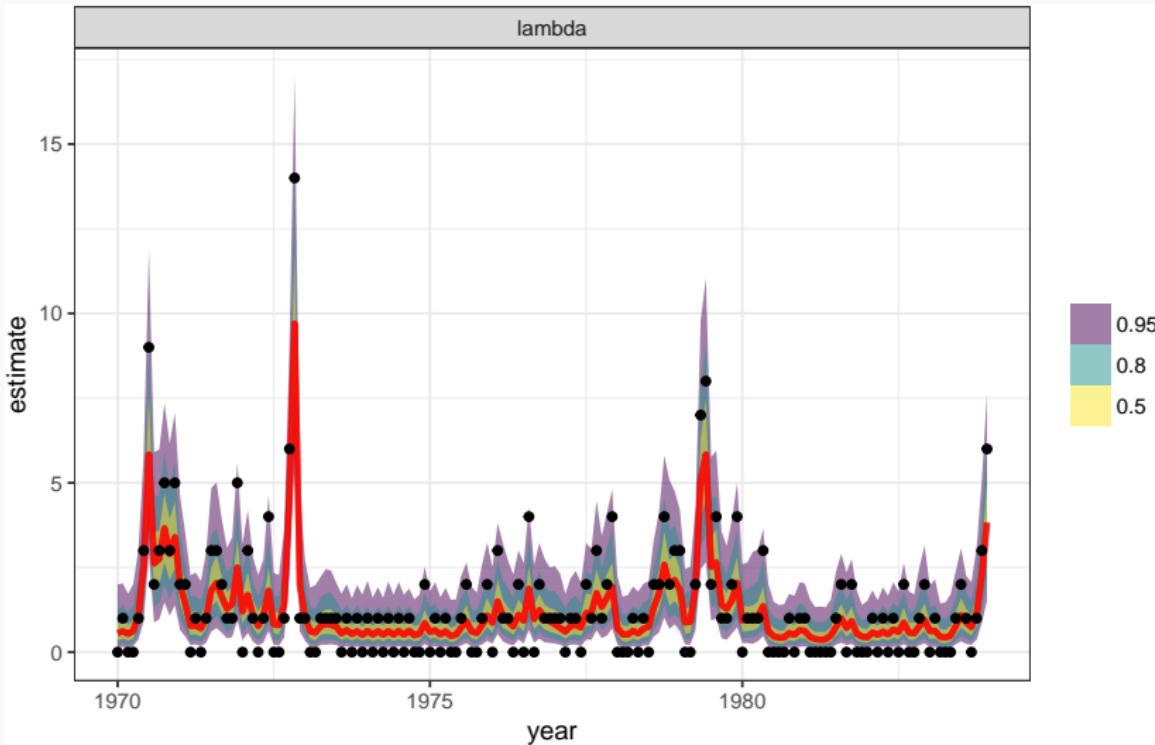
$$\phi \sim \text{Unif}\left(\frac{3}{6}, \frac{3}{1/12}\right)$$

$$1/\sigma^2 \sim \text{Gamma}(2, 1)$$

Model Results - Diagnostics



Model Results - Fit



Spatial data in R

Analysis of geospatial data in R

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

Some core packages (CRAN - Spatial task view):

- **sp** - core classes for handling spatial data, additional utility functions.
- **rgdal** - R interface to **gdal** (Geospatial Data Abstraction Library) for reading and writing spatial data.
- **rgeos** - R interface to **geos** (Geometry Engine Open Source) library for querying and manipulating spatial data. Reading and writing WKT.
- **sf** - Combines the functionality of **sp**, **rgdal**, and **rgeos** into a single package based on tidy principles.
- **lwgeom** - additional functionality for **sf** using PostGIS' `liblwgeom`.
- **raster** - classes and tools for handling raster data.

Installing sf

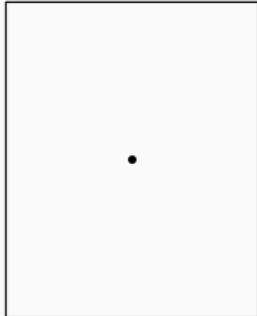
This is the hardest part of using the `sf` package, difficulty comes from its dependence on several external libraries (`geos`, `gdal`, and `proj`).

- *Windows* - installing from source works when Rtools is installed (system requirements are downloaded from rwinlib)
- *MacOS* - install dependencies via homebrew: `gdal2`, `geos`, `proj`.
- *Linux* - Install development packages for GDAL (>= 2.0.0), GEOS (>= 3.3.0) and Proj.4 (>= 4.8.0) from your package manager of choice.

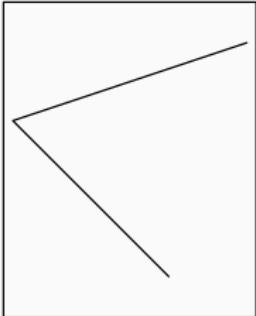
More specific details are included in the repo readme on github.

Simple Features

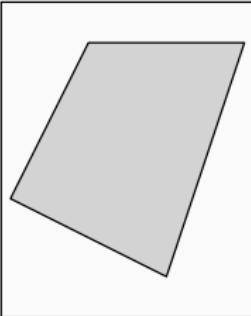
Point



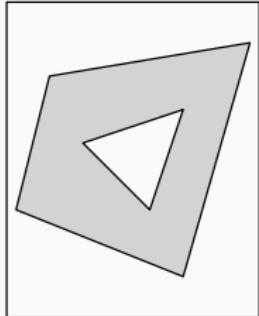
Linestring



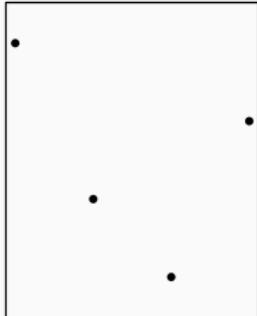
Polygon



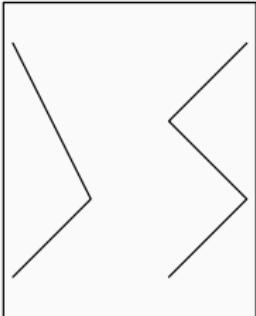
Polygon w/ Hole(s)



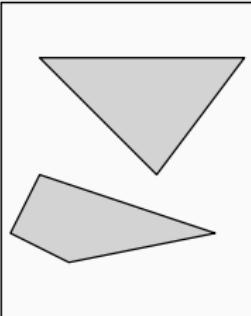
Multipoint



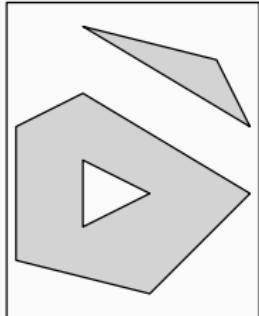
Multilinestring



Multipolygon



Multipolygon w/ Hole(s)



Reading, writing, and converting simple features

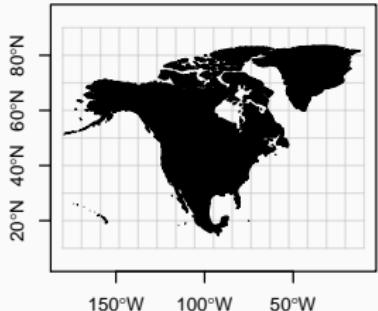
- `maptools`
 - `readShapePoints / writeShapePoints` - Shapefile w/ points
 - `readShapeLines / writeShapeLines` - Shapefile w/ lines
 - `readShapePoly / writeShapePoly` - Shapefile w/ polygons
 - `readShapeSpatial / writeShapeSpatial` - Shapefile
- `rgdal`
 - `readOGR / writeOGR` - Shapefile, GeoJSON, KML, ...
- `rgeos`
 - `readWKT / writeWKT` - Well Known Text
- `sf`
 - `st_read / st_write` - Shapefile, GeoJSON, KML, ...
 - `st_as_sfc / st_as_wkt` - WKT
 - `st_as_sfc / st_as_binary` - WKB
 - `st_as_sfc / as(x, "Spatial")` - sp

See `sf` vignette #2.

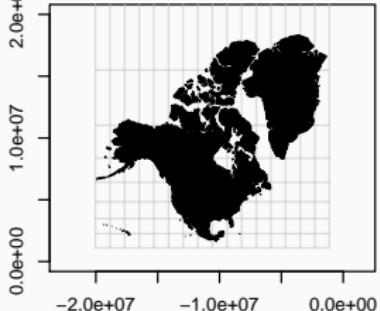
Geospatial data in the real world

Projections

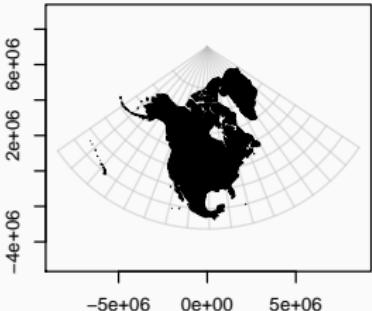
Lat/Long (epsg:4326)



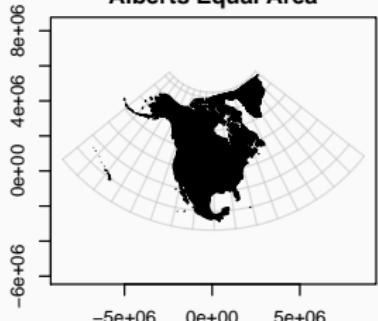
Google / Web Mercator (epsg:3857)



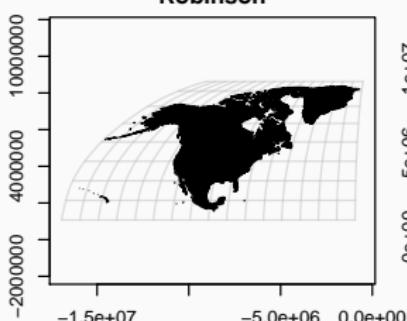
Lambert Conformal Conic:



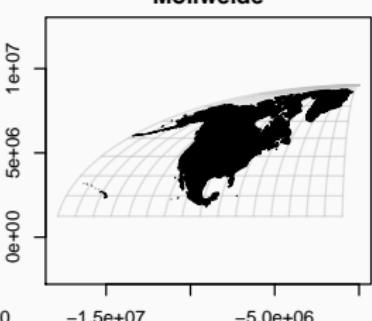
Alberts Equal Area



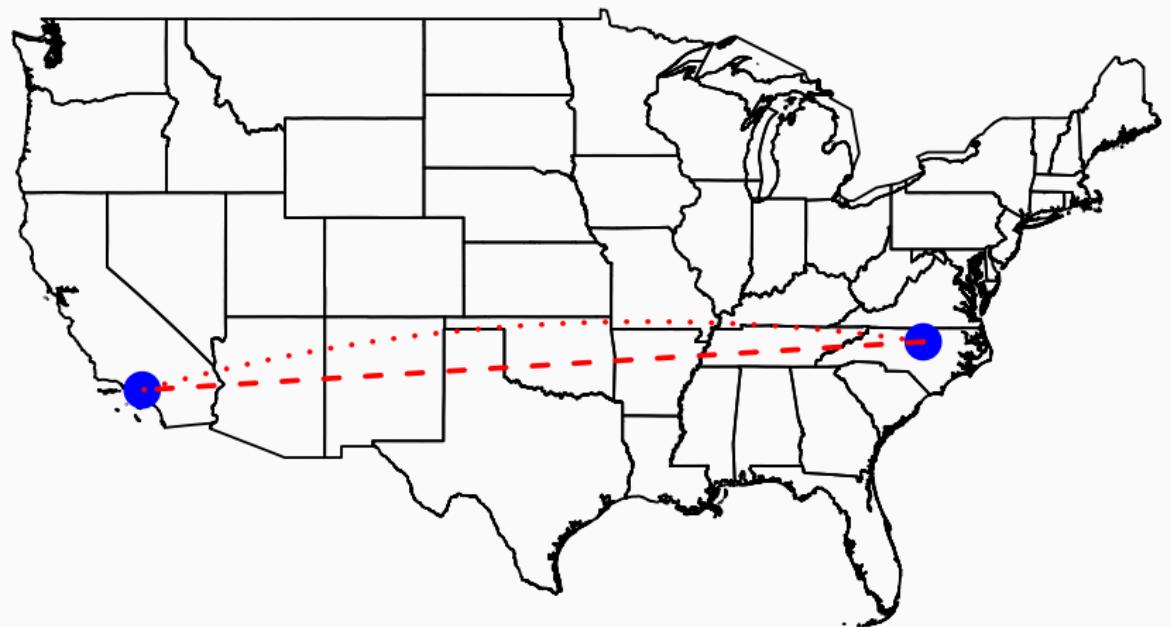
Robinson



Mollweide

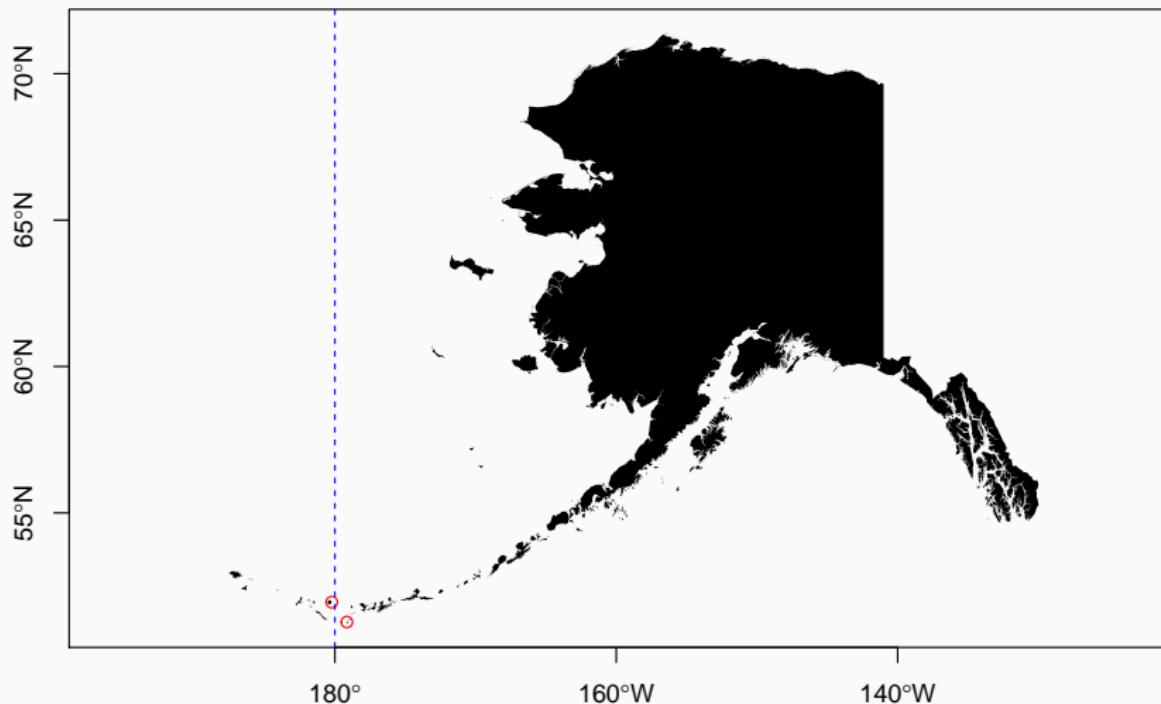


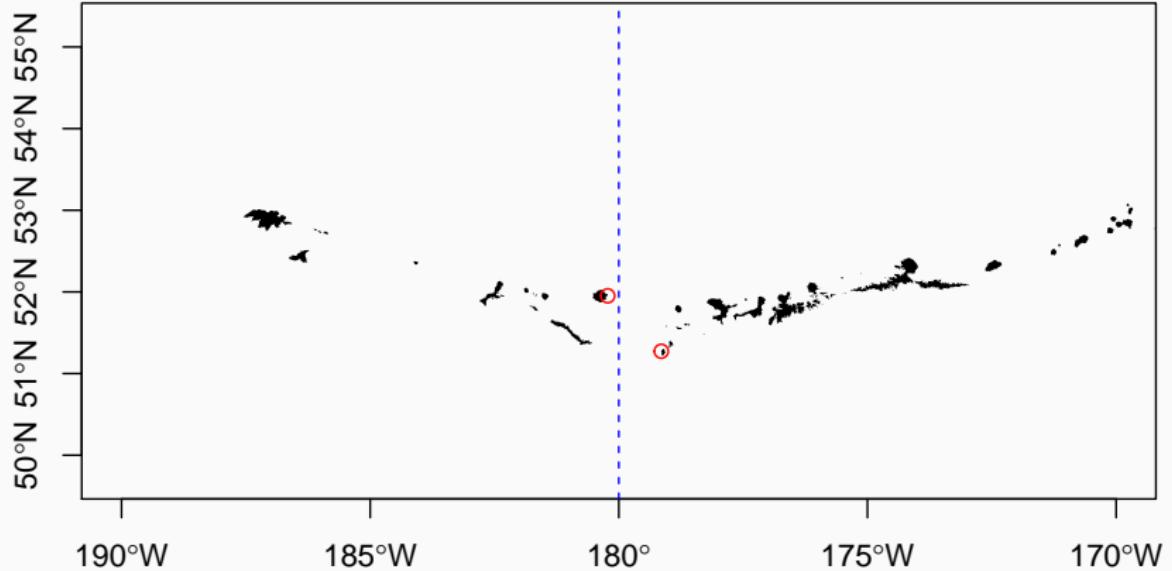
Distance on a Sphere

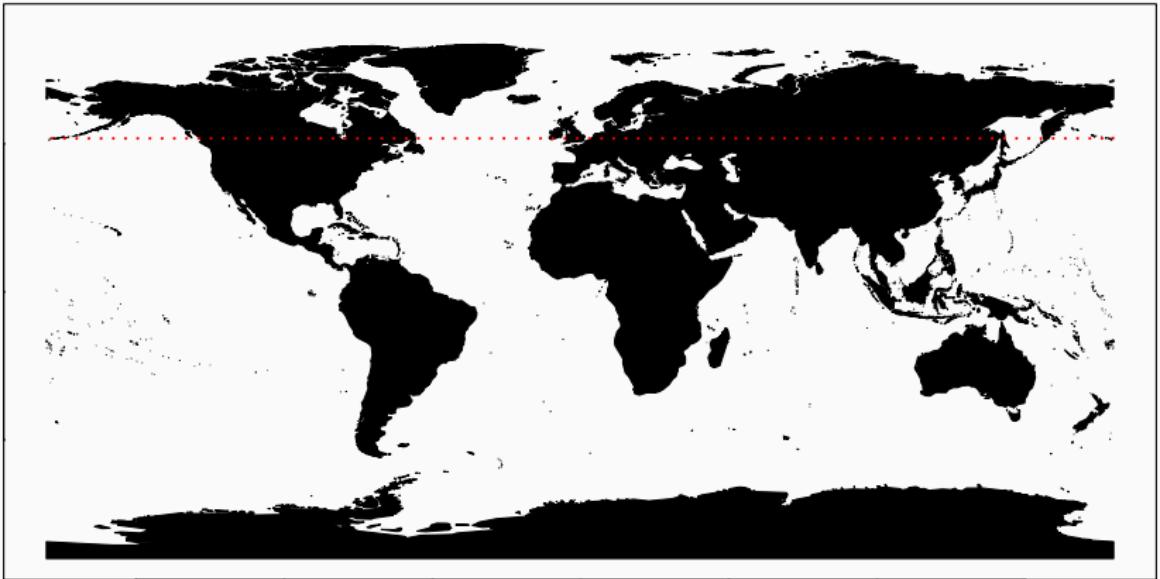


Dateline

Want to fly from the Western most point in the US to the Eastern most point?







Using sf

Example data

```
nc  = st_read("../data/gis/nc_counties/", quiet=TRUE, stringsAsFactors=FALSE)
air = st_read("../data/gis/airports/", quiet=TRUE, stringsAsFactors=FALSE)
hwy = st_read("../data/gis/us_interstates/", quiet=TRUE, stringsAsFactors=FALSE)

tbl_df(nc)
## # A tibble: 100 x 9
##   AREA PERIMETER COUNTY P010 STATE COUNTY    FIPS STATE_FIPS SQUARE_MIL
##   <dbl>     <dbl>    <dbl> <chr>  <chr>    <chr> <chr>      <dbl>
## 1 0.112      1.61    1994. NC    Ashe Cou~ 37009 37        429.
## 2 0.0616     1.35    1996. NC    Alleghan~ 37005 37        236.
## 3 0.140       1.77    1998. NC    Surry Co~ 37171 37        539.
## 4 0.0891     1.43    1999. NC    Gates Co~ 37073 37        342.
## 5 0.0687     4.43    2000. NC    Currituc~ 37053 37        264.
## 6 0.119       1.40    2001. NC    Stokes C~ 37169 37        456.
## 7 0.0626     2.11    2002. NC    Camden C~ 37029 37        241.
## 8 0.115       1.46    2003. NC    Warren C~ 37185 37        444.
## 9 0.143       2.40    2004. NC    Northamp~ 37131 37        551.
## 10 0.0925    1.81    2005. NC   Hertford~ 37091 37        356.
## # ... with 90 more rows, and 1 more variable: geometry <sf_geometry>
## #   [degree]
```

```
tbl_df(air)
## # A tibble: 940 x 17
##   AIRPRTX010 FEATURE ICAO  IATA  AIRPT_NAME CITY STATE STATE_FIPS COUNTY
##   <dbl> <chr>   <chr> <chr> <chr>    <chr> <chr> <chr>   <chr>
## 1     0. AIRPORT KGON  GON   GROTON-NE~ GROT~ CT    09   NEW L~
## 2     3. AIRPORT K6S5  6S5   RAVALLI C~ HAMI~ MT    30   RAVAL~
## 3     4. AIRPORT KMHV  MHV   MOJAVE AI~ MOJA~ CA    06   KERN
## 4     6. AIRPORT KSEE  SEE   GILLESPIE~ SAN ~ CA    06   SAN D~
## 5     7. AIRPORT KFPR  FPR   ST LUCIE ~ FORT~ FL    12   ST LU~
## 6     8. AIRPORT KRYY  RYY   COBB COUN~ ATLA~ GA    13   COBB
## 7    10. AIRPORT KMKL  MKL   MC KELLAR~ JACK~ TN    47   MADIS~
## 8    11. AIRPORT KCCR  CCR   BUCHANAN ~ CONC~ CA    06   CONTR~
## 9    13. AIRPORT KJYO  JYO   LEESBURG ~ LEES~ VA    51   LOUDO~
## 10   15. AIRPORT KCAD  CAD   WEXFORD C~ CADI~ MI    26   WEXFO~
## # ... with 930 more rows, and 8 more variables: FIPS <chr>, TOT_ENP <dbl>,
## #   LATITUDE <dbl>, LONGITUDE <dbl>, ELEV <dbl>, ACT_DATE <chr>,
## #   CNTL_TWR <chr>, geometry <sf_geometry [degree]>
```

```
tbl_df(hwy)
## # A tibble: 233 x 4
##   ROUTE_NUM DIST_MILES DIST_KM           geometry
##   <chr>        <dbl>    <dbl>      <sf_geometry [m]>
## 1 I10          2449.   3941. MULTILINESTRING ((-1881200 ...
## 2 I105         20.8     33.4  MULTILINESTRING ((-1910156 ...
## 3 I110         41.4     66.6  MULTILINESTRING ((1054139 3...
## 4 I115         1.58     2.55  MULTILINESTRING ((-1013796 ...
## 5 I12          85.3     137.  MULTILINESTRING ((680741.7 ...
## 6 I124         1.73     2.79  MULTILINESTRING ((1201467 3...
## 7 I126         3.56     5.72  MULTILINESTRING ((1601502 3...
## 8 I129         3.10     4.99  MULTILINESTRING ((217446 47...
## 9 I135         96.3     155.  MULTILINESTRING ((96922.97 ...
## 10 I15         1436.   2311. MULTILINESTRING ((-882875.7...
## # ... with 223 more rows
```

sf classes

```
str(nc)
## Classes 'sf' and 'data.frame': 100 obs. of 9 variables:
## $ AREA      : num 0.1118 0.0616 0.1402 0.0891 0.0687 ...
## $ PERIMETER : num 1.61 1.35 1.77 1.43 4.43 ...
## $ COUNTYP010: num 1994 1996 1998 1999 2000 ...
## $ STATE     : chr "NC" "NC" "NC" "NC" ...
## $ COUNTY    : chr "Ashe County" "Alleghany County" "Surry County" "Gates County"
## $ FIPS      : chr "37009" "37005" "37171" "37073" ...
## $ STATE_FIPS: chr "37" "37" "37" "37" ...
## $ SQUARE_MIL: num 429 236 539 342 264 ...
## $ geometry  :sfc_MULTIPOLYGON of length 100; first list element: List of 1
##   ..$ :List of 1
##     ...$ : num [1:1030, 1:2] -81.7 -81.7 -81.7 -81.6 -81.6 ...
##     -- attr(*, "class")= chr "XY" "MULTIPOLYGON" "sfg"
##     - attr(*, "sf_column")= chr "geometry"
##     - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA
##     -- attr(*, "names")= chr "AREA" "PERIMETER" "COUNTYP010" "STATE" ...

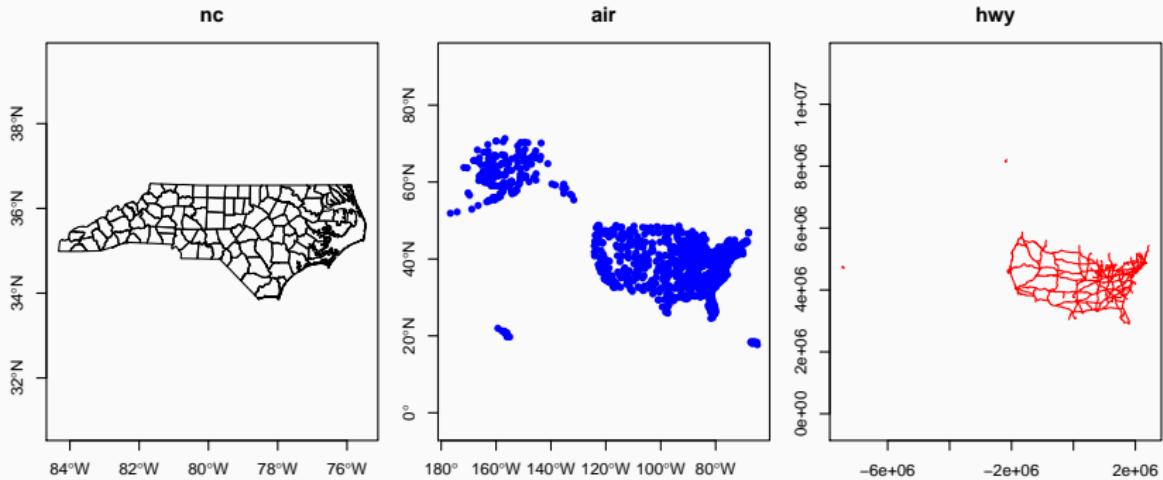
class(nc)
## [1] "sf"           "data.frame"

class(nc$geometry)
## [1] "sfc_MULTIPOLYGON" "sfc"

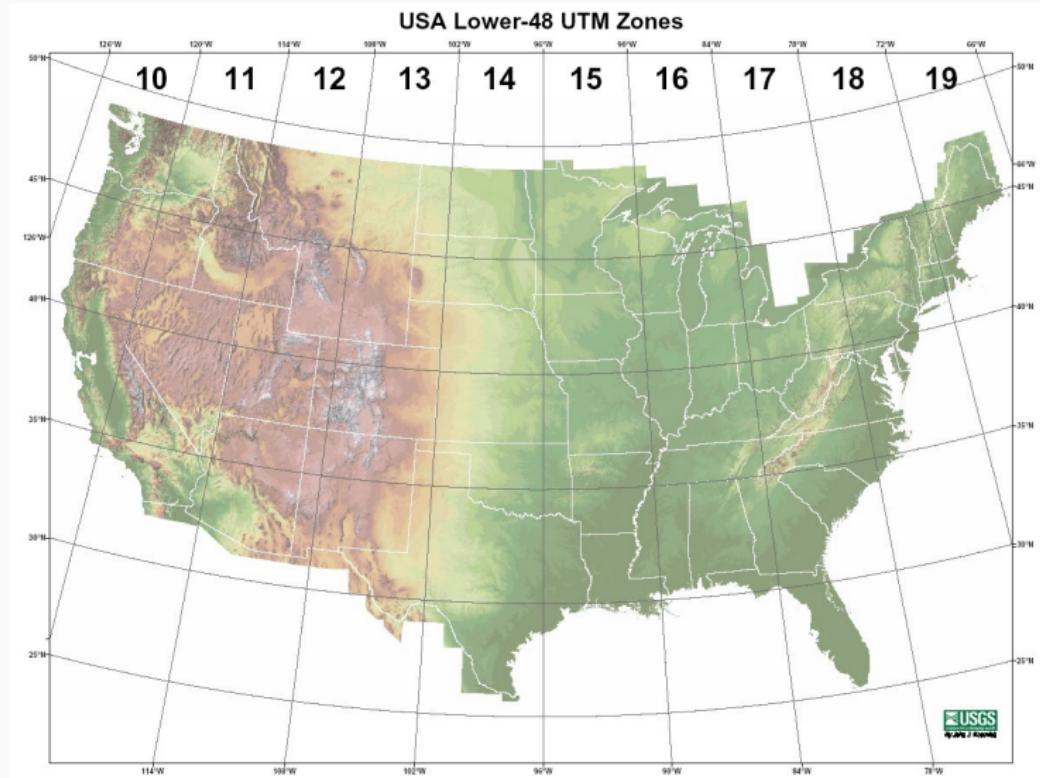
class(nc$geometry[[1]])
## [1] "XY"           "MULTIPOLYGON" "sfg"
```

Projections

```
st_crs(nc)$proj4string  
## [1] "+proj=longlat +datum=NAD83 +no_defs"  
st_crs(air)$proj4string  
## [1] "+proj=longlat +datum=NAD83 +no_defs"  
st_crs(hwy)$proj4string  
## [1] "+proj=utm +zone=15 +datum=NAD83 +units=m +no_defs"
```

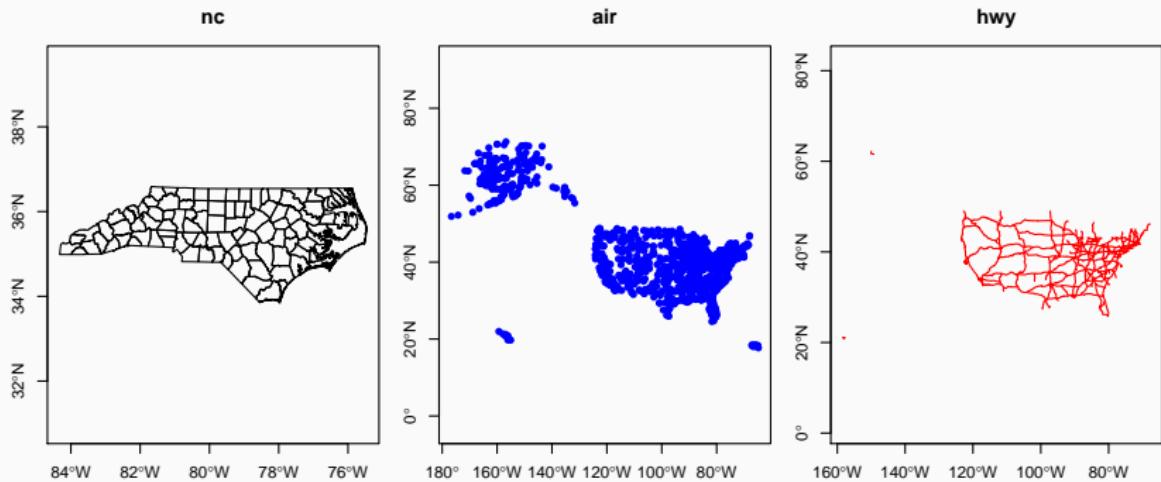


UTM Zones



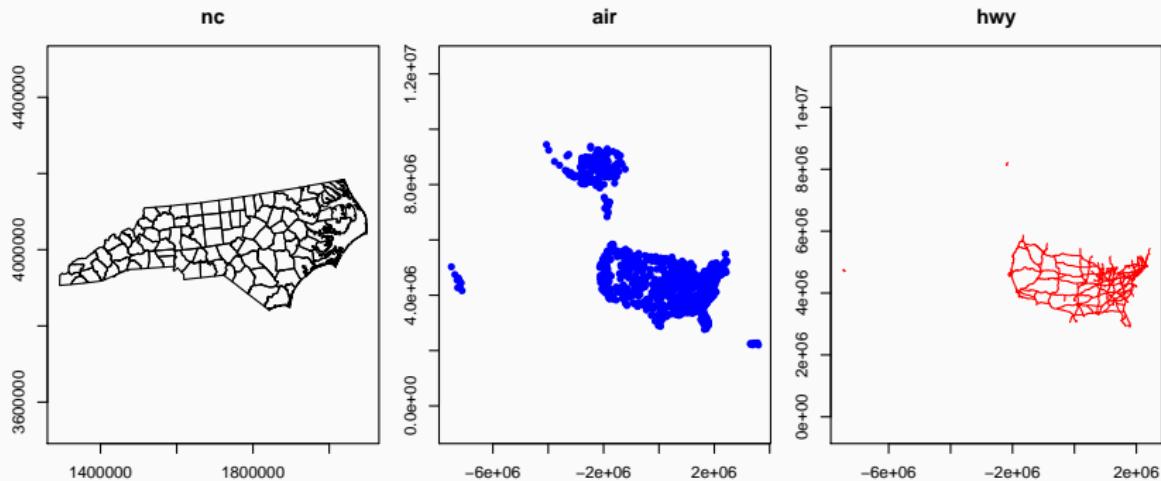
Lat/Long

```
nc_ll = nc  
air_ll = air  
hwy_ll = lwgeom::st_transform_proj(hwy, st_crs(nc)$proj4string)
```



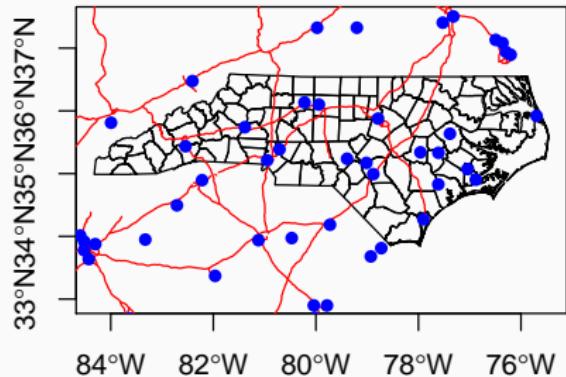
UTM

```
nc_utm = lwgeom::st_transform_proj(nc, st_crs(hwy)$proj4string)
air_utm = lwgeom::st_transform_proj(air, st_crs(hwy)$proj4string)
hwy_utm = hwy
```

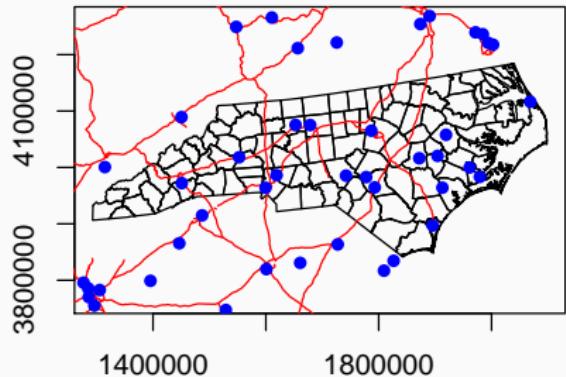


Comparison

Lat/Long



UTM



Geometry Predicates

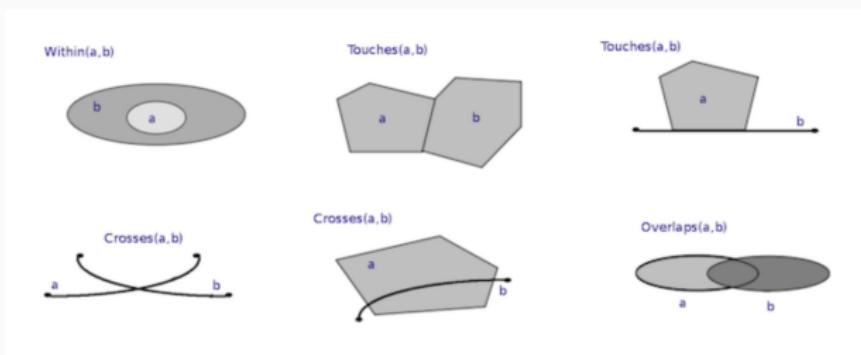


b



	Interior	Boundary	Exterior
Interior	 $\dim[I(a) \cap I(b)] = 2$	 $\dim[I(a) \cap B(b)] = 1$	 $\dim[I(a) \cap E(b)] = 2$
Boundary	 $\dim[B(a) \cap I(b)] = 1$	 $\dim[B(a) \cap B(b)] = 0$	 $\dim[B(a) \cap E(b)] = 1$
Exterior	 $\dim[E(a) \cap I(b)] = 2$	 $\dim[E(a) \cap B(b)] = 1$	 $\dim[E(a) \cap E(b)] = 2$

Spatial predicates



st_within(a,b)

$$\begin{bmatrix} T & * & F \\ * & * & F \\ * & * & * \end{bmatrix}$$

st_crosses(a,b)

$$\text{If } \dim(a) < \dim(b) \quad \text{If } \dim(a) > \dim(b) \quad \text{If } \dim(a \cap b) = 1$$

$$\begin{bmatrix} T & * & T \\ * & * & * \\ * & * & * \end{bmatrix} \quad \begin{bmatrix} T & * & * \\ * & * & * \\ T & * & * \end{bmatrix} \quad \begin{bmatrix} 0 & * & * \\ * & * & * \\ * & * & * \end{bmatrix}$$

st_touches(a,b)

$$\begin{bmatrix} F & T & * \\ * & * & * \\ * & * & * \end{bmatrix} \cup \begin{bmatrix} F & * & * \\ T & * & * \\ * & * & * \end{bmatrix} \cup \begin{bmatrix} F & * & * \\ * & T & * \\ * & * & * \end{bmatrix}$$

st_overlaps(a,b) ($\dim(a) = \dim(b)$)

$$\text{If } \dim \in \{0, 2\} \quad \text{If } \dim = 1$$

$$\begin{bmatrix} T & * & T \\ * & * & * \\ T & * & * \end{bmatrix} \quad \begin{bmatrix} 1 & * & T \\ * & * & * \\ T & * & * \end{bmatrix}$$

Sparse vs Full Results

```
st_intersects(nc[20:30], air) %>% str()
## although coordinates are longitude/latitude, st_intersects assumes that t
## 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] "20" "21" "22" "23" ...
## - attr(*, "ncol")= int 940
## - attr(*, "class")= chr "sgbp"
```

```
st_intersects(nc, air, sparse=FALSE) %>% str()
## although coordinates are longitude/latitude, st_intersects assumes that t
## logi [1:100, 1:940] FALSE FALSE FALSE FALSE FALSE FALSE ...
```

Examples

- Which counties are adjacent to Durham County?
- Which counties have more than 4 neighbors?
- Which counties have an airport?