GPs for GLMs + Spatial Data

Lecture 16

Dr. Colin Rundel

GPs and GLMs

Logistic Regression

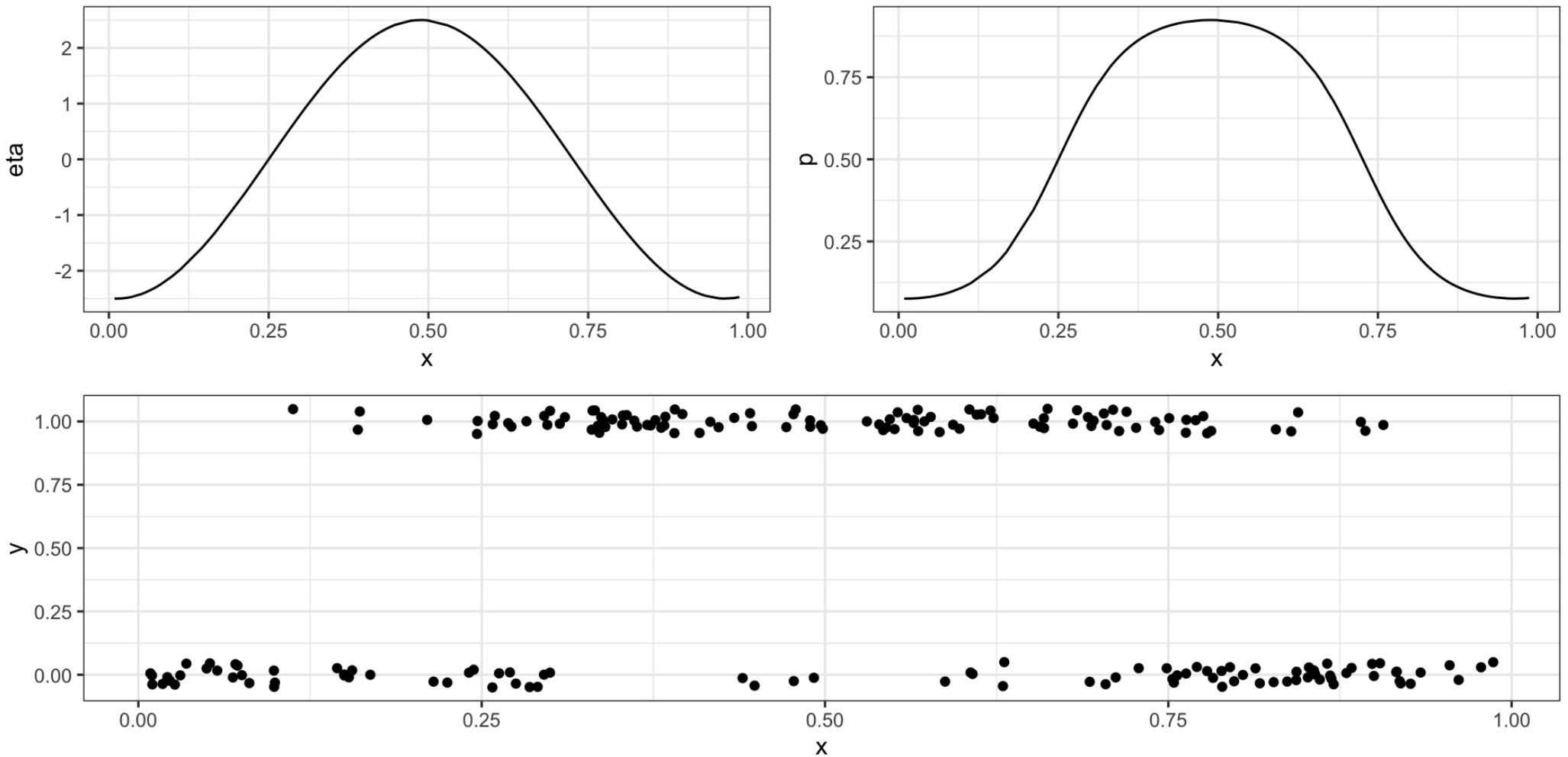
A typical logistic regression problem uses the following model,

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

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

$$\begin{aligned} y_i &\sim \text{Bern}(p_i) \\ \text{logit}(p_i) &= \eta_i = \boldsymbol{X}\boldsymbol{\beta} + w(\boldsymbol{x}) \\ w(\boldsymbol{x}) &\sim \text{N}(0, \Sigma) \end{aligned}$$

A toy example



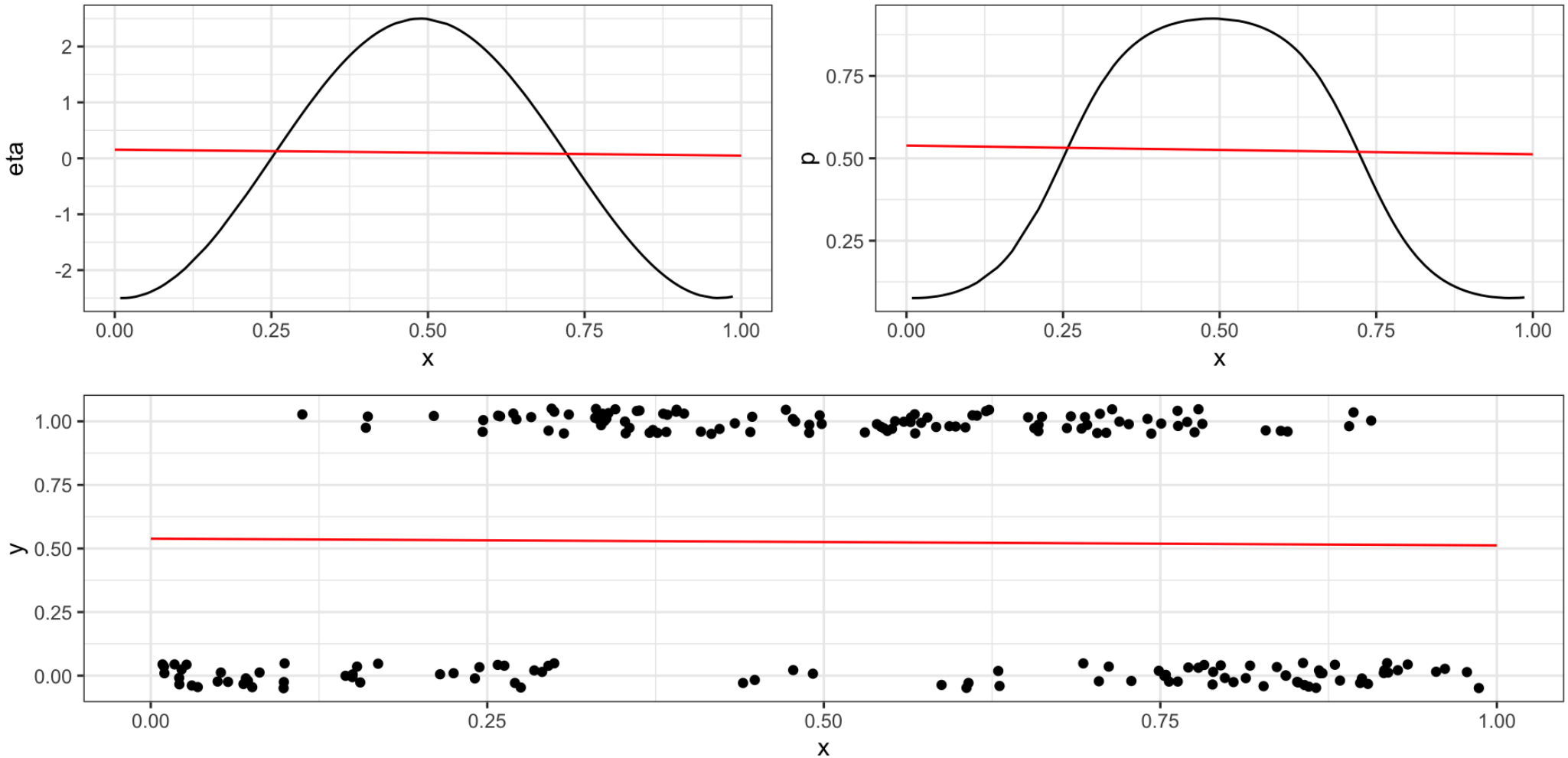
A standard GLM

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

```
Call:  glm(formula = y ~ x, family = "binomial", data = d)

Coefficients:
(Intercept)            x
     0.1552      -0.1065

Degrees of Freedom: 199 Total (i.e. Null);  198 Residual
Null Deviance:      276.8
Residual Deviance: 276.7    AIC: 280.7
```


eta
2
1
0
-1
-2
0.00
0.25
0.50
0.75
1.00
x
p
0.75
0.50
0.25
0.00
0.25
0.50
0.75
1.00
x
y
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
x

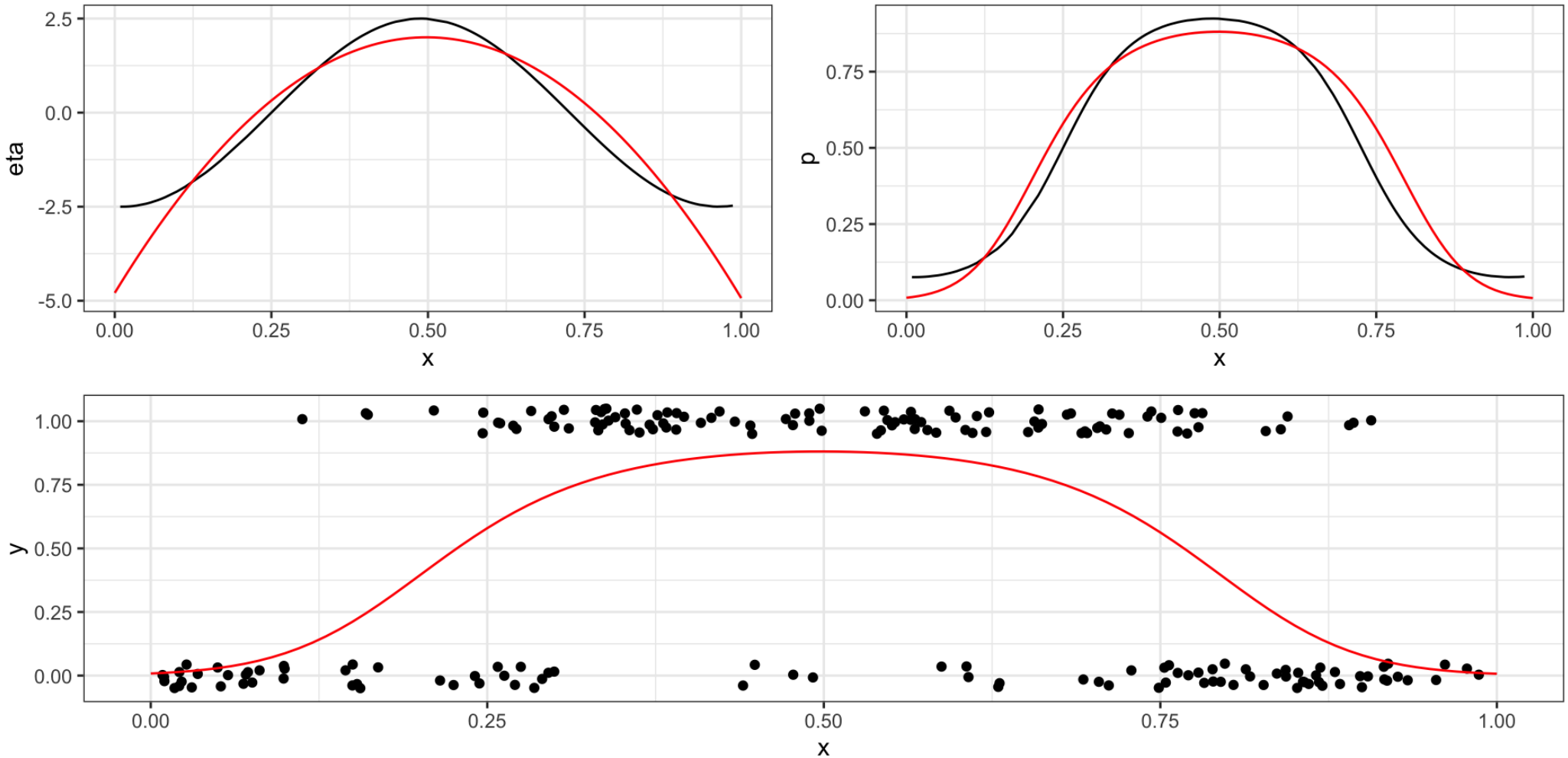
A quadratic GLM

```r
(g2 = glm(y~poly(x,2), family="binomial", data=d))
```

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

Coefficients:
(Intercept)  poly(x, 2)1  poly(x, 2)2
   -0.08571      1.40019    -26.82507

Degrees of Freedom: 199 Total (i.e. Null);  197 Residual
Null Deviance:      276.8
Residual Deviance: 184.7    AIC: 190.7
```

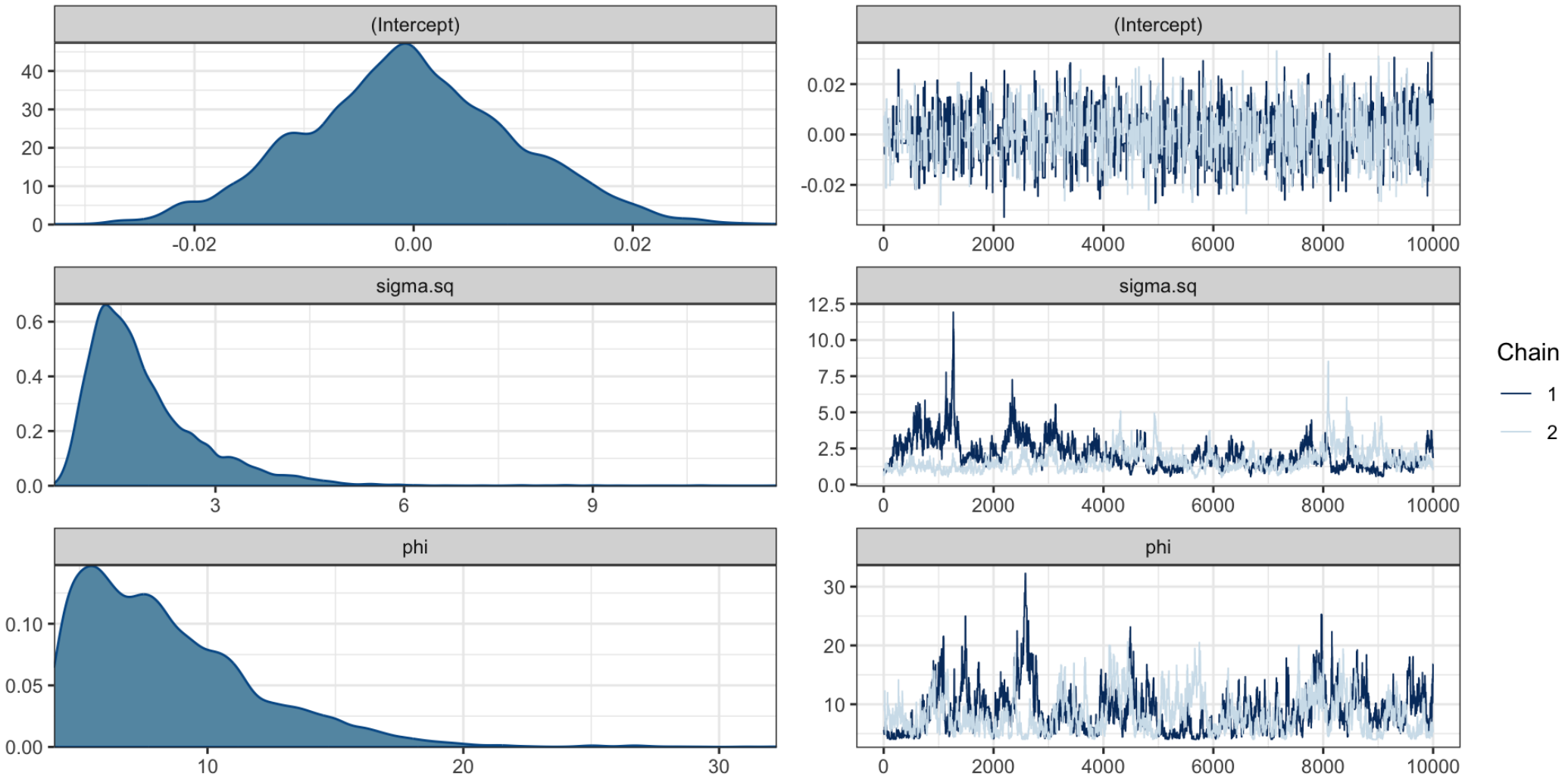

eta
2.5
0.0
-2.5
-5.0
0.00
0.25
0.50
0.75
1.00
x
p
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
x
y
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
x

Model fitting

```
m = gpglm(
  chains = 2,
  y~1, family="binomial",
  data = d, coords = c("x"),
  cov_model = "exponential",
  n_batch = 200,
  batch_len = 100,
  starting=list(
    "beta"=0, "phi"=3/0.5, "sigma.sq"=5, "w"=0
  ),
  priors=list(
    "beta.Normal"=list(0,0.01),
    "phi.unif"=c(3/0.75, 3/0.01),
    "sigma.sq.ig"=c(2, 1)
  ),
  tuning=list(
    "beta"=0.5, "phi"=0.5, "sigma.sq"=0.5, "w"=0.5
  ),
  verbose=TRUE
)
```

Model diagnostics

```
1 plot(m)
```



Model predictions

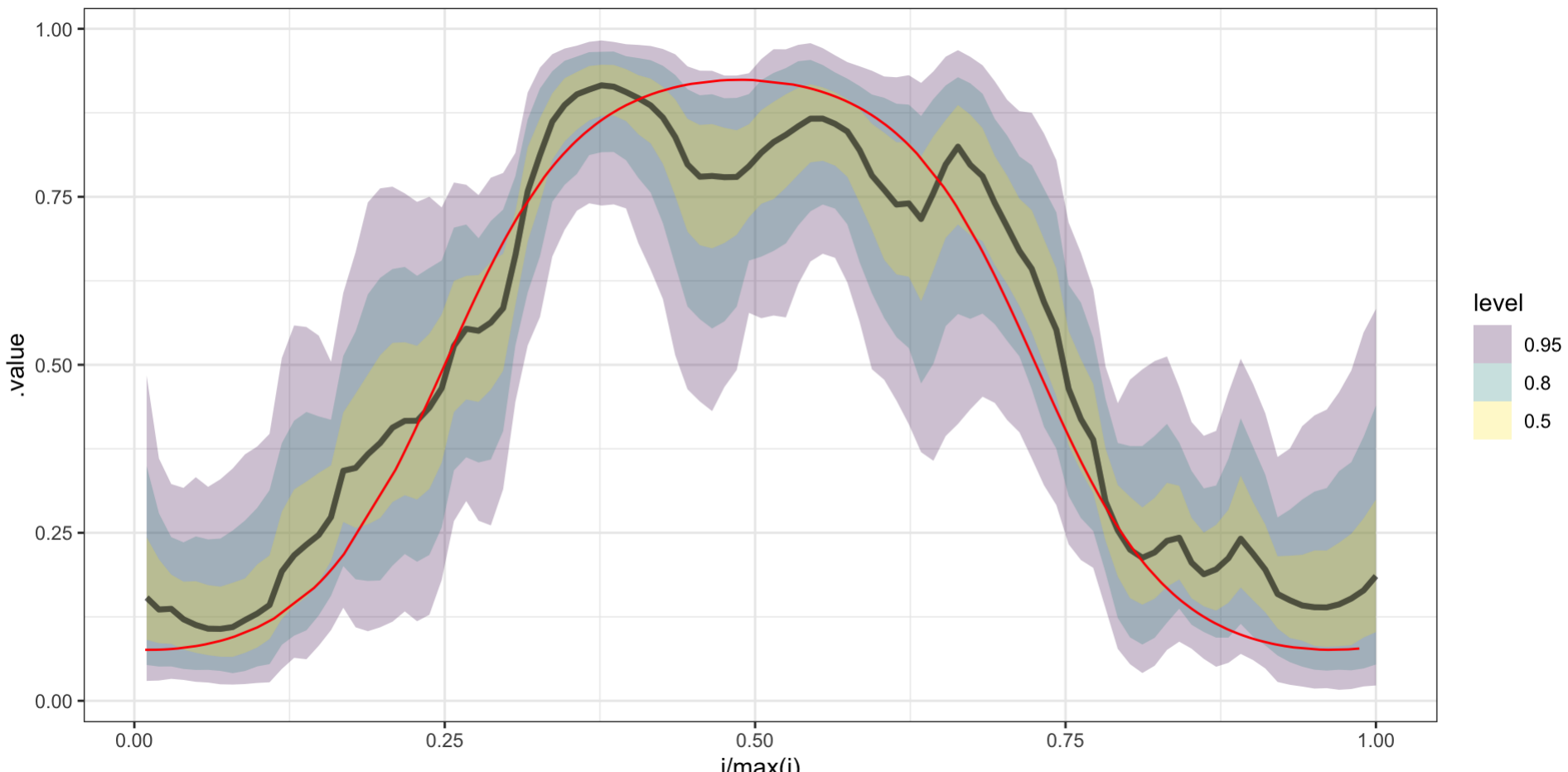
```r
newdata = data.frame(
  x=seq(0,1,length.out=101)
)

(p = predict(m, newdata=newdata, coords="x", thin=10))
```

```
# A draws_matrix: 1000 iterations, 2 chains, and 202 variables
    variable
draw w[1] w[2] w[3] w[4] w[5] w[6] w[7] w[8]
  1  -2.7 -3.1 -3.3 -3.1 -3.1 -3.2 -2.9 -3.2
  2  -3.4 -3.1 -3.3 -3.0 -3.2 -3.2 -3.0 -3.0
  3  -3.0 -3.3 -3.3 -3.1 -2.5 -3.0 -2.9 -3.1
  4  -3.6 -3.2 -3.3 -3.2 -2.8 -2.6 -3.1 -2.9
  5  -2.8 -3.3 -3.2 -2.9 -2.8 -2.3 -2.9 -2.8
  6  -3.0 -3.2 -3.3 -3.2 -3.3 -3.2 -3.2 -2.6
  7  -3.8 -3.1 -3.3 -3.0 -2.8 -2.7 -2.7 -2.6
  8  -3.5 -3.5 -3.2 -3.5 -2.9 -2.6 -2.8 -2.9
  9  -3.4 -3.4 -3.4 -3.4 -3.0 -2.8 -2.5 -2.9
  10 -3.1 -3.5 -3.4 -3.1 -2.9 -2.7 -2.3 -2.5
# ... with 1990 more draws, and 194 more variables
```

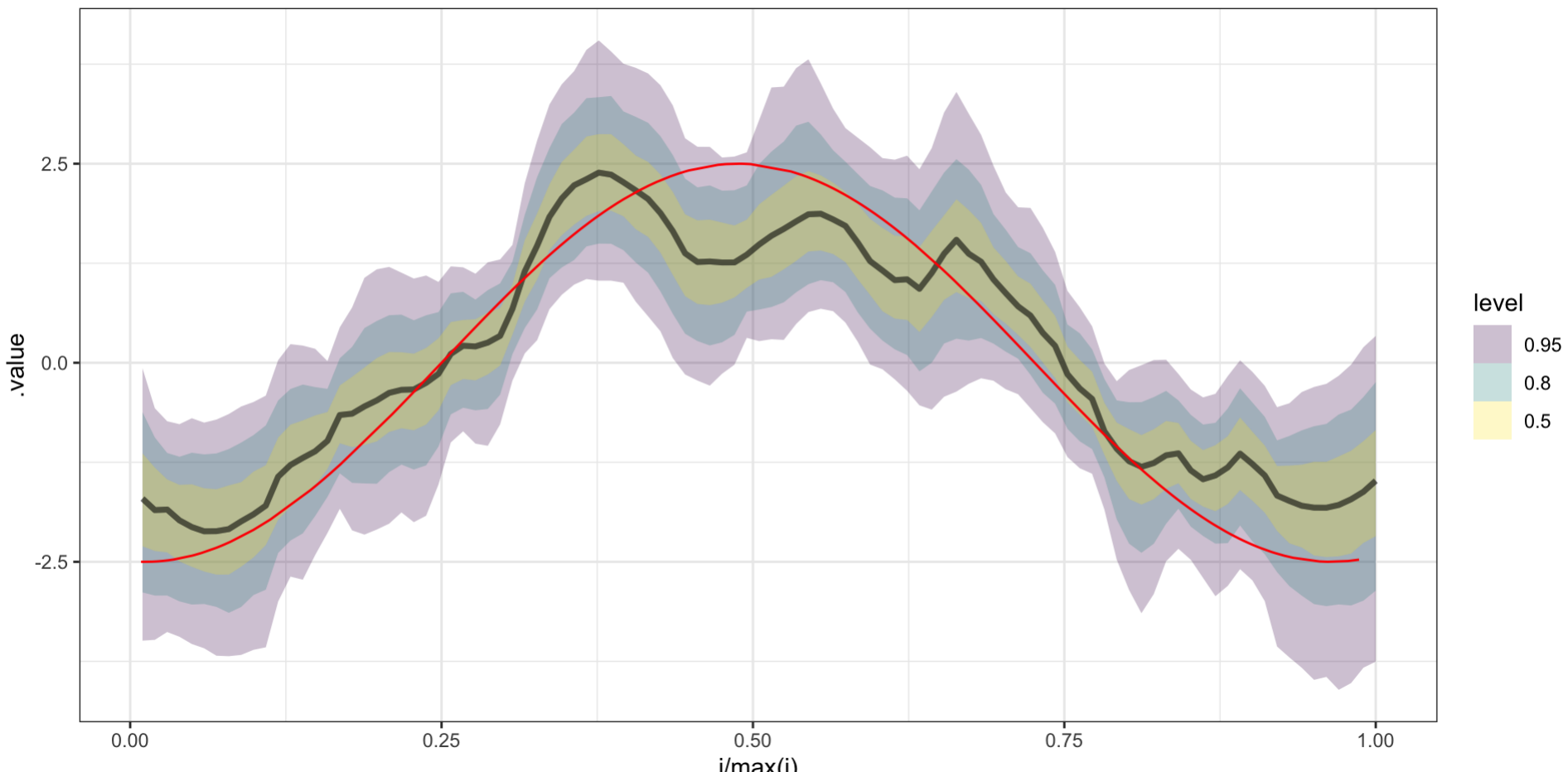
Predicted y

```
p |>
  tidybayes::gather_draws(y[i]) |>
  ggplot2::ggplot(ggplot2::aes(x=i/max(i),y=.value)) +
    tidybayes::stat_lineribbon(alpha=0.25) +
    geom_line(data=d |> arrange(x), aes(x=x, y=p), color='red')
```



Predicted w

```
p |>
  tidybayes::gather_draws(w[i]) |>
  ggplot2::ggplot(ggplot2::aes(x=i/max(i),y=.value)) +
    tidybayes::stat_lineribbon(alpha=0.25) +
    geom_line(data=d |> arrange(x), aes(x=x, y=eta), color='red')
```

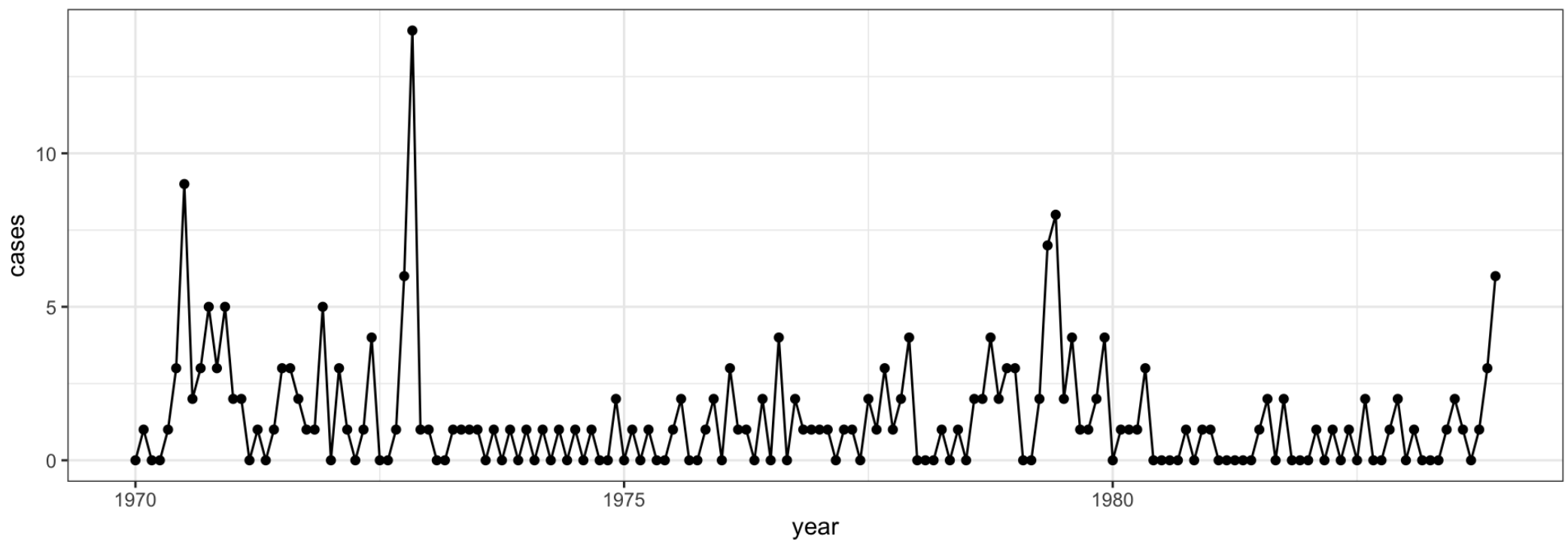


Count data

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:

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

$$w(\boldsymbol{t}) \sim N(0, \Sigma)$$
$$\{\boldsymbol{\Sigma}\}_{ij} = \sigma^2 \exp(-|l\, d_{ij}|)$$

Priors:

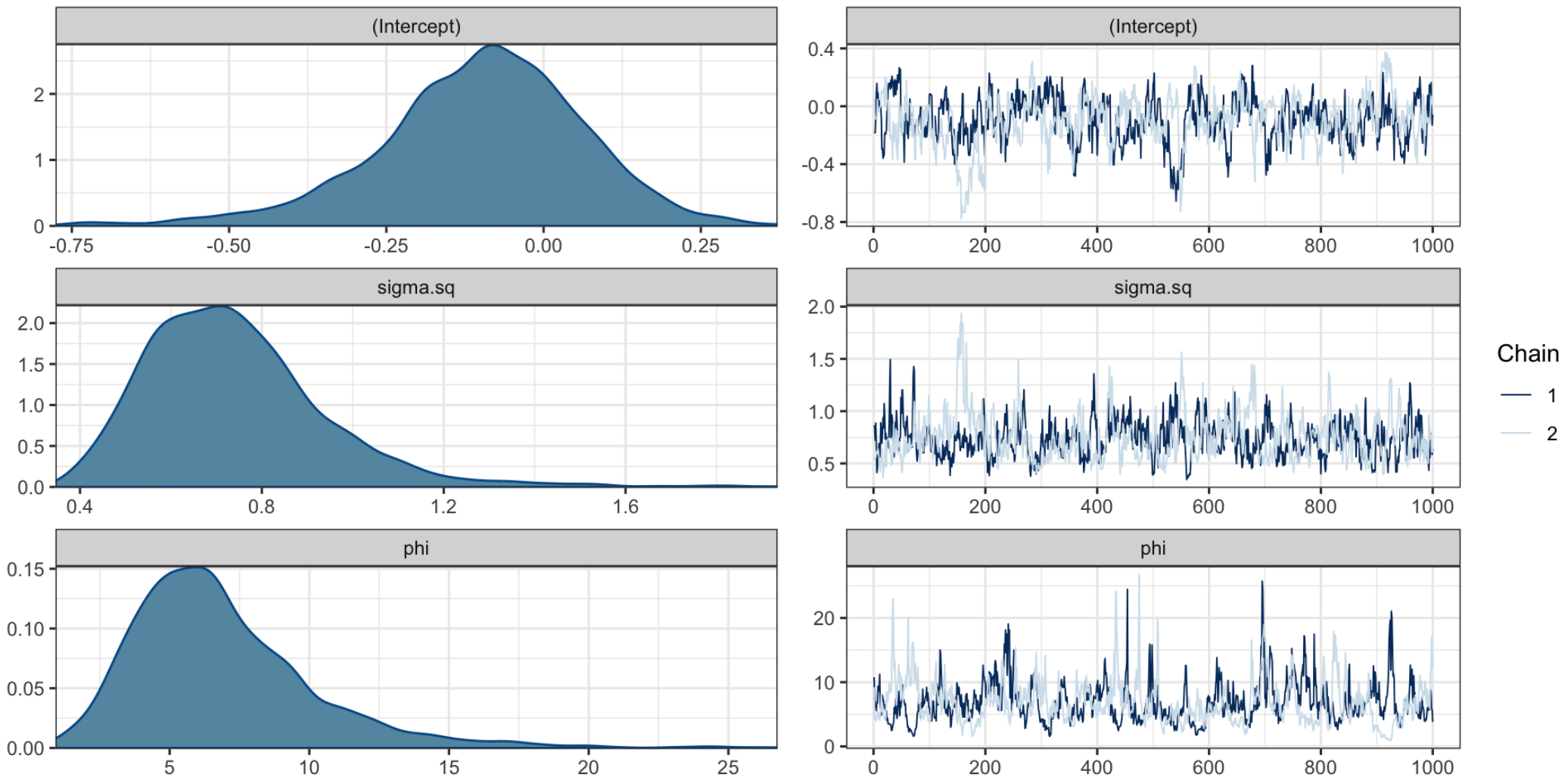
$$\beta_0 \sim N(0, 1)$$
$$\phi \sim \text{Unif}\left(\frac{3}{6}, \frac{3}{1/12}\right)$$
$$\sigma^2 \sim \text{Inv-Gamma}(2, 1)$$

Model fitting

```
m = gpglm(
  cases~1, family="poisson",
  data = polio, coords = c("year"),
  cov_model = "exponential",
  starting=list(
    "beta"=0, "phi"=3/2, "sigma.sq"=1, "w"=0
  ),
  tuning=list(
    "beta"=0.5, "phi"=0.5, "sigma.sq"=0.5, "w"=0.5
  ),
  priors=list(
    "beta.Normal"=list(0,1),
    "phi.unif"=c(3/6, 3/(1/12)),
    "sigma.sq.ig"=c(2, 1)
  ),
  n_batch = 100,
  batch_len = 100,
  verbose = FALSE
)
```

Model diagnostics

```
1 plot(m, thin=5)
```



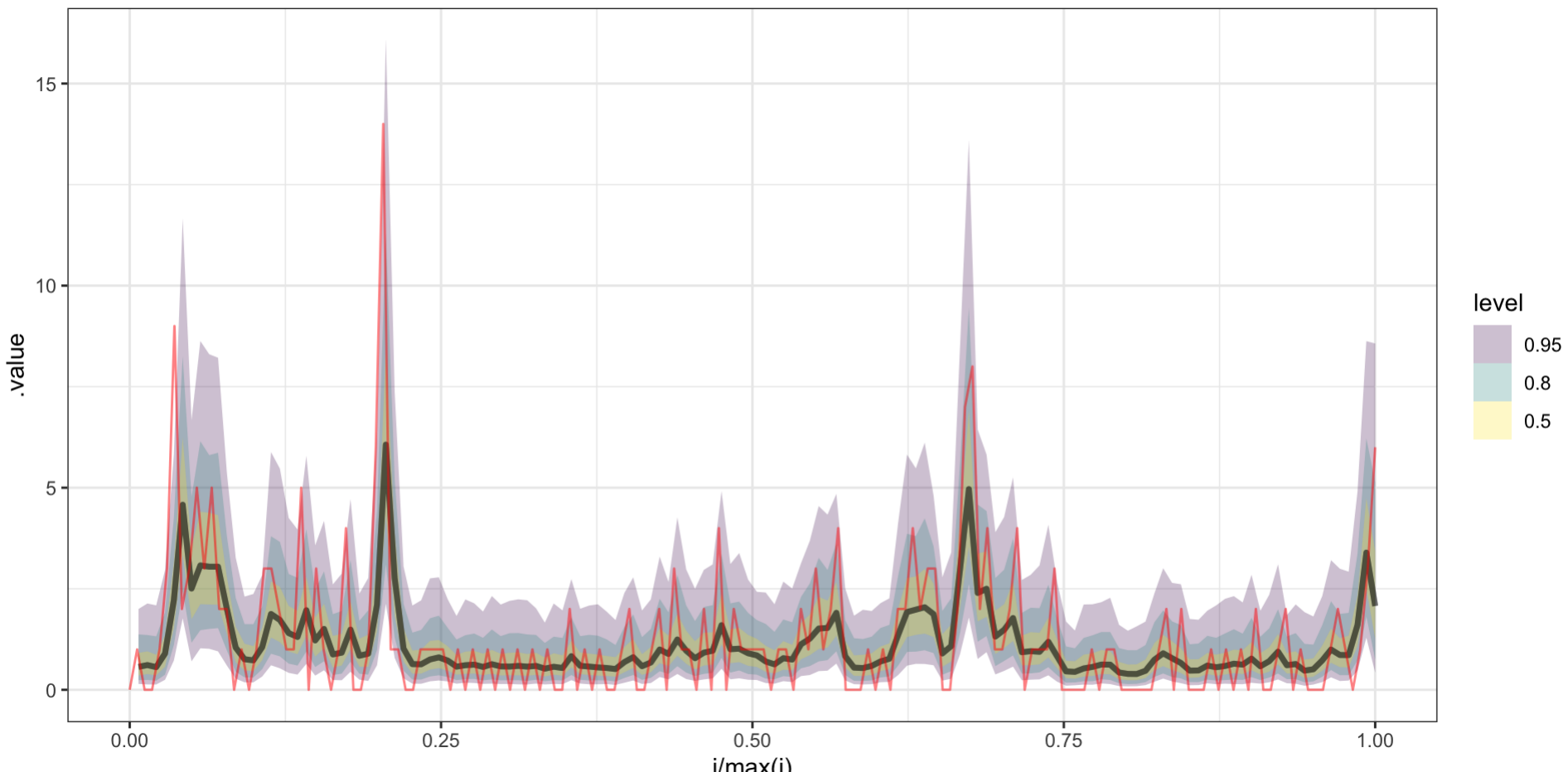
Model fit

```
newdata = data.frame(
  year = seq(1970, 1984, by=0.1) |> jitter()
)
(p = predict(m, newdata=newdata, coords="year", thin=5))
```

```
# A draws_matrix: 1000 iterations, 2 chains, and 282 variables
    variable
draw   w[1]   w[2]  w[3]  w[4] w[5] w[6]  w[7] w[8]
  1   0.035 -1.455 -2.21 -1.48 1.86  2.0 0.172 1.66
  2  -0.435 -1.488 -0.95  0.12 1.02  2.4 0.947 1.96
  3   0.310 -0.438 -1.08  0.20 0.77  1.4 0.747 0.98
  4   0.050 -0.824 -0.54 -0.50 0.60  1.7 0.562 0.64
  5  -0.451 -0.960 -0.61  0.20 1.16  1.7 0.019 0.89
  6  -0.643 -0.907 -1.07 -1.08 0.76  1.8 0.091 0.79
  7  -0.331  0.101 -0.50 -0.94 0.22  1.0 0.422 0.23
  8  -0.292  0.102 -0.57 -0.86 0.40  1.9 1.205 0.22
  9  -0.582  0.193  1.11  0.47 1.05  1.2 0.463 0.98
  10 -0.472 -0.092 -0.43  0.76 1.21  2.9 0.625 1.04
# ... with 1990 more draws, and 274 more variables
```

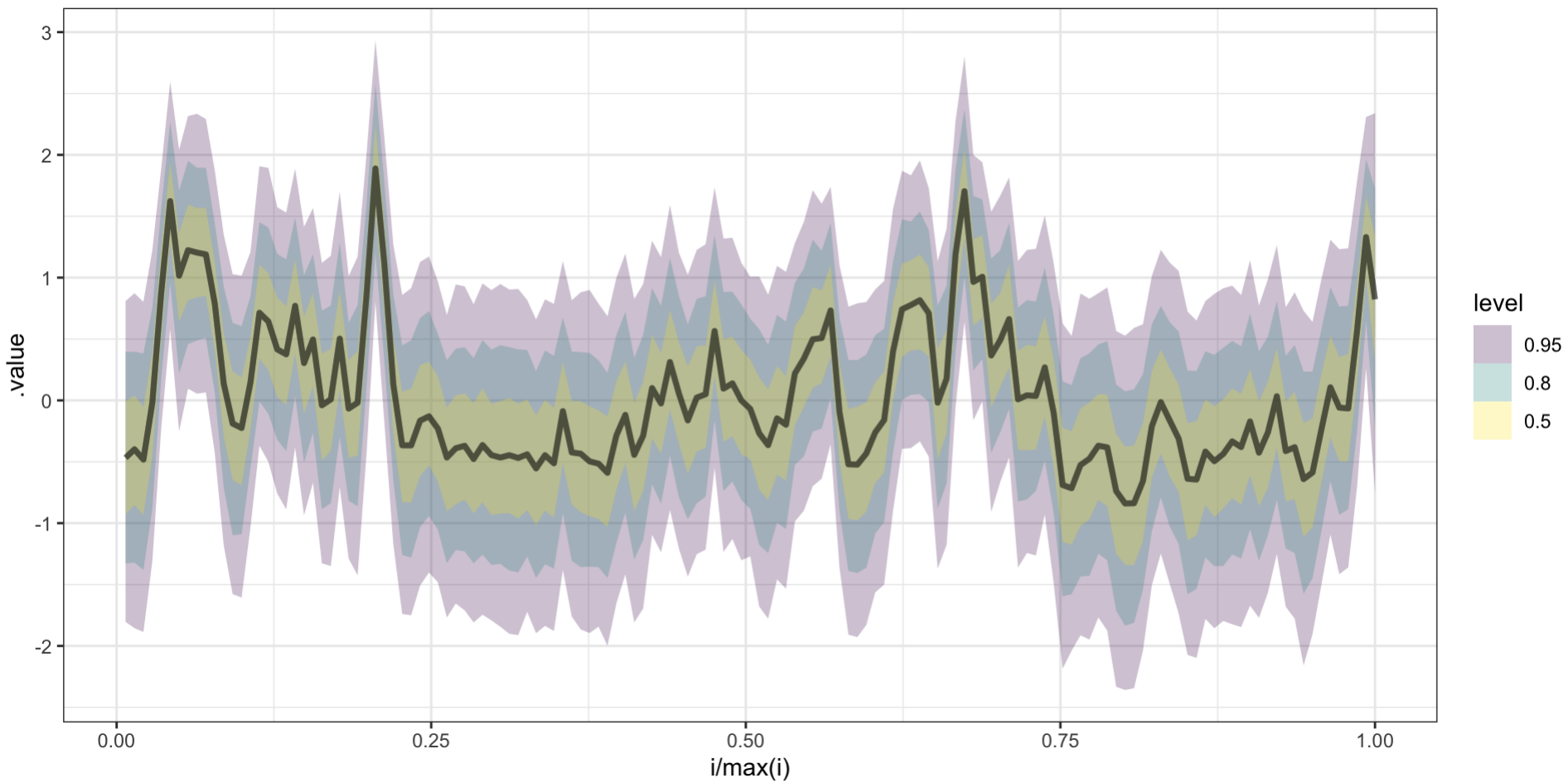
Predicted y

```
p |>
  tidybayes::gather_draws(y[i]) |>
  ggplot2::ggplot(ggplot2::aes(x=i/max(i),y=.value)) +
    tidybayes::stat_lineribbon(alpha=0.25) +
    geom_line(data=polio, aes(x=(year-min(year))/(max(year)-min(year)), y=cases), color='red', a
```



Predicted w

```
p |>
  tidybayes::gather_draws(w[i]) |>
  ggplot2::ggplot(ggplot2::aes(x=i/max(i),y=.value)) +
    tidybayes::stat_lineribbon(alpha=0.25)
```



Spatial data in 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**
- `raster` - classes and tools for handling spatial raster data.
- `sf` - Combines the functionality of `sp`, `rgdal`, and `rgeos` into a single package based on tidy simple features.
- `stars` - Reading, manipulating, writing and plotting spatiotemporal arrays (rasters)
- `terra` - Methods for spatial data analysis with vector (points, lines, polygons) and raster (grid) data. Replaces `raster`.

See more - Spatial task view

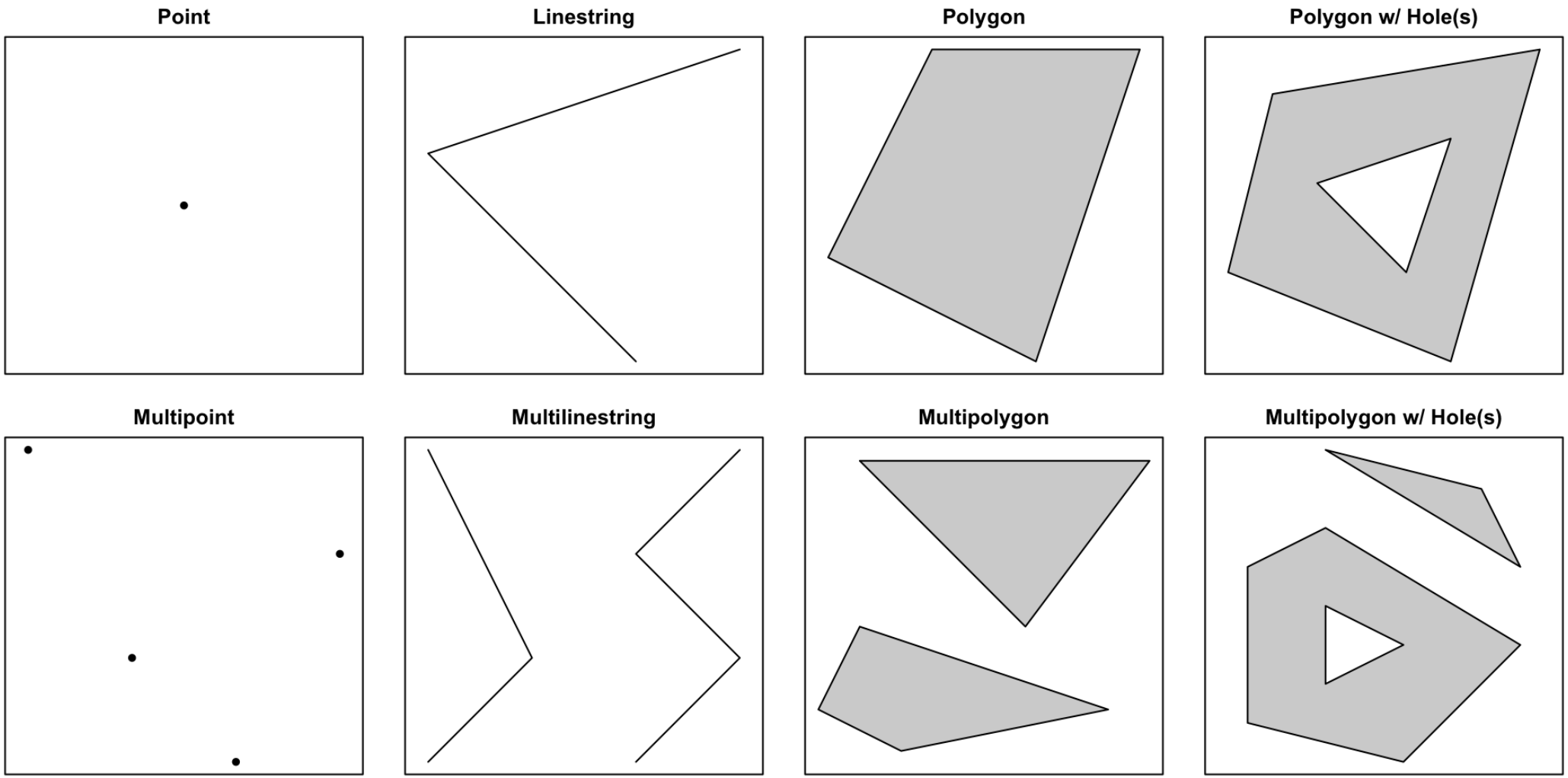
Installing `sf`

This is the hardest part of using the `sf` package, difficulty comes from its dependence on several external libraries (`geos`, `gdal`, `proj`, and `udunits2`).

- *Windows* - installing from source works when Rtools is installed (system requirements are downloaded from rwinlib)
- *MacOS* - install dependencies via homebrew: `gdal2`, `geos`, `proj`, `udunits2`.
- *Linux* - Install development packages for GDAL (>= 2.0.0), GEOS (>= 3.3.0), Proj.4 (>= 4.8.0), and udunits2 from your package manager of choice.

More specific details are included in the README on github.

Simple Features



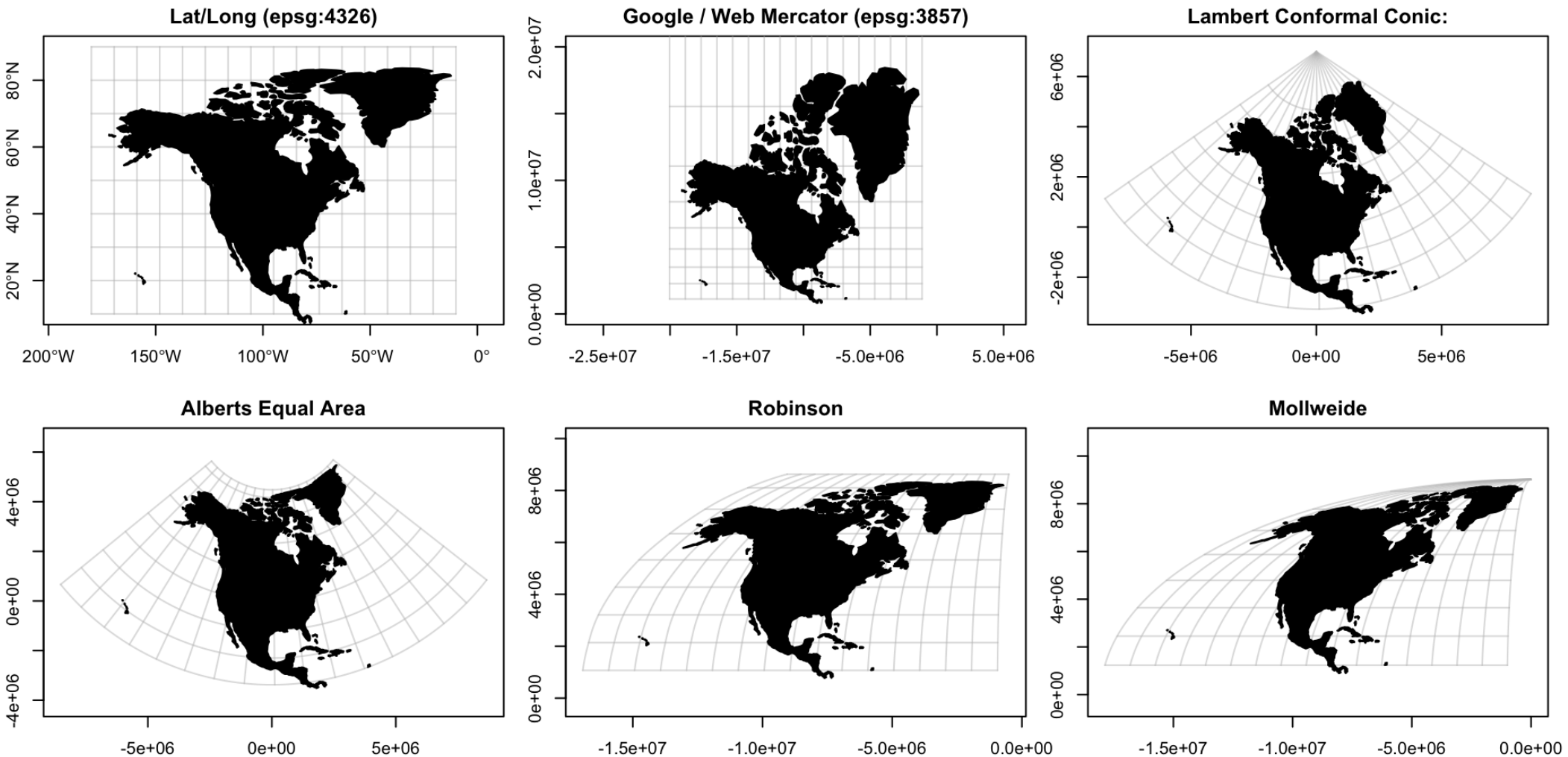
Reading, writing, and converting

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

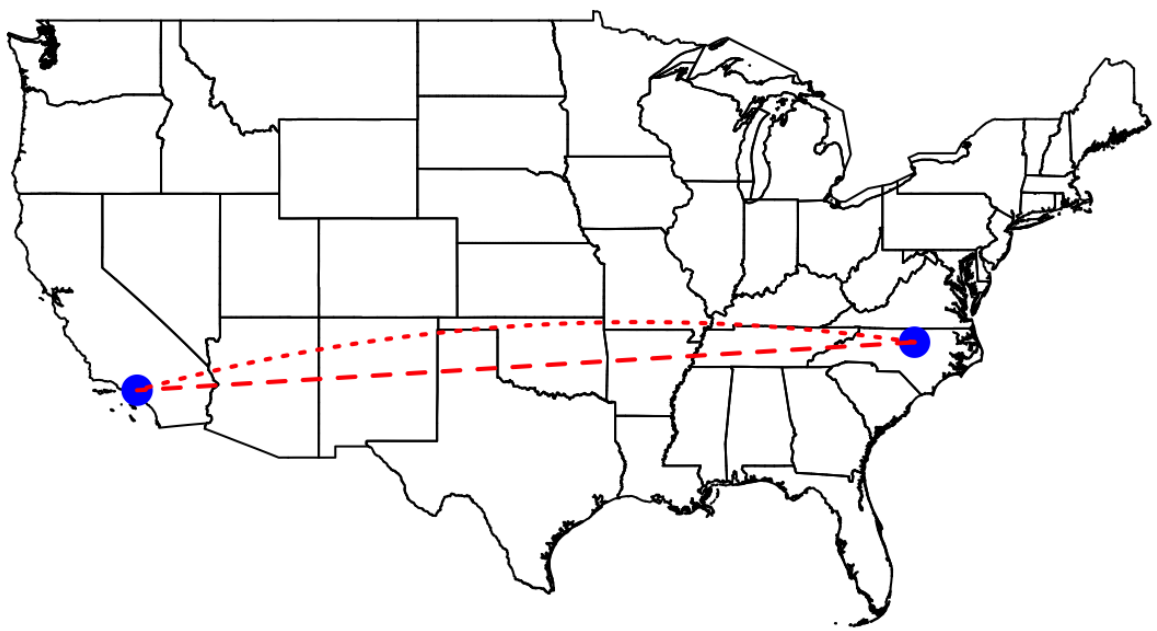
See sf vignette #2 - Reading, Writing and Converting Simple Features.

Geospatial data in the real world

Projections

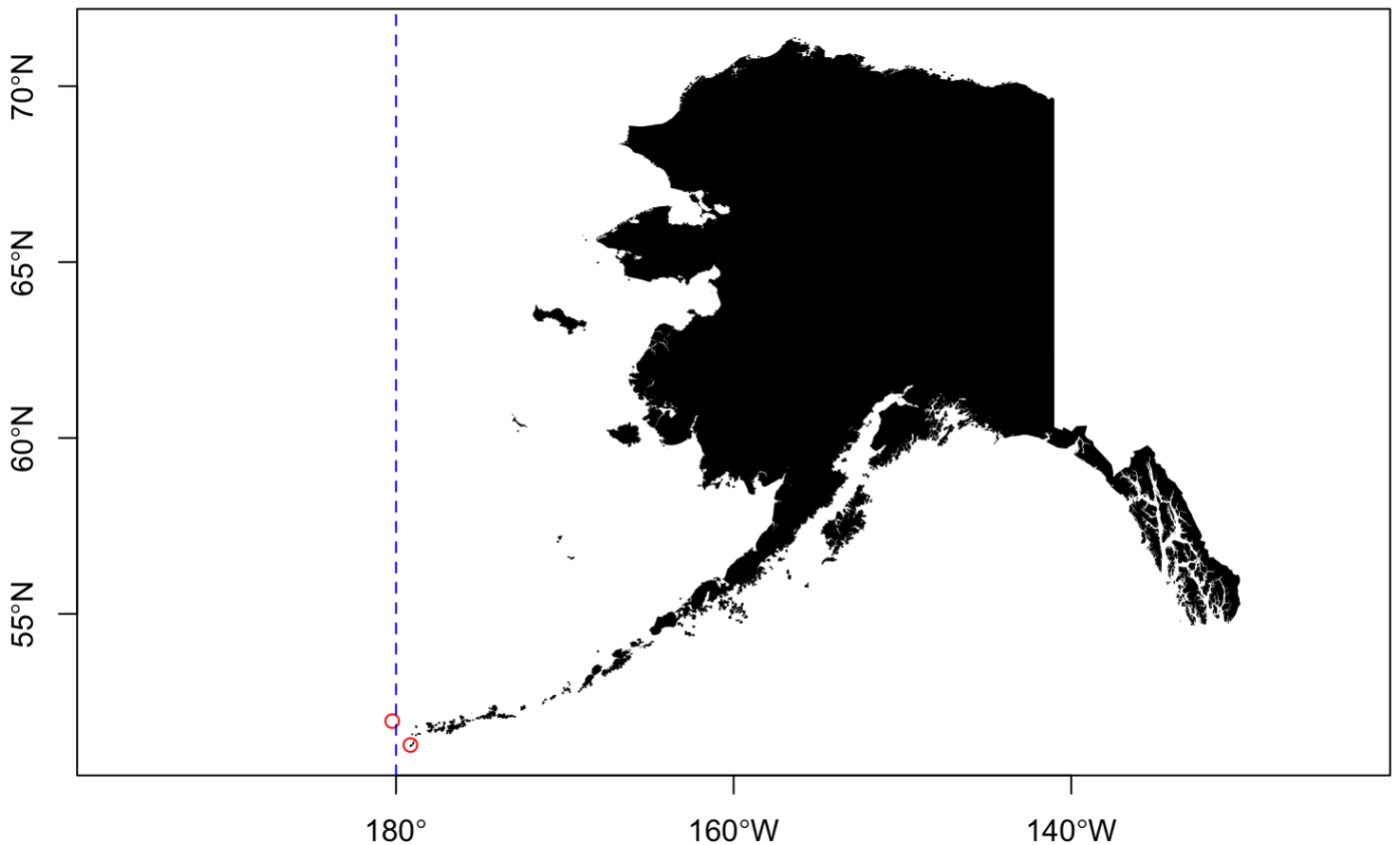


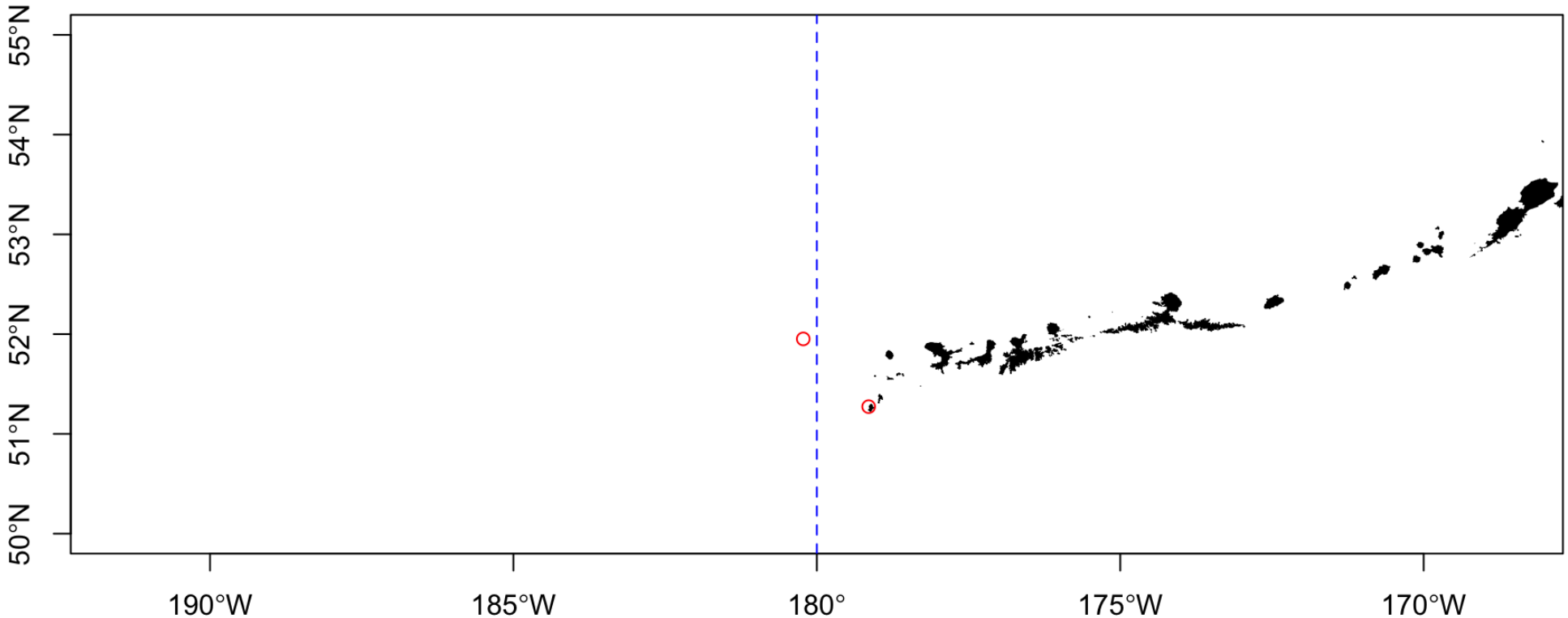
Distance on a Sphere



Dateline

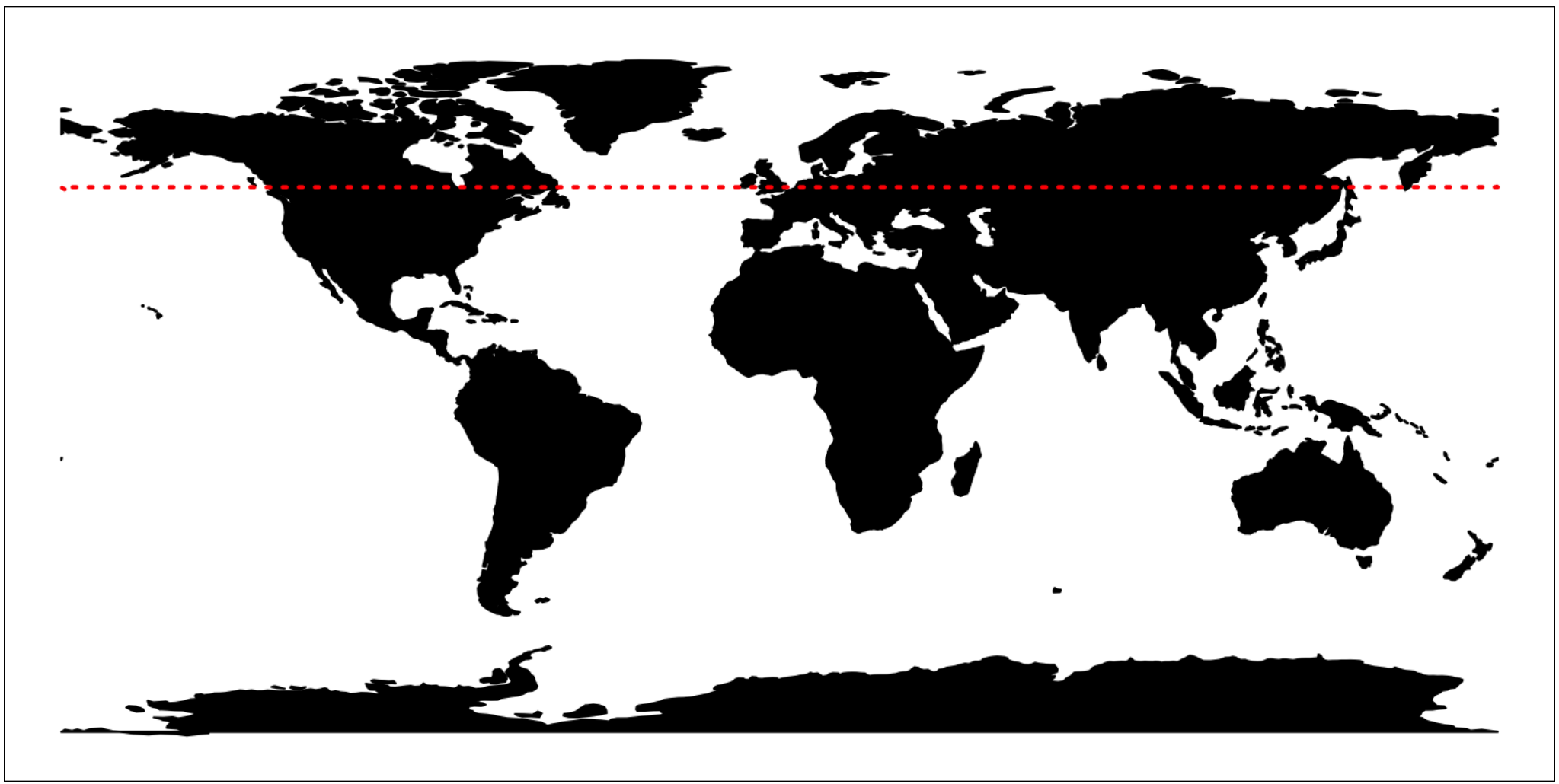
How long is the flight between the Western most and the Eastern most points in the US?





Sta 344/644 - Fall 2023

```r
path = geosphere::gcIntermediate(
  c(179.776, 51.952), c(-179.146, 51.273),
  n=50, addStartEnd=TRUE
)
```



Using `sf`

Example data

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

```
nc
```

```
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
    <dbl>     <dbl>      <dbl> <chr> <chr>  <chr> <chr>
 1 0.112       1.61       1994 NC    Ashe … 37009 37
 2 0.0616      1.35       1996 NC    Alleg… 37005 37
 3 0.140       1.77       1998 NC    Surry… 37171 37
 4 0.0891      1.43       1999 NC    Gates… 37073 37
 5 0.0687      4.43       2000 NC    Curri… 37053 37
 6 0.119       1.40       2001 NC    Stoke… 37169 37
 7 0.0626      2.11       2002 NC    Camde… 37029 37
 8 0.115       1.46       2003 NC    Warre… 37185 37
 9 0.143       2.40       2004 NC    North… 37131 37
10 0.0925      1.81       2005 NC    Hertf… 37091 37
# ℹ 90 more rows
```

```
# ℹ 2 more variables: SQUARE_MIL <dbl>,
#   geometry <MULTIPOLYGON [°]>
```

```
1 air
```

```
Simple feature collection with 940 features and 16 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -176.646 ymin: 17.70156 xmax: -64.80172 ymax: 71.28545
Geodetic CRS:  NAD83
# A tibble: 940 × 17
   AIRPRTX010 FEATURE ICAO  IATA  AIRPT_NAME     CITY  STATE
        <dbl> <chr>   <chr> <chr> <chr>          <chr> <chr>
 1          0 AIRPORT KGON  GON   GROTON-NEW LO… GROT… CT
 2          3 AIRPORT K6S5  6S5   RAVALLI COUNT… HAMI… MT
 3          4 AIRPORT KMHV  MHV   MOJAVE AIRPORT MOJA… CA
 4          6 AIRPORT KSEE  SEE   GILLESPIE FIE… SAN … CA
 5          7 AIRPORT KFPR  FPR   ST LUCIE COUN… FORT… FL
 6          8 AIRPORT KRYY  RYY   COBB COUNTY-M… ATLA… GA
 7         10 AIRPORT KMKL  MKL   MC KELLAR-SIP… JACK… TN
 8         11 AIRPORT KCCR  CCR   BUCHANAN FIEL… CONC… CA
 9         13 AIRPORT KJYO  JYO   LEESBURG EXEC… LEES… VA
10         15 AIRPORT KCAD  CAD   WEXFORD COUNT… CADI… MI
# ℹ 930 more rows
# ℹ 10 more variables: STATE_FIPS <chr>, COUNTY <chr>,
#   FIPS <chr>, TOT_ENP <dbl>, LATITUDE <dbl>,
#   LONGITUDE <dbl>, ELEV <dbl>, ACT_DATE <chr>
```

```
1 hwy
```

```
Simple feature collection with 233 features and 3 fields
Geometry type: MULTILINESTRING
Dimension:     XY
Bounding box:  xmin: -7472582 ymin: 2911107 xmax: 2443707 ymax: 8208428
Projected CRS: NAD83 / UTM zone 15N
# A tibble: 233 × 4
   ROUTE_NUM DIST_MILES DIST_KM                       geometry
   <chr>          <dbl>   <dbl>          <MULTILINESTRING [m]>
 1 I10           2449.   3941.   ((-1881200 4072307, -187992…
 2 I105            20.8    33.4  ((-1910156 5339585, -191013…
 3 I110            41.4    66.6  ((1054139 3388879, 1054287 …
 4 I115             1.58    2.55 ((-1013796 5284243, -101313…
 5 I12             85.3   137.   ((680741.7 3366581, 682709.…
 6 I124             1.73    2.79 ((1201467 3906285, 1201643 …
 7 I126             3.56    5.72 ((1601502 3829718, 1602136 …
 8 I129             3.1     4.99 ((217446 4705389, 217835.1 …
 9 I135            96.3   155.   ((96922.97 4313125, 96561.8…
10 I15           1436.   2311    ((-882875.7 5602902, -88299…
# ℹ 223 more rows
```

sf structure

```
1 str(nc)
```

```
sf [100 × 9] (S3: sf/tbl_df/tbl/data.frame)
 $ AREA      : num [1:100] 0.1118 0.0616 0.1402 0.0891 0.0687 ...
 $ PERIMETER : num [1:100] 1.61 1.35 1.77 1.43 4.43 ...
 $ COUNTYP010: num [1:100] 1994 1996 1998 1999 2000 ...
 $ STATE     : chr [1:100] "NC" "NC" "NC" "NC" ...
 $ COUNTY    : chr [1:100] "Ashe County" "Alleghany County" "Surry County" "Gates County" ...
 $ FIPS      : chr [1:100] "37009" "37005" "37171" "37073" ...
 $ STATE_FIPS: chr [1:100] "37" "37" "37" "37" ...
 $ SQUARE_MIL: num [1:100] 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 [1:3] "XY" "MULTIPOLYGON" "sfg"
 - 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" ...
```

`sf` classes

```
1 class(nc)
```

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

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

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

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

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

Projections

```
1 st_crs(nc)
```

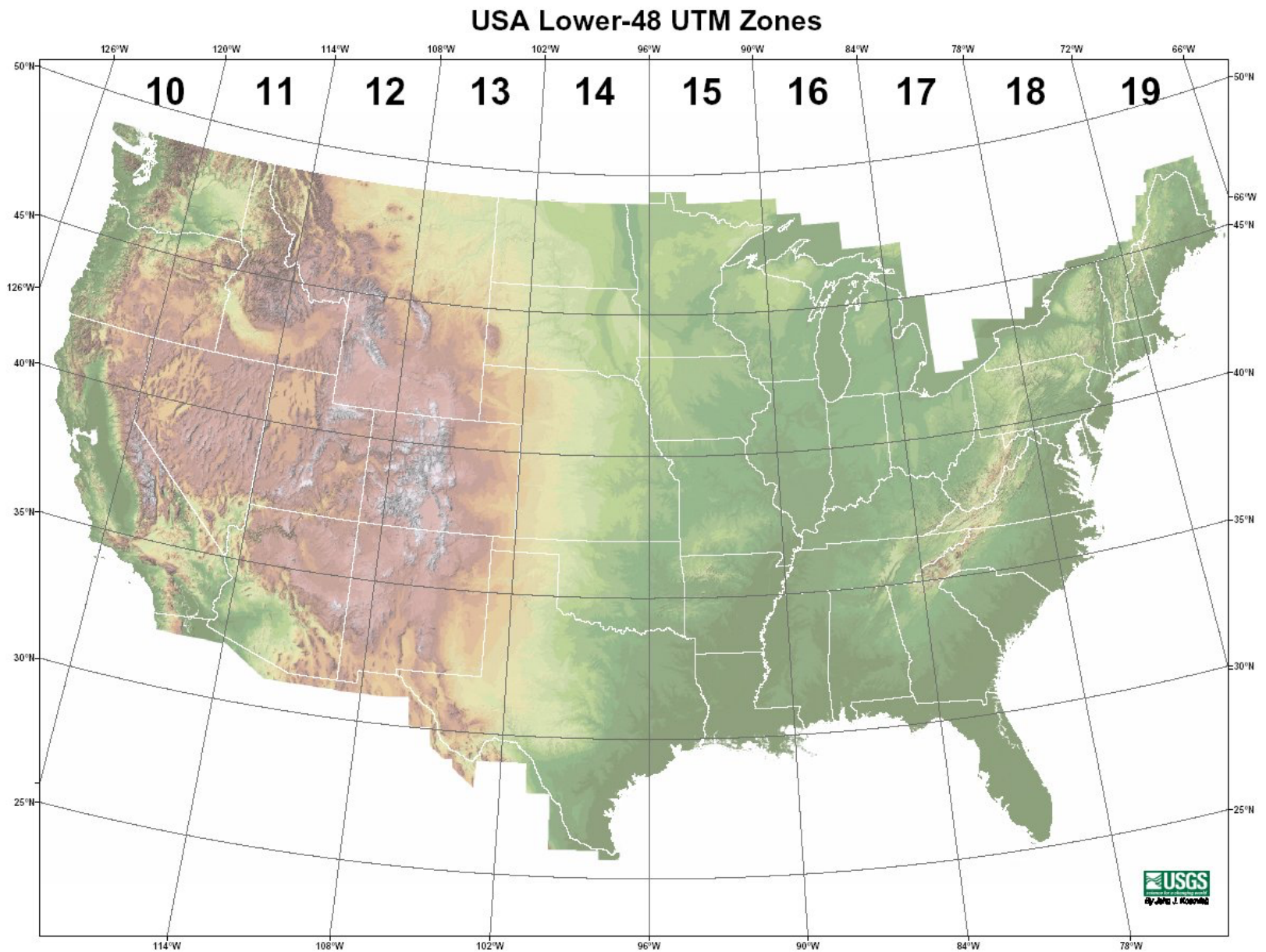
```
Coordinate Reference System:
  User input: NAD83
  wkt:
GEOGCRS["NAD83",
    DATUM["North American Datum 1983",
        ELLIPSOID["GRS 1980",6378137,298.257222101,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["latitude",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["longitude",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    ID["EPSG",4269]]
```

For EPSG details see here

```r
st_crs(hwy)
```

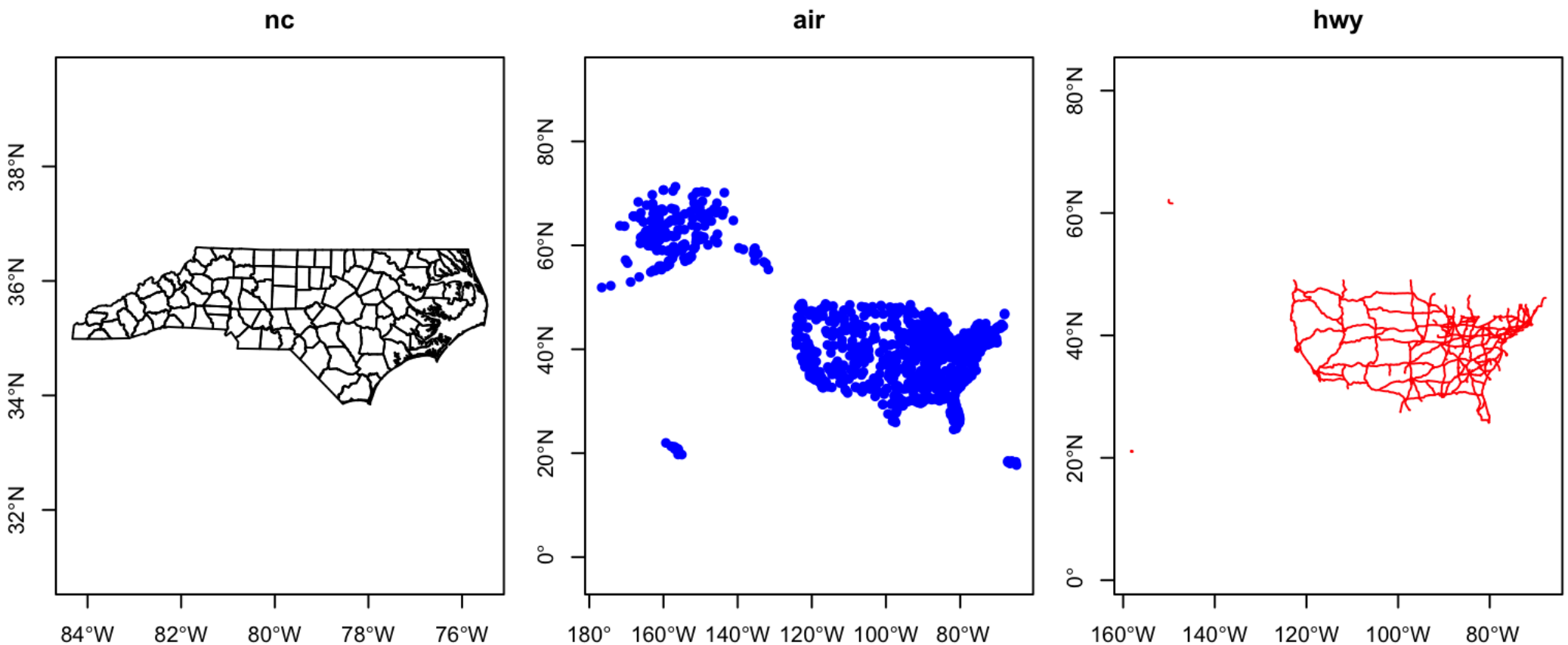
```
Coordinate Reference System:
  User input: NAD83 / UTM zone 15N
  wkt:
PROJCRS["NAD83 / UTM zone 15N",
    BASEGEOGCRS["NAD83",
        DATUM["North American Datum 1983",
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4269]],
    CONVERSION["UTM zone 15N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",-93,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1]
```

UTM Zones


USA Lower-48 UTM Zones
10
11
12
13
14
15
16
17
18
19
126°W
120°W
114°W
108°W
102°W
96°W
90°W
84°W
78°W
72°W
66°W
50°N
45°N
40°N
35°N
30°N
25°N
USGS

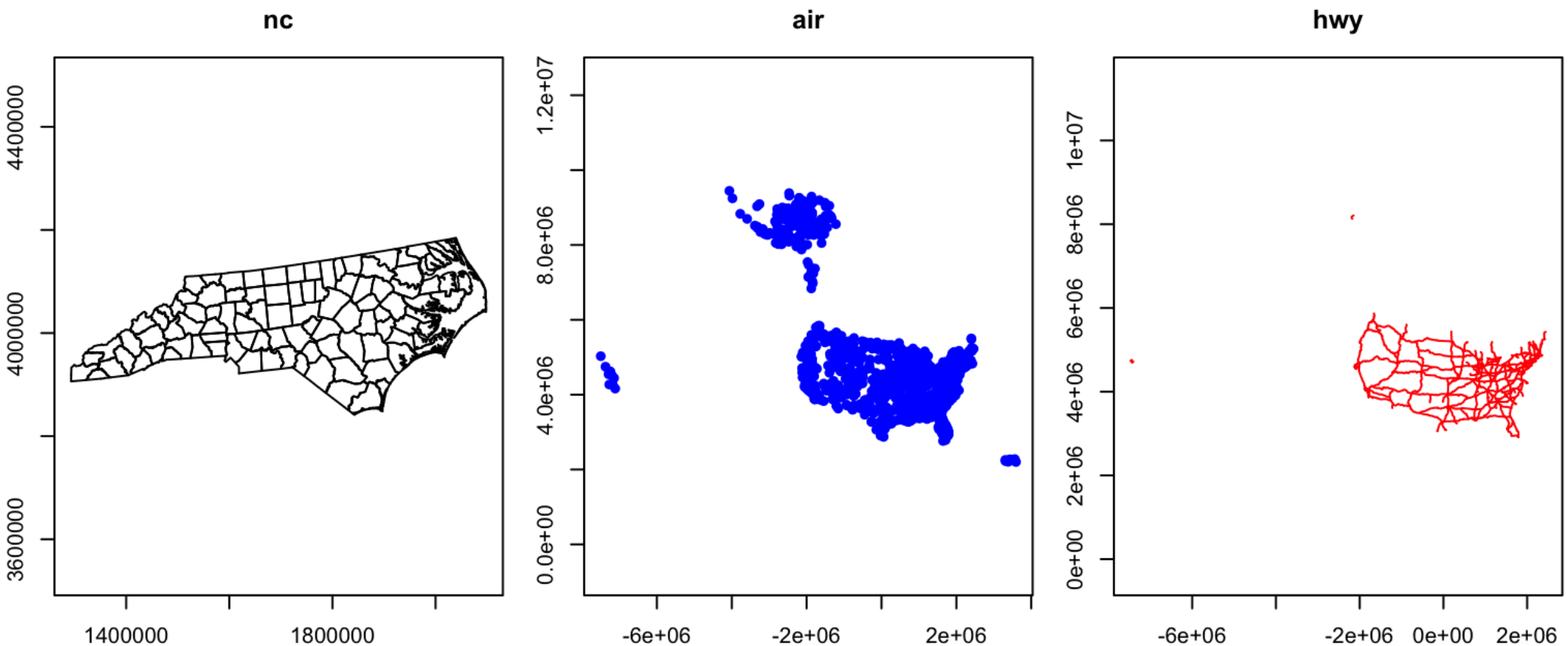
Lat/Long

```
1 nc_ll = nc
2 air_ll = air
3 hwy_ll = st_transform(hwy, st_crs(nc))
```



UTM

```
1 nc_utm = st_transform(nc, st_crs(hwy))
2 air_utm = st_transform(air, st_crs(hwy))
3 hwy_utm = hwy
```



Comparison

Lat/Long | UTM

```
par(mar=c(3,5,0.1,0.1), las=1)
plot(st_geometry(nc_ll),  axes=TRUE)
plot(st_geometry(hwy_ll), col="red", add=TRUE)
plot(st_geometry(air_ll), pch=16, col="blue", add=TRUE)
```