Maps & layers

Hadley Wickham

Assistant Professor / Dobelman Family Junior Chair

Department of Statistics / Rice University

July 2010

1. Introduction to map data
2. Map projections
3. Loading & converting shape files
4. Plot layers

Caveats

R is not a GIS, but it can do a lot, particularly with spatial modelling (e.g. the `sp` **package)**

I'm not an expert in either cartography or maps. I do know how to work with spatial data in R.

Getting started

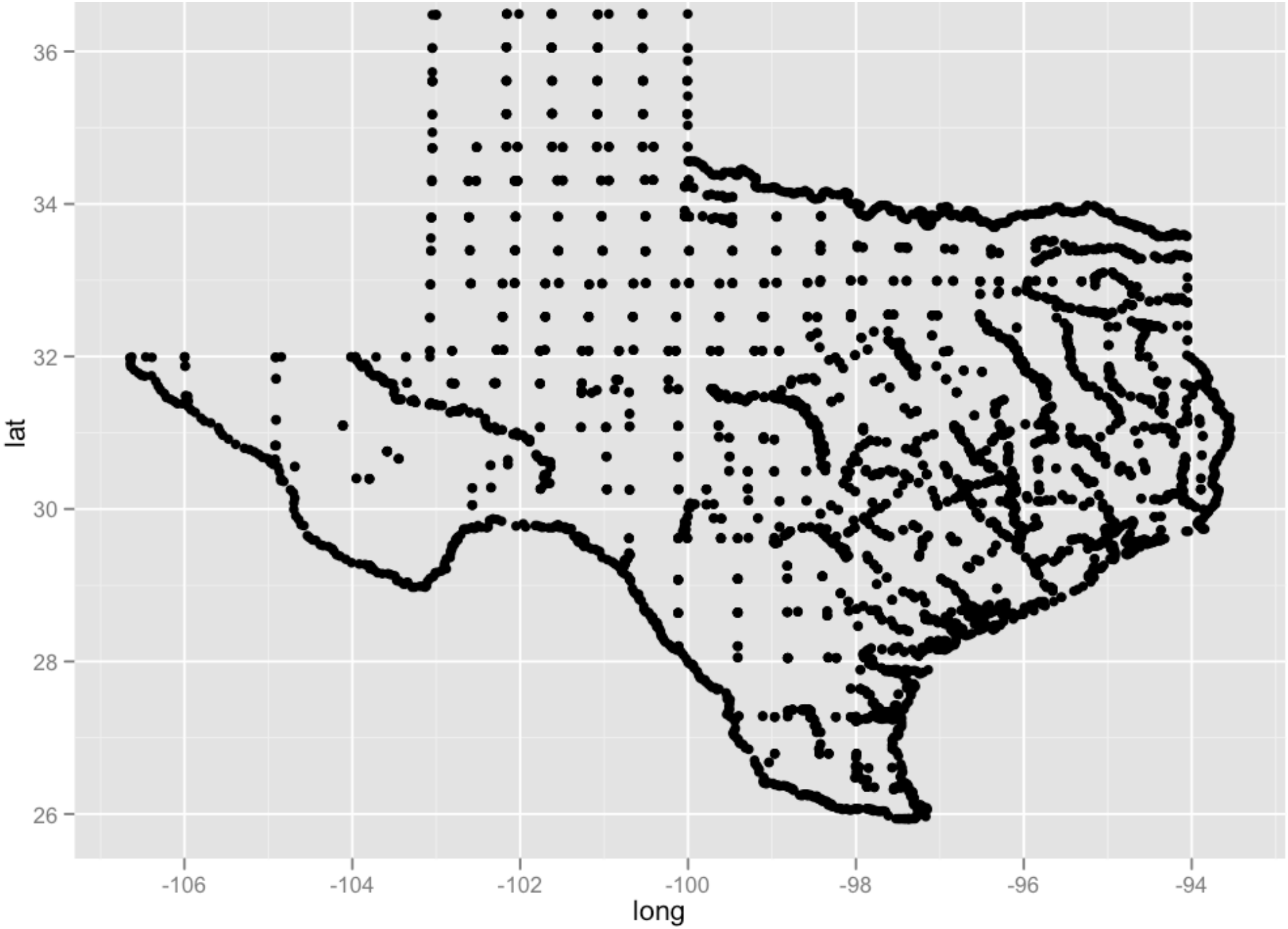
```
library(ggplot2)

# County borders:
borders <- read.csv("tx-borders.csv")
# Country centres:
centres <- read.csv("tx-centres.csv")

# Big cities:
data(us.cities, package = "maps")
cities <- subset(us.cities,
  country.etc == "TX" & pop > 5e5)
```

```
> head(borders, 20)
        long      lat group order state   county
1  -95.75271 31.53560     1     1 texas anderson
2  -95.76989 31.55852     1     2 texas anderson
3  -95.76416 31.58143     1     3 texas anderson
4  -95.72979 31.58143     1     4 texas anderson
5  -95.74698 31.61008     1     5 texas anderson
6  -95.72405 31.63873     1     6 texas anderson
7  -95.75271 31.67311     1     7 texas anderson
8  -95.76989 31.66738     1     8 texas anderson
9  -95.77563 31.63300     1     9 texas anderson
10 -95.79855 31.63873     1    10 texas anderson
11 -95.81000 31.67311     1    11 texas anderson
12 -95.79282 31.71321     1    12 texas anderson
13 -95.82146 31.70748     1    13 texas anderson
14 -95.87876 31.71321     1    14 texas anderson
15 -95.87876 31.75332     1    15 texas anderson
16 -95.91887 31.78197     1    16 texas anderson
17 -95.94751 31.78197     1    17 texas anderson
18 -95.98762 31.80489     1    18 texas anderson
19 -95.98189 31.83927     1    19 texas anderson
20 -95.99908 31.86218     1    20 texas anderson
```

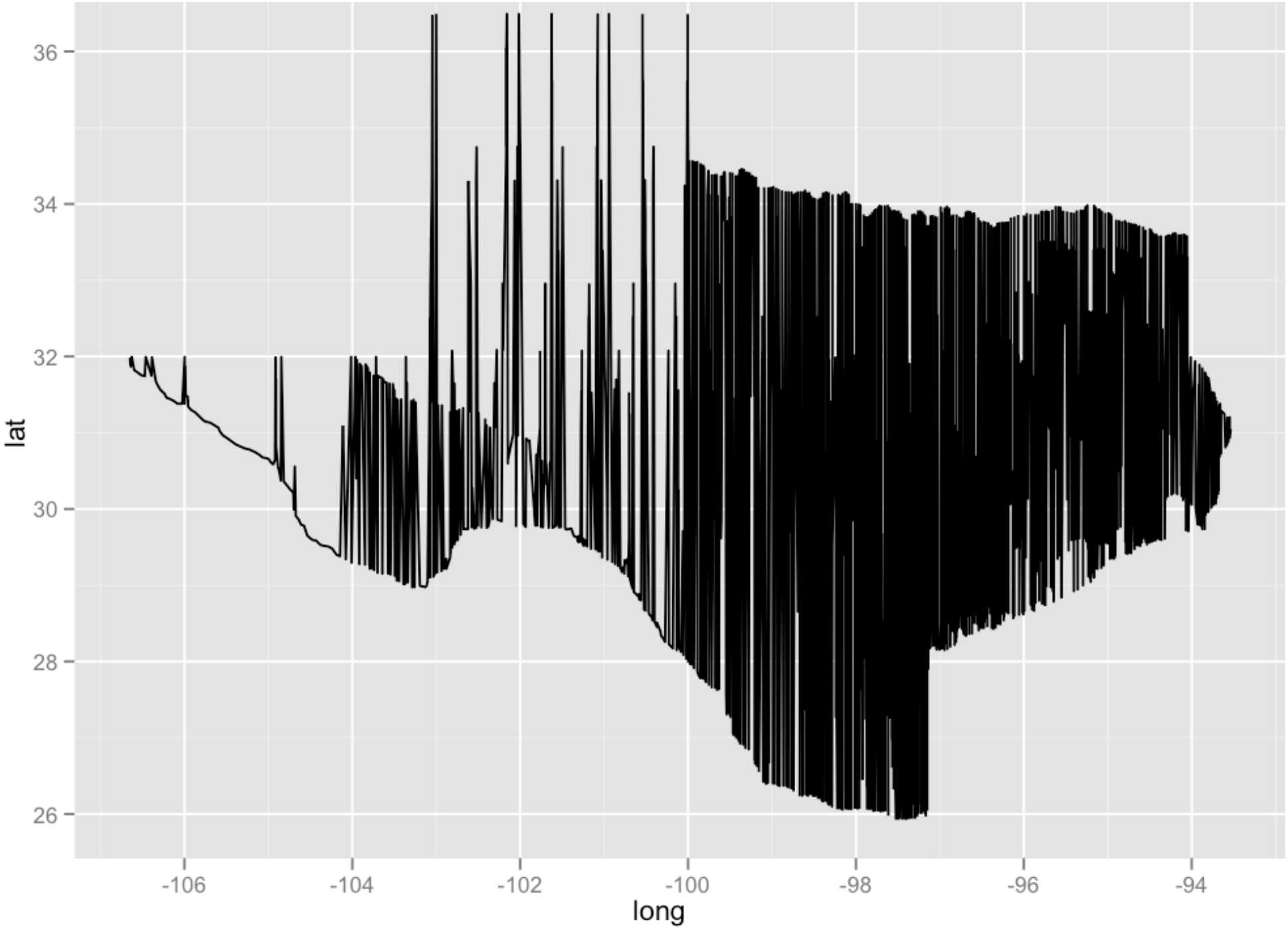
State and **county** are obvious. So are **lat** and **long**.

What are **group** and **order**? What does each each row represent?

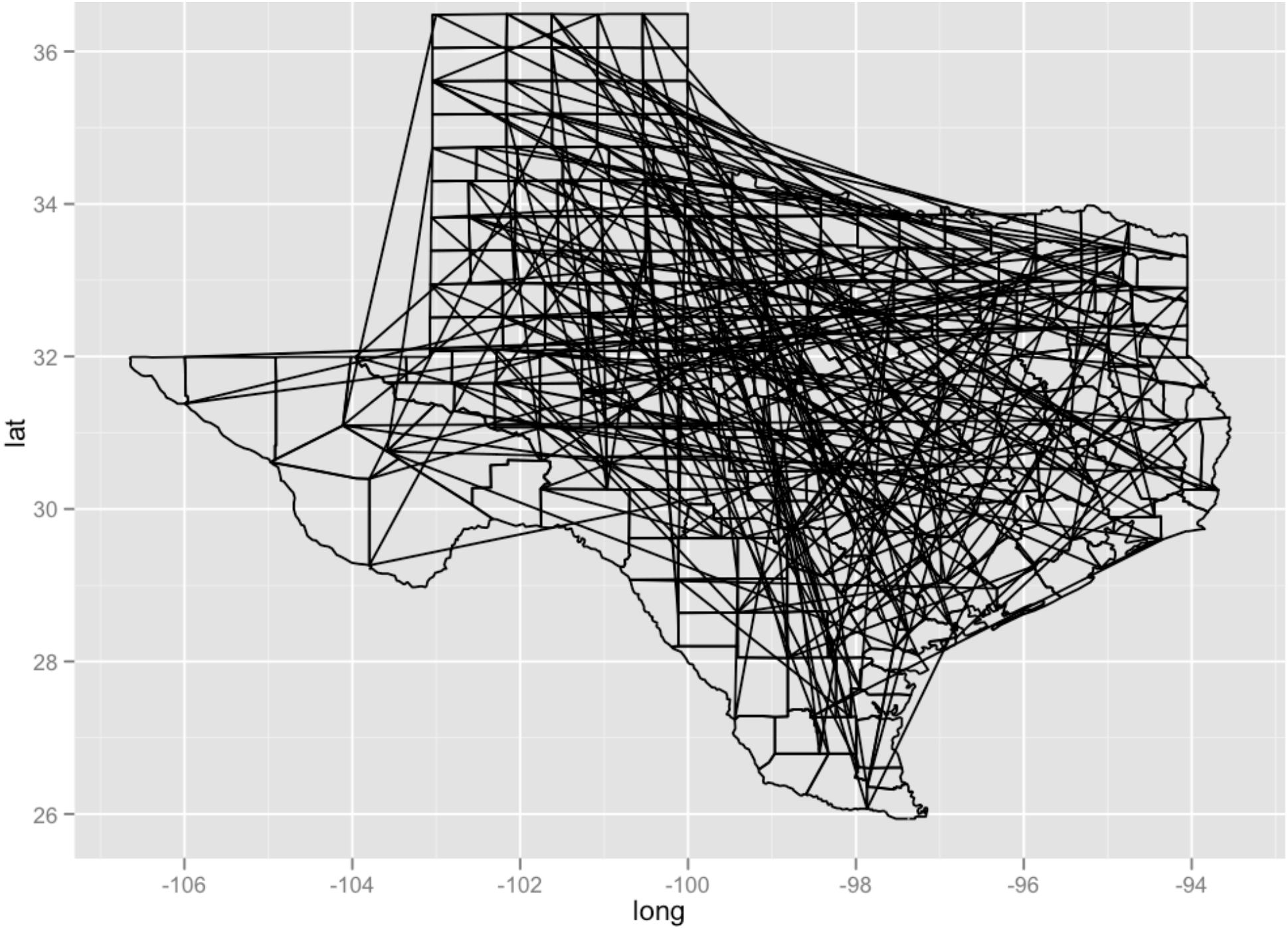
```
qplot(long, lat, data = borders)
```



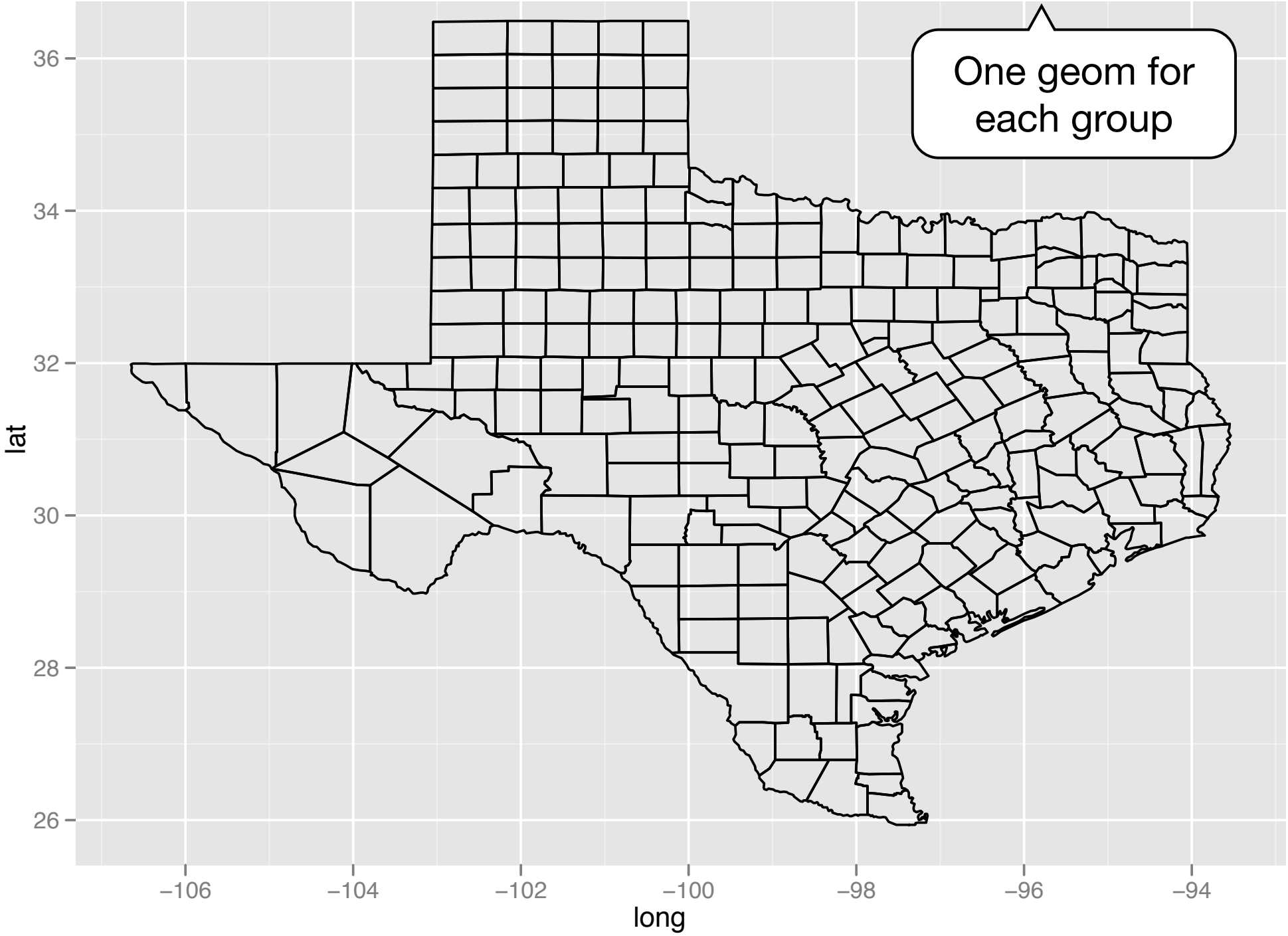
```
qplot(long, lat, data = borders, geom = "line")
```



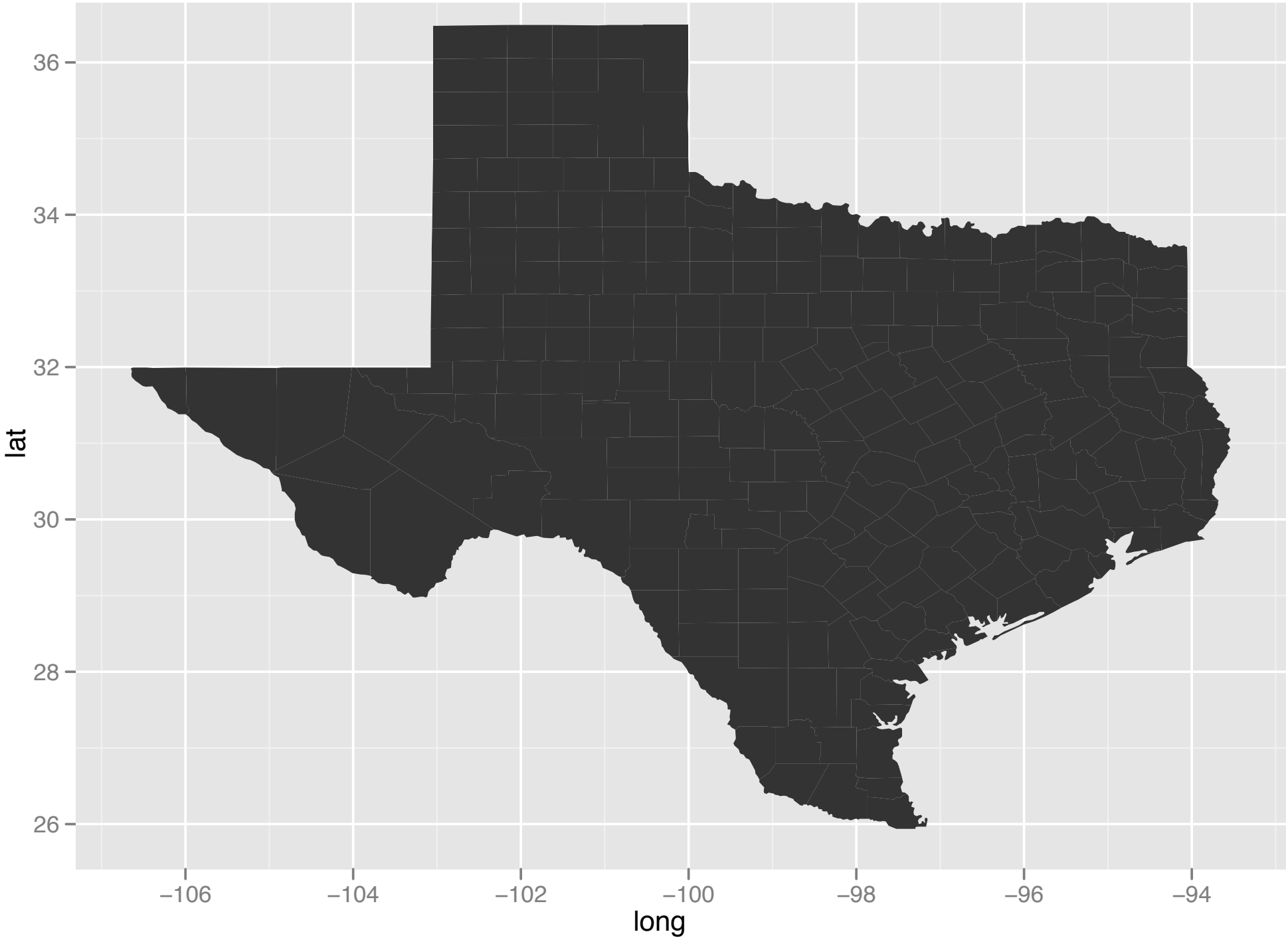
```
qplot(long, lat, data = borders, geom = "path")
```



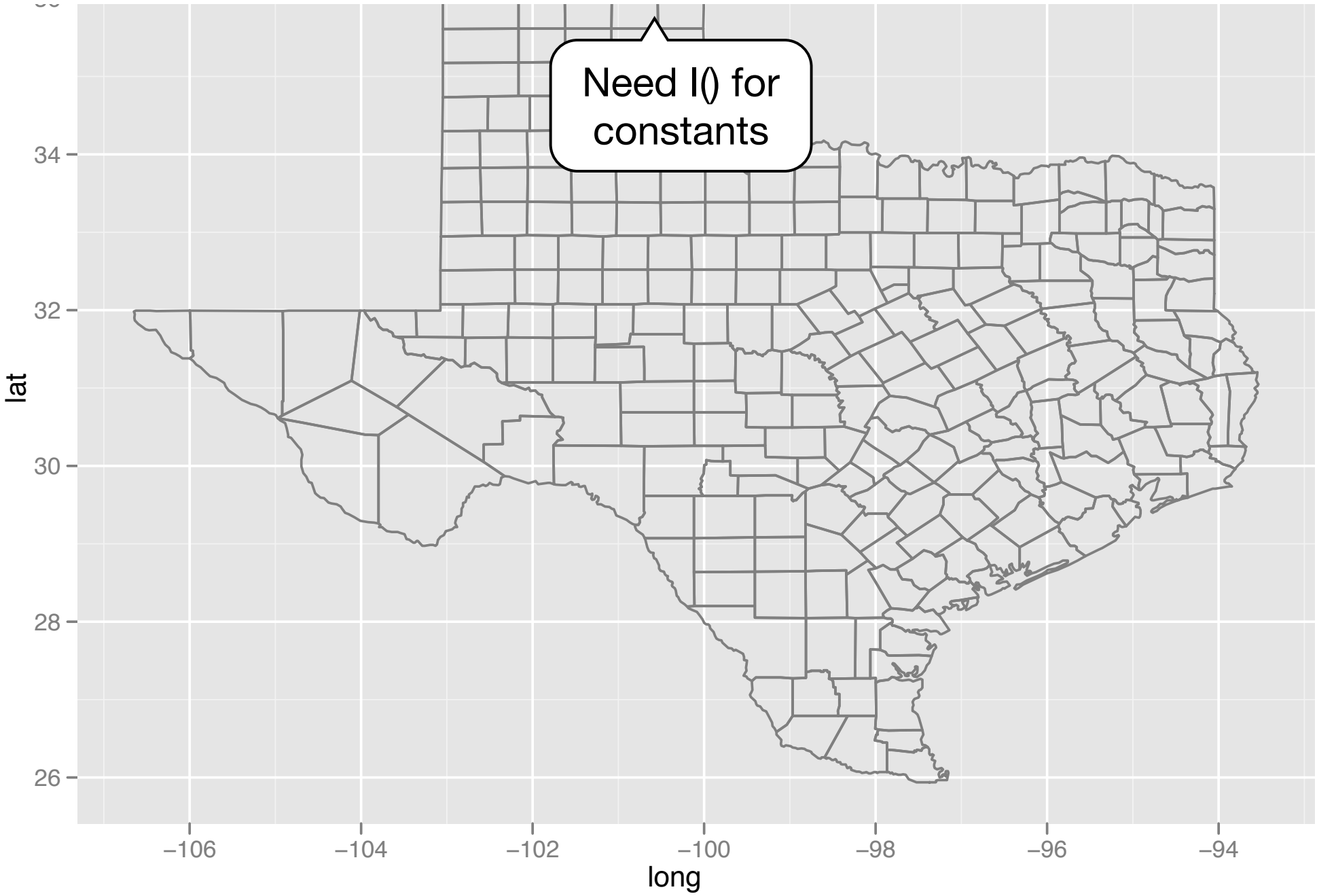
```
qplot(long, lat, data = borders, geom = "path", group = group)
```



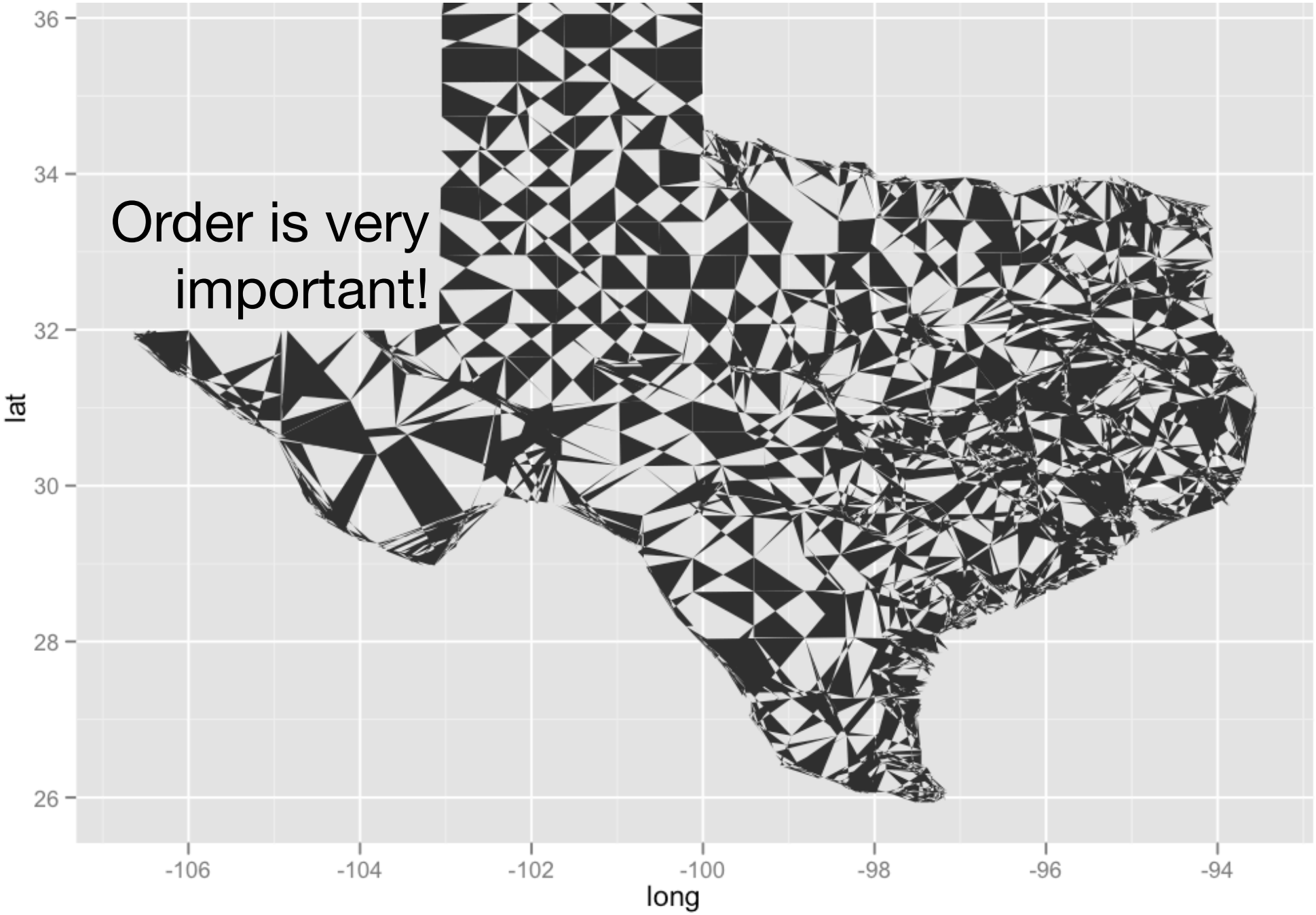
```
qplot(long, lat, data = borders, geom = "polygon", group = group)
```



```
qplot(long, lat, data = borders, geom = "polygon",
  group = group, colour = I("grey50"), fill = I(NA))
```



```
borders <- borders[sample(nrow(borders)), ]
qplot(long, lat, data = borders, geom = "polygon", group = group)
```



```
borders <- borders[order(borders$order), ]
qplot(long, lat, data = borders, geom = "polygon", group = group)
```

Map projections

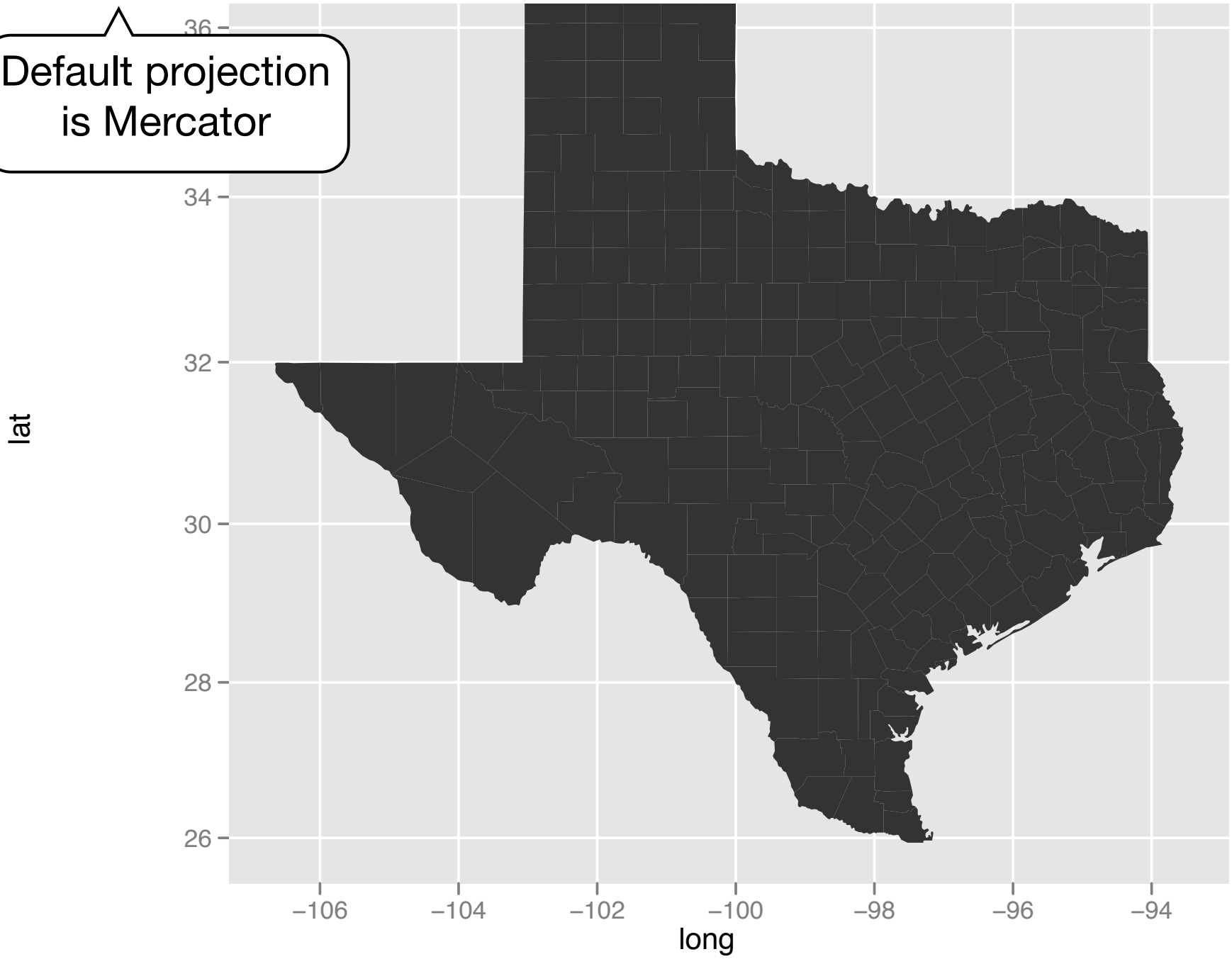
Map projections

Deal with the fact that the earth is a sphere, but our paper/screen is a flat rectangle. Modifies **coordinate system**.

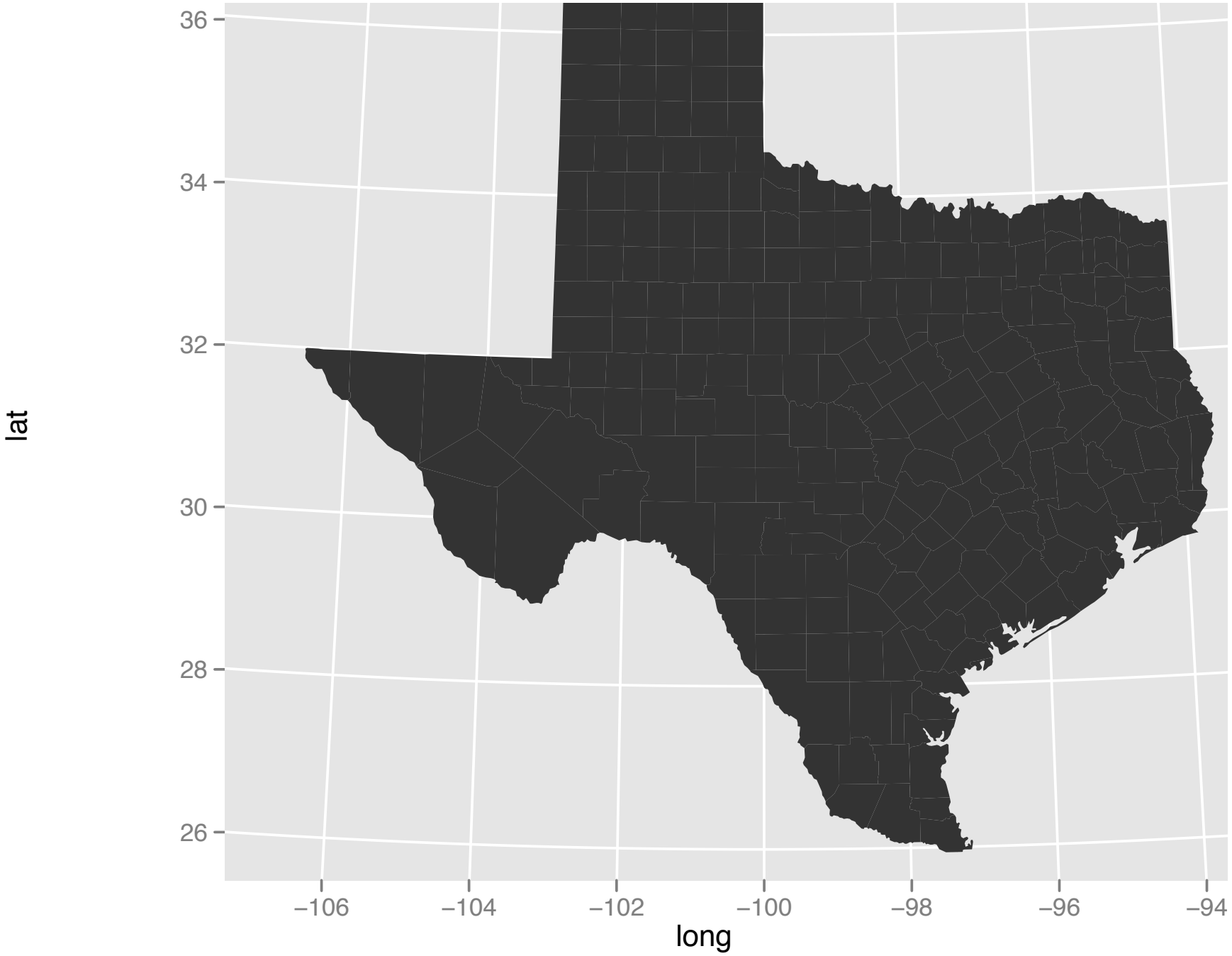
All projections done by **mapproj** in **maptools** package. See its documentation for more details. Must compromise between accurate depiction of angle, area, bearing, distance and scale.

```
qplot(long, lat, data = borders, geom = "polygon", group = group) +
  coord_map()
```

Default projection is Mercator



```
qplot(long, lat, data = borders, geom = "polygon", group = group)
  coord_map("lambert", lat0 = 27.416667, lat1 = 34.916667)
```



WHO shapefiles

```
library(maptools)

world <- readShapeSpatial("who-world/general_2008.shp")
plot(world)

# Extract meta data
meta <- as.data.frame(world)

# Convert to data frame for use in ggplot2
gpclibPermit()
worlddf <- fortify(world, region = "ISO_2_CODE")

head(worlddf)
str(worlddf)
```

```
> head(worlddf, 20)
       long      lat order  hole piece group id
1  1.445833 42.60194     1 FALSE     1  AD.1 AD
2  1.481111 42.64722     2 FALSE     1  AD.1 AD
3  1.491944 42.65361     3 FALSE     1  AD.1 AD
4  1.556389 42.65639     4 FALSE     1  AD.1 AD
5  1.563056 42.65555     5 FALSE     1  AD.1 AD
6  1.698333 42.62611     6 FALSE     1  AD.1 AD
7  1.738611 42.61639     7 FALSE     1  AD.1 AD
8  1.781720 42.56996     8 FALSE     1  AD.1 AD
9  1.774722 42.57111     9 FALSE     1  AD.1 AD
10 1.768055 42.57111    10 FALSE     1  AD.1 AD
11 1.753889 42.57000    11 FALSE     1  AD.1 AD
12 1.743055 42.56361    12 FALSE     1  AD.1 AD
13 1.735278 42.55444    13 FALSE     1  AD.1 AD
14 1.725278 42.53000    14 FALSE     1  AD.1 AD
15 1.724444 42.52499    15 FALSE     1  AD.1 AD
16 1.723333 42.51527    16 FALSE     1  AD.1 AD
17 1.723611 42.50944    17 FALSE     1  AD.1 AD
18 1.718889 42.50305    18 FALSE     1  AD.1 AD
19 1.661111 42.47138    19 FALSE     1  AD.1 AD
20 1.654444 42.46777    20 FALSE     1  AD.1 AD
```

Two new variables: **piece** and **hole**.

Geometry of world more complicated: some countries have non-contiguous **pieces**. Lakes (**holes**) important part of some countries.

Data-maps

After lunch we'll see how to add data to these maps. But before that we need to learn a little more about the underlying theory.

Maps are usually made up of multiple layers (borders, rivers, cities). How do we add additional layers to a ggplot2 plot?

Map layers

Layers

Most maps (and many plots) have multiple layers of data. The layers may be from the same or different datasets.

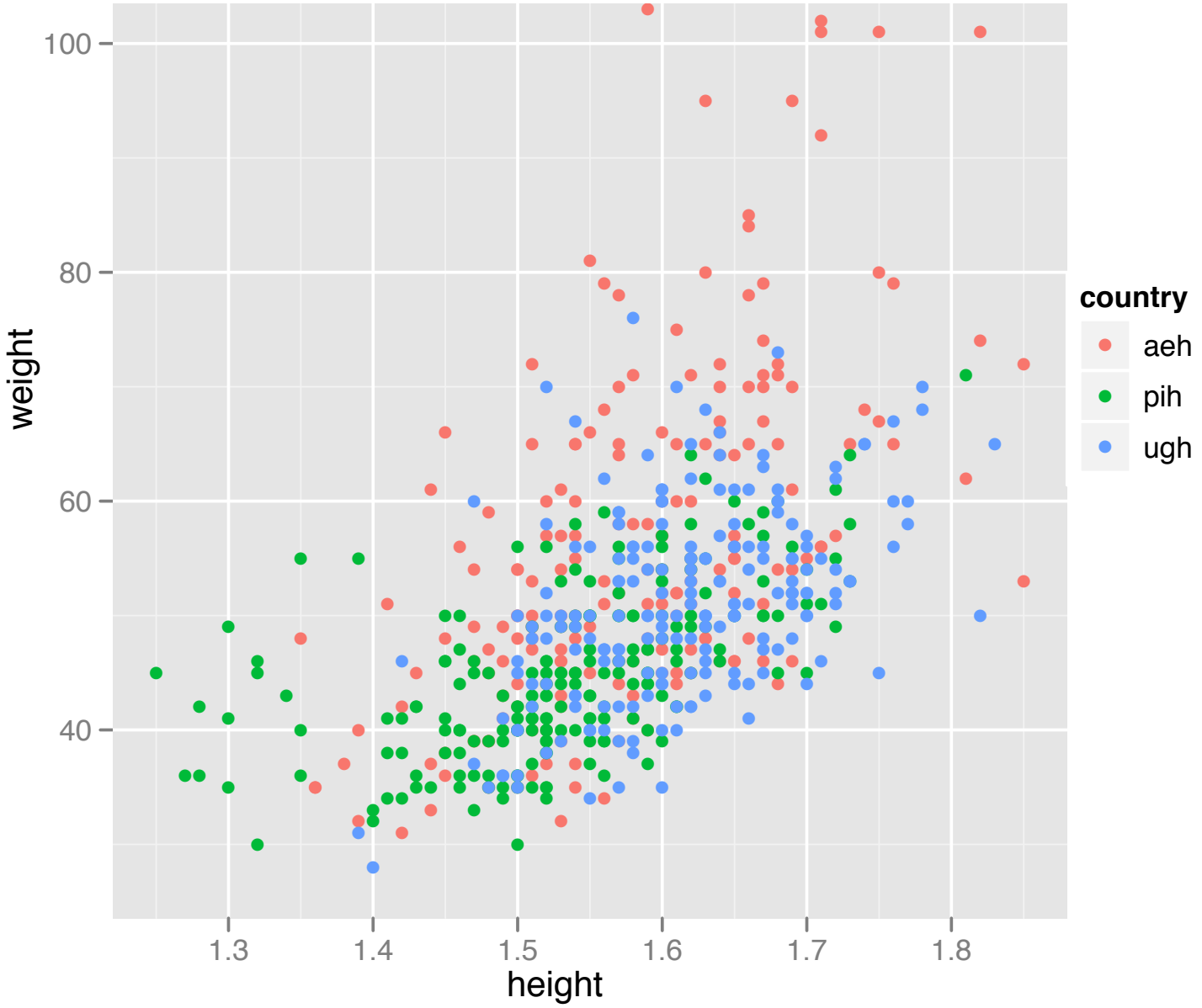
ggplot2 build around this same idea. Very easy to add additional layers to the plot. To do this we need to understand a little more about the underlying theory.

What is a plot?

A plot is composed of:

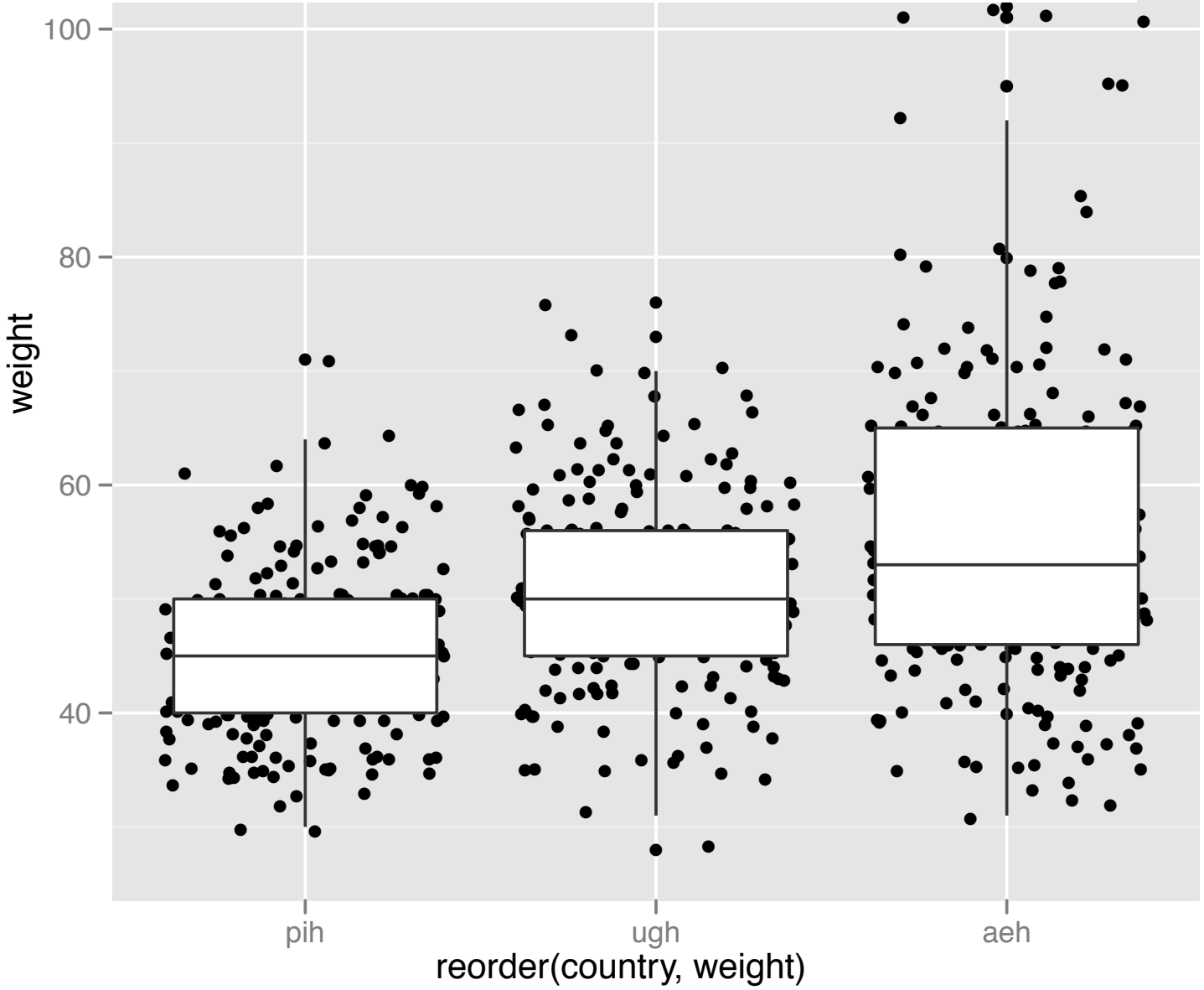
- A default dataset and set of aesthetic mappings
- Multiple layers
- A scale for each aesthetic
- A facetting specification
- A coordinate system

Data: sample. **Mapping**: x = height, y = weight, colour = country. **Layers**: points. **Scales**: x & y position, discrete colour. **Faceting**: none

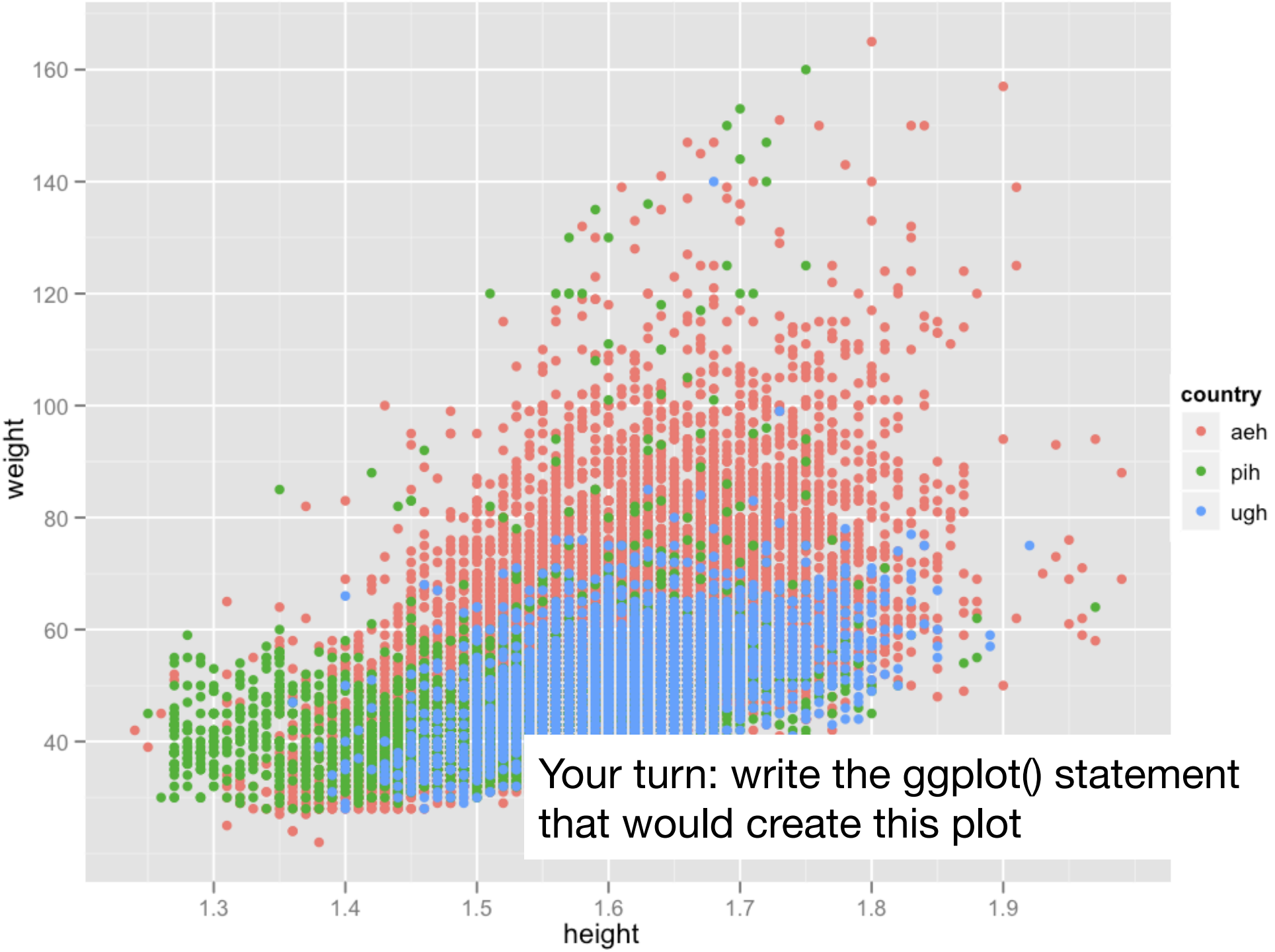


```
qplot(height, weight, data = sample, colour = country)

ggplot(sample, aes(x = height, y = weight, colour = country) +
  geom_point() +
  scale_x_continuous() +
  scale_y_continuous() +
  scale_colour_discrete()

# But we don't need to be quite so verbose.  Scales are
# added automatically and first two aes params are x and y:
ggplot(sample, aes(height, weight, colour = country)) +
  geom_point()
```

Data: sample. **Mapping**: x = country, y = weight.
Layers: jittered points, boxplots.
Scales: x & y position. **Faceting**: none



```
qplot(reorder(country, weight), weight,
  data = sample, geom = c("jitter", "boxplot"))

ggplot(sample, aes(reorder(country, weight), weight)) +
  geom_jitter() +
  geom_boxplot()
```


160
140
120
100
80
60
40
weight
1.3
1.4
1.5
1.6
1.7
1.8
1.9
height
country
aeh
pih
ugh
Your turn: write the ggplot() statement that would create this plot

What is a layer?

- A dataset and aesthetic mappings (to override default)
- A geom
- A stat
- A position adjustment

	Geom	Stat
Scatterplot	point	identity
Histogram	bar	bin
Smooth	line + ribbon	smooth
2d histogram	rect	bin2d

```
# What's the difference between these functions?

wborders <- read.csv("world-borders.csv")
wcentres <- read.csv("world-centres.csv")

qplot(long, lat, data = wborders, geom = "polygon", group = group) +
  coord_map()

ggplot(wborders, aes(long, lat))  +
  geom_polygon(aes(group = group)) +
  coord_map()

qplot(long, lat, data = wborders, geom = "polygon", group = group,
  colour = I("grey50"), fill = I(NA)) +
  coord_map()

ggplot(wborders, aes(long, lat))  +
  geom_polygon(aes(group = group), colour = "grey50", fill = NA) +
  coord_map()

# Once you've figured it out, try and reproduce other plots that we
# created this morning using ggplot()
```

Overriding defaults

Layers can override the default data set and aesthetic mappings. This is useful if we want to put multiple datasets on the same plot.

What does this code do? First think, then run.

```
ggplot(wborders, aes(long, lat))  +
  geom_polygon(aes(group = group)) +
  geom_point(data = wcentres) +
  coord_map()

txborders <- read.csv("tx-borders.csv")
txcentres <- read.csv("tx-centres.csv")

ggplot(txborders, aes(long, lat)) +
  geom_polygon(aes(group = group), fill = NA,
    colour = "grey80") +
  geom_point(data = txcentres) +
  geom_text(data = txcentres, aes(label = county),
    size = 3, angle = 30, hjust = -0.05)
```

Plot default: `aes(mpg, wt)`

Add	`aes(colour = cyl)`	`aes(mpg, wt, colour = cyl)`
Override	`aes(y = disp)`	`aes(mpg, disp)`
Remove	`aes(y = NULL)`	`aes(mpg)`

Learning more

That's the essence of all the more complex plots you might create: add additional layers varying the parameters, aesthetics or data.

See http://had.co.nz/stat405/resources/drills/ggplot2.html for more examples.