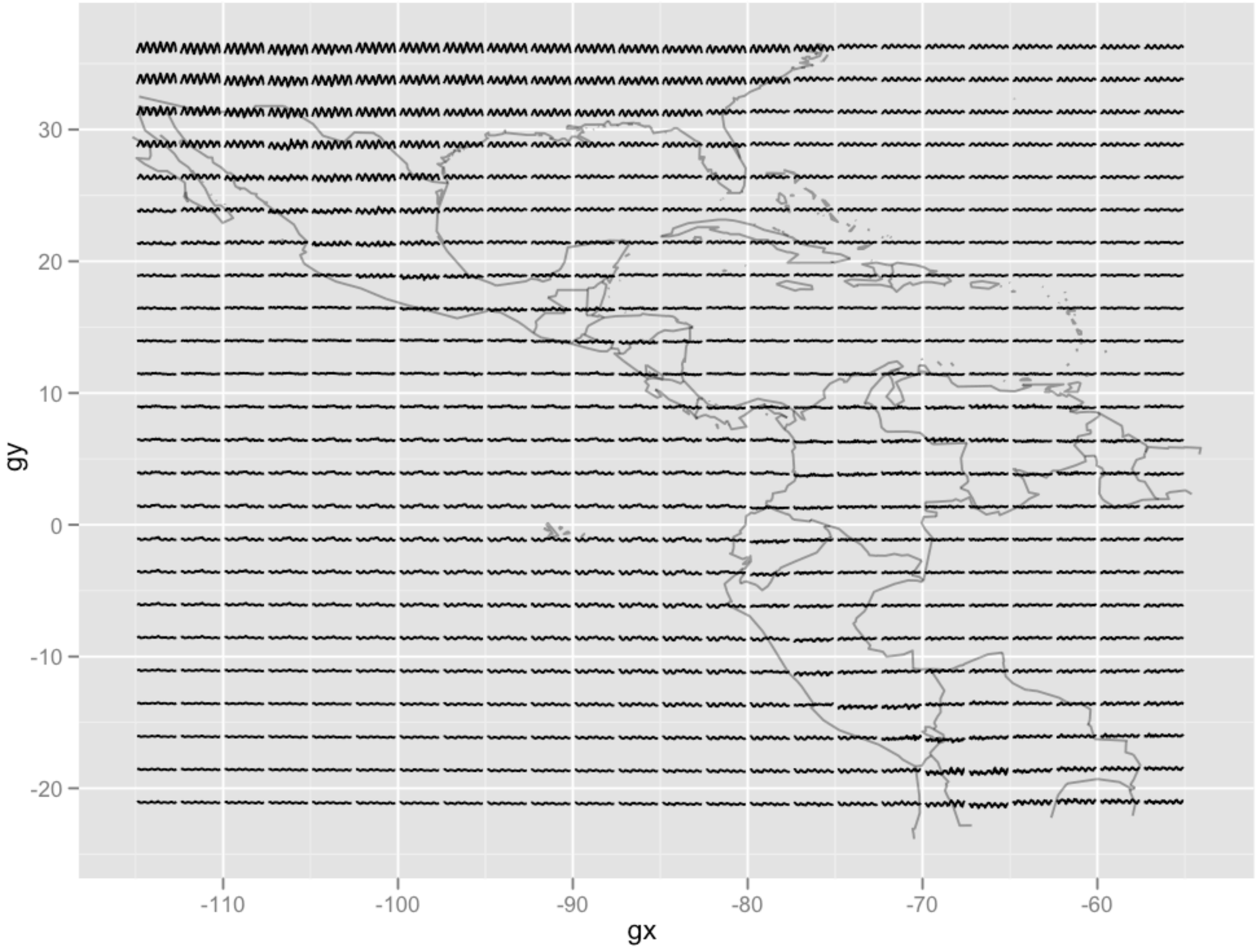
Group-wise modelling

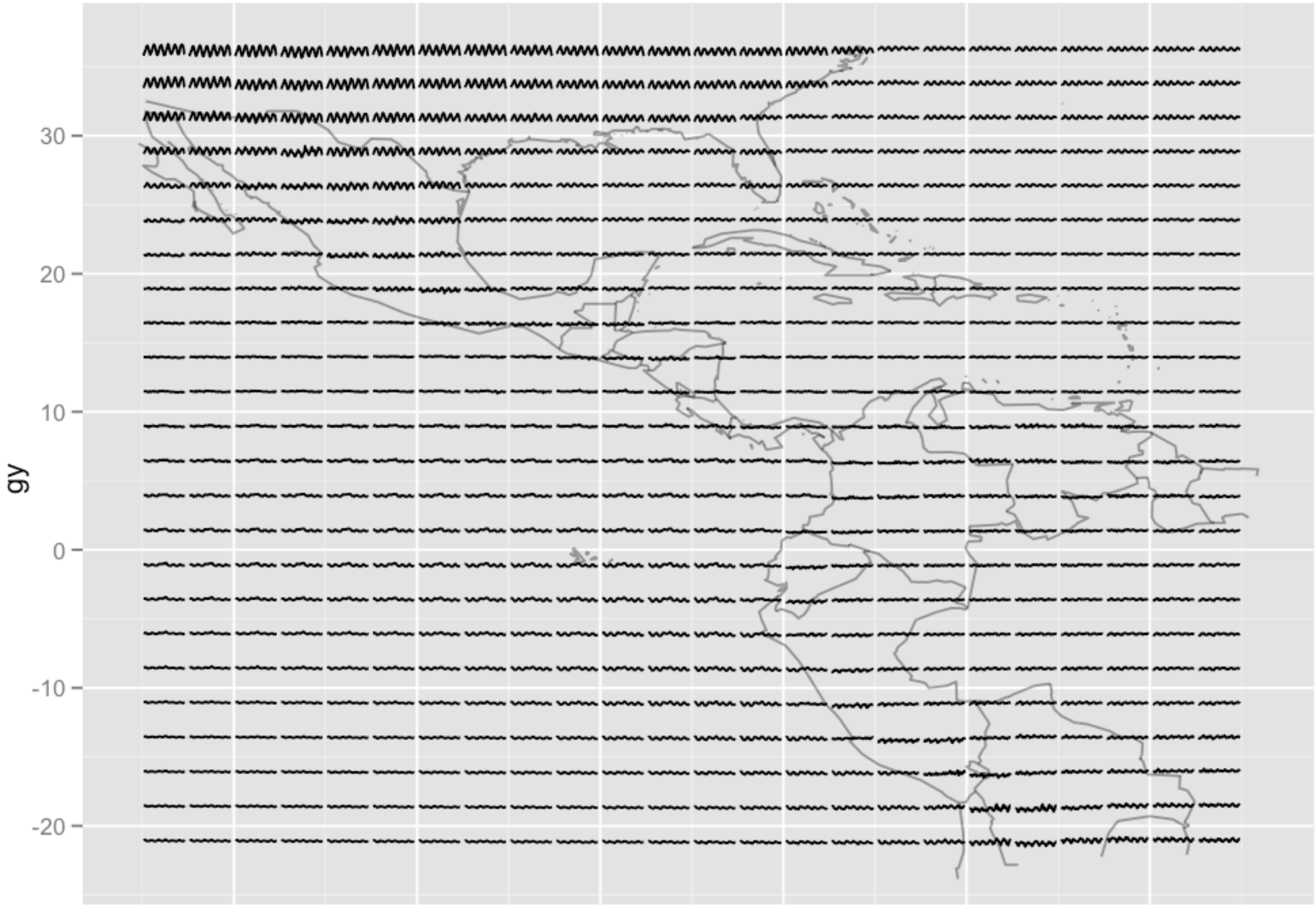
Hadley Wickham

Assistant Professor / Dobelman Family Junior Chair

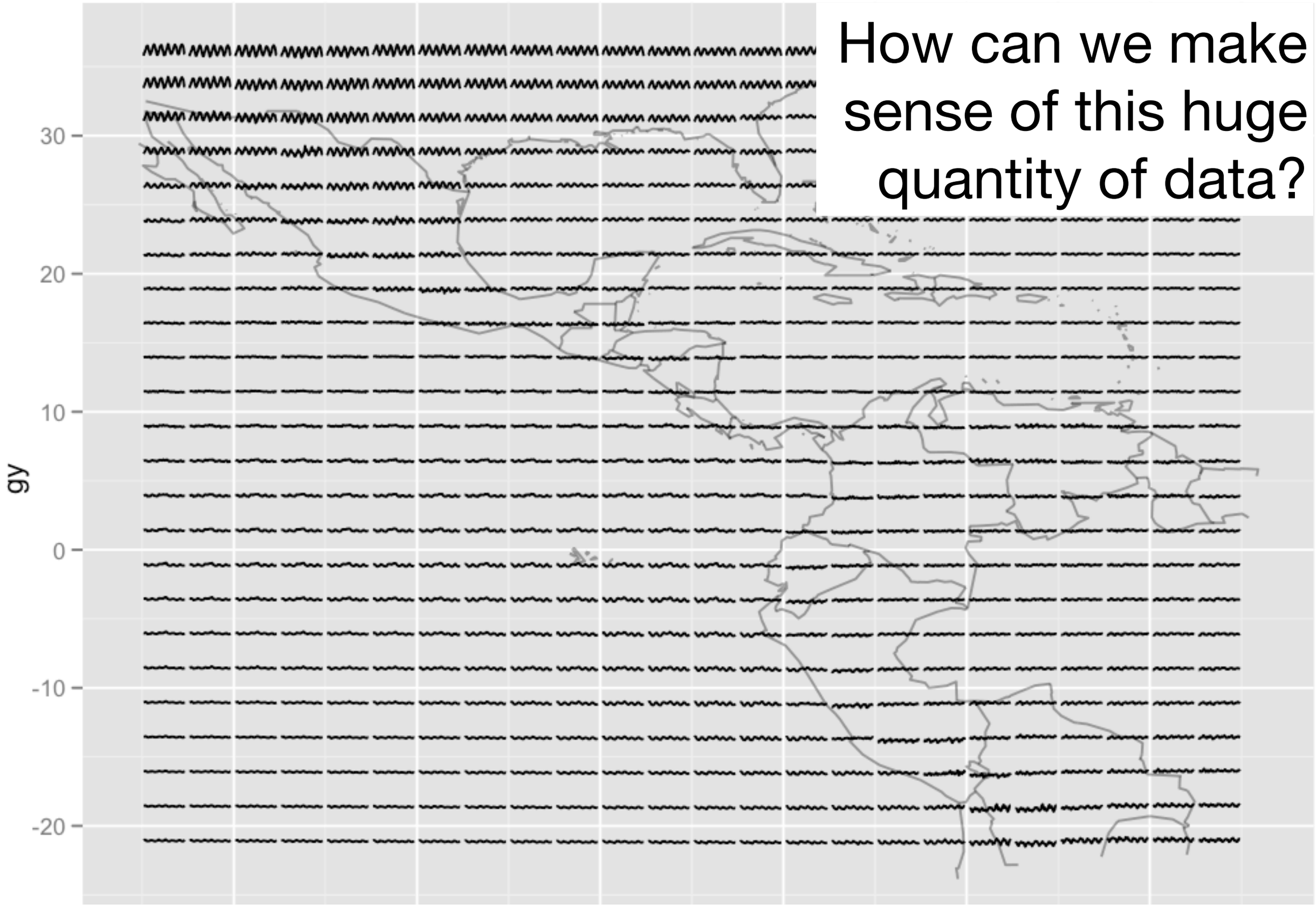
Department of Statistics / Rice University

August 2011


30
20
10
0
-10
-20
gy
-110
-100
-90
-80
-70
-60
gx




```
nasa <- glyphs(nasa, "long", "day", "lat", "surftemp")
qplot(gx, gy, data = nasa, geom = "path", group = id) + map
```




```
nasa <- glyphs(nasa, "long", "day", "lat", "surftemp")
qplot(gx, gy, data = nasa, geom = "path", group = id) + map
```

Data

- 24 x 24 grid, 72 time points = 41,472 observations
- 7 meteorological variables
- We’re just going to focus on surface temperature (`surftemp`)

1. Introduction to models in R
2. Advanced aggregation: fitting a model to each location
3. Extracting coefficients
4. Making predictions
5. Viewing residuals
6. Collating summary statistics

```
# Get started by loading libraries, data
# and the glyph function

library("ggplot2")
library("mgcv")

source("glyphs.r")
source("06-nasa.r")

nasa <- glyphs(nasa, "long", "day",
  "lat", "surftemp")
qplot(gx, gy, data = nasa, geom = "path",
  group = id) + map
```

Modelling basics

```r
# response ~ predictor
lm(surftemp ~ year, data = nasa)

# use factor to force a categorical predictor
# matrix of dummy variables is made automatically
lm(surftemp ~ factor(month), data = nasa)

# default output is minimal. Use functions to
# extract to extract more details
mod <- lm(surftemp ~ factor(month), data = nasa)

summary(mod); anova(mod); coef(mod)
predict(mod); resid(mod)
vcov(mod)
```

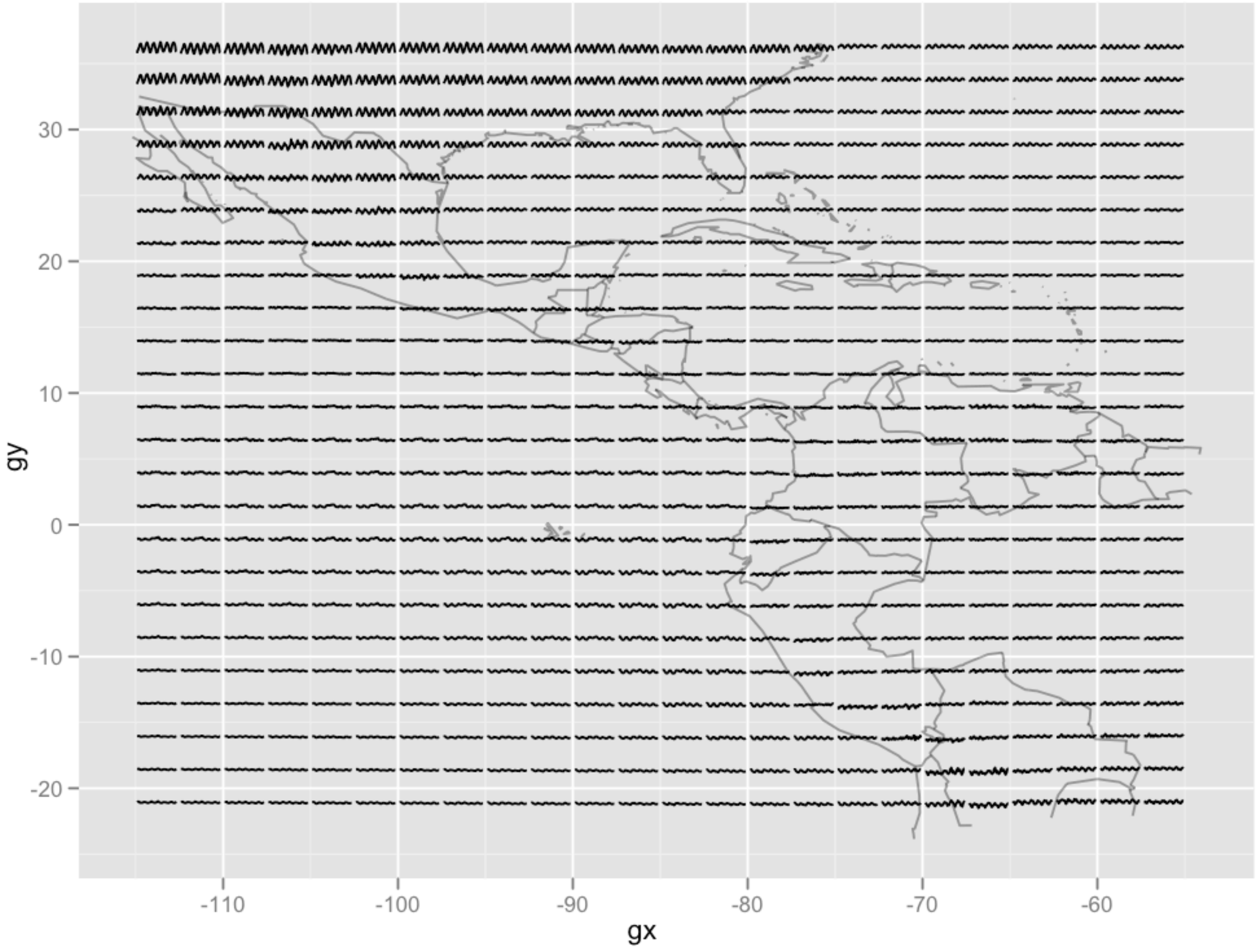
Your turn

Can you work out what +, : and * do?

```
lm(surftemp ~ factor(month) + year,
   data = nasa)
lm(surftemp ~ factor(month):year,
   data = nasa)
lm(surftemp ~ factor(month) * year,
   data = nasa)
```

```
# + adds more predictors
lm(surftemp ~ factor(month) + year, data = nasa)

# : makes an interaction
lm(surftemp ~ factor(month):year, data = nasa)

# * adds main effects and interactions
lm(surftemp ~ factor(month) * year, data = nasa)
```

Per-location models


30
20
10
0
-10
-20
gy
-110
-100
-90
-80
-70
-60
gx

Challenge

Fit a model to each location then **visualise** model summaries.

Similar to the summaries we did with `ddply`, but we'll do it in two steps. 1) Create a model for each location 2) Extract summaries

Requires two new functions: `dlply` and `ldply`. **l** is short for list

```r
# dlply works like ddply, but instead of a data frame
# it returns a LIST

temp_models <- dlply(nasa, c("lat", "long"), function(df) {
  lm(surftemp ~ factor(month), data = df)
})

# You can also try out this model:
# lm(surftemp ~ factor(month) - 1, data = df)
# What's the difference?
```

Your turn

Using what you know about data structures, subsetting and models:

Explore the `temp_models` object. What is it?

View the model summary of the 1st model

Extract the residuals from the 7th model

```
temp_models[1:10]
length(temp_models)
temp_models[[1]]
summary(temp_models[[1]])
anova(temp_models[[1]])
coef(temp_models[[1]])
resid(temp_models[[1]])
```

Next

We have one model for each location.

Next we'll compute and visualise coefficients, predictions, residuals and summary statistics.

Coefficients

```
# ldply is the inverse of dlply - it takes a list
# and combines the output into a data frame

coefs <- ldply(temp_models, coef)
```

Your turn

`coef` doesn't provide the data in quite the right format. We want a column for month, and a column for the seasonal effect. How can you compute that from the model?

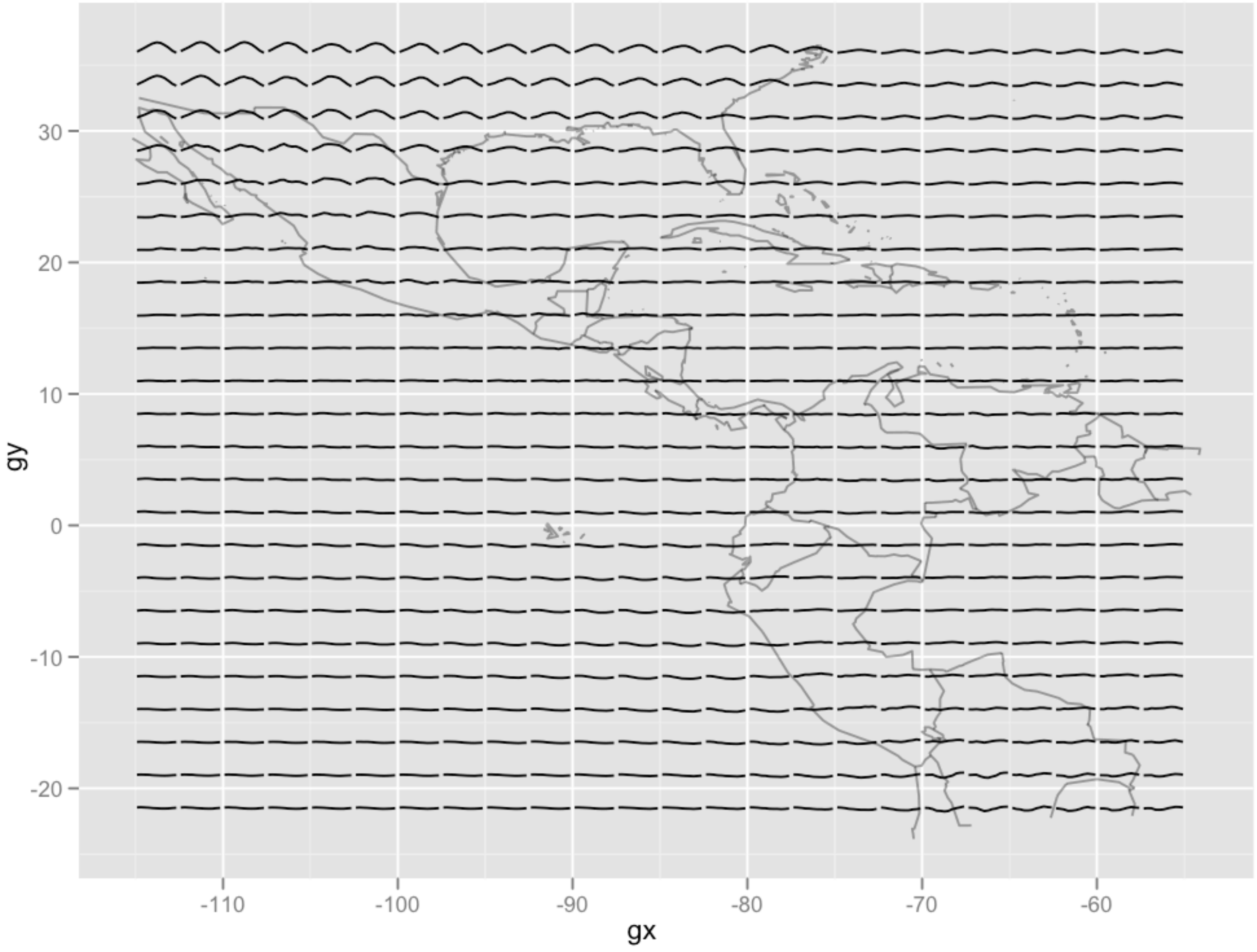
Hint: Experiment with one model

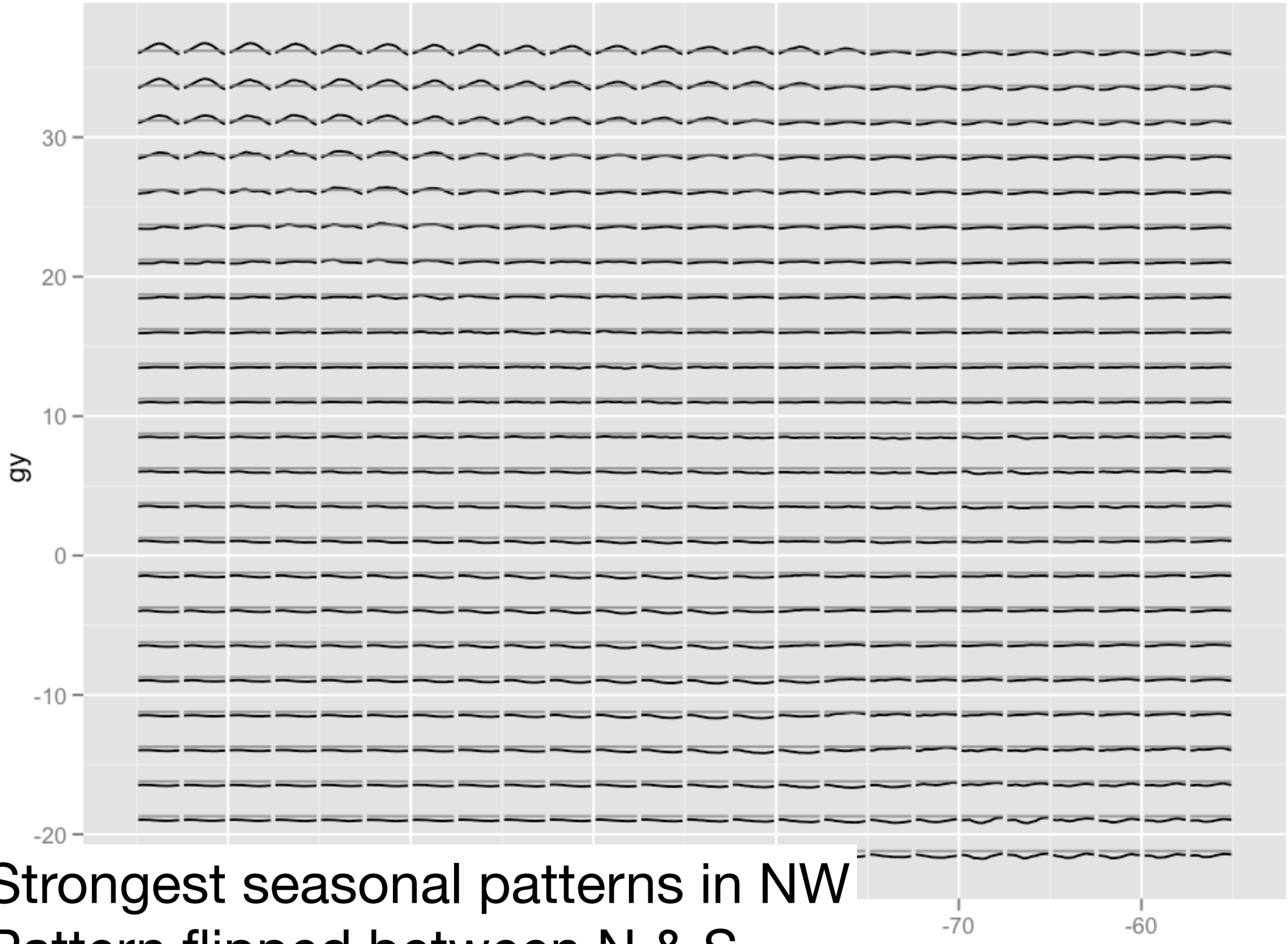
```
one <- temp_models[[1]]
```

```
one <- temp_models[[1]]
coef(one)
cf <- coef(one)

as.data.frame(cf)
data.frame(coef = names(cf), value = cf)
data.frame(month = 2:12, coef = cf[-1])
```

```
coefs <- ldply(temp_models, function(x) {
  data.frame(month = 2:12, coef = coef(x)[-1])
})
```

```
qplot(gx, gy, data = coefs, geom = "line", group = gid) + map
```


30
20
10
0
-10
-20
gy
-110
-100
-90
-80
-70
-60
gx



Strongest seasonal patterns in NW
Pattern flipped between N & S

Predictions

Predictions

Typically coefficients are not that useful - they're fiddly to extract correctly, and not always interpretable

Often better to visualise **predictions** from the model.

Three steps: 1) model, 2) build prediction grid, 3) make predictions

Model

```
# Slightly more sophisticated model:
# seasonal effect + (very) smooth long term trend

temp_models <- dlply(nasa, c("lat", "long"), function(df) {
  lm(surftemp ~ year + factor(month), data = df)
})
```

Make grid

Don't need expand.grid here, but is useful generally

```
# To explore long-term trend, hold month
# constant:
year_grid <- expand.grid(
  year = unique(nasa$year),
  month = 1)

# To explore seasonal trend, hold year constant:
month_grid <- expand.grid(
  year = 2000,
  month = 1:12)
```

Make predictions

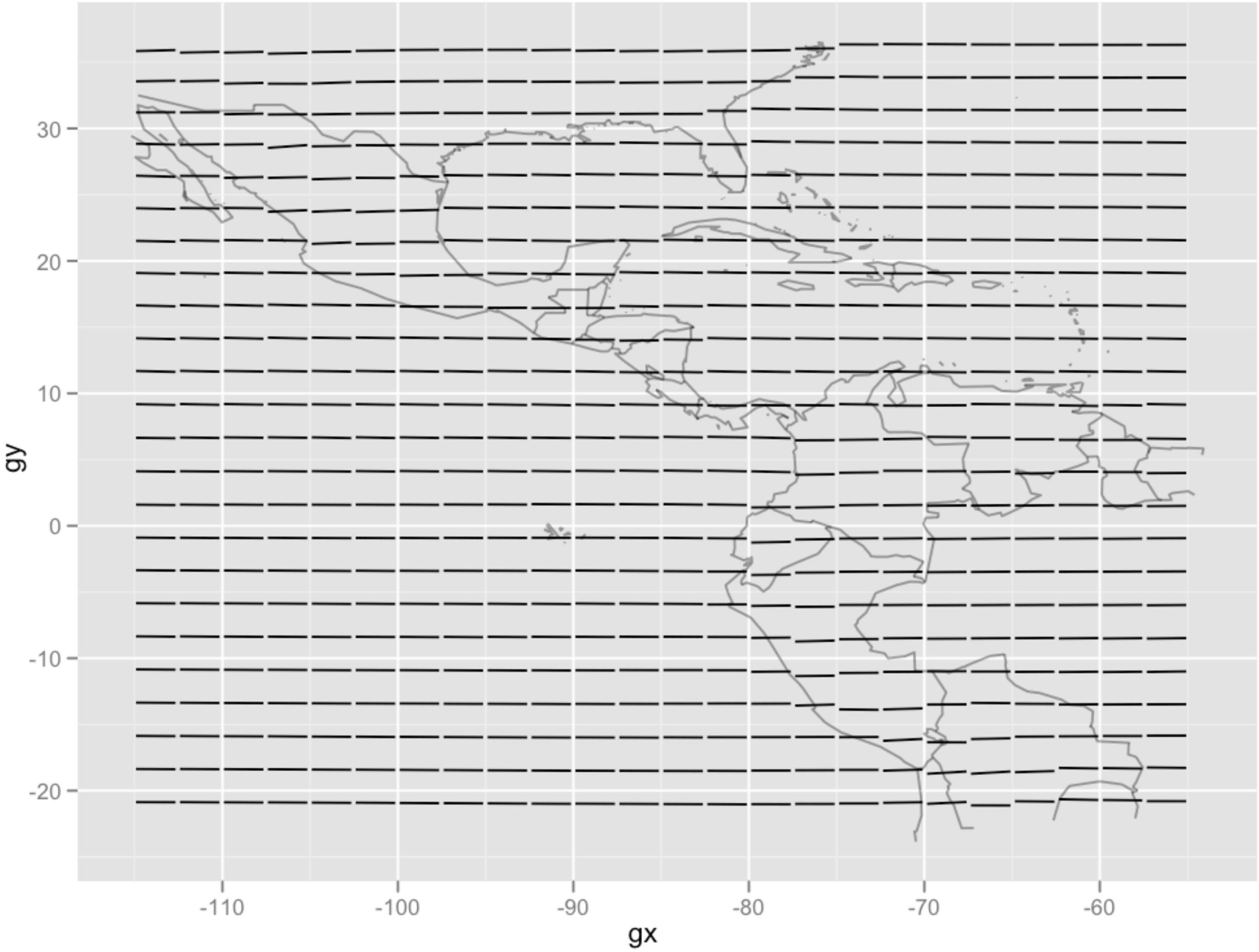
```
month_preds <- ldply(temp_models, function(mod) {
  month_grid$pred <- predict(mod, newdata = month_grid)
  month_grid
})
year_preds <- ldply(temp_models, function(mod) {
  year_grid$pred <- predict(mod, newdata = year_grid)
  year_grid
})

head(month_preds)
head(year_preds)
```

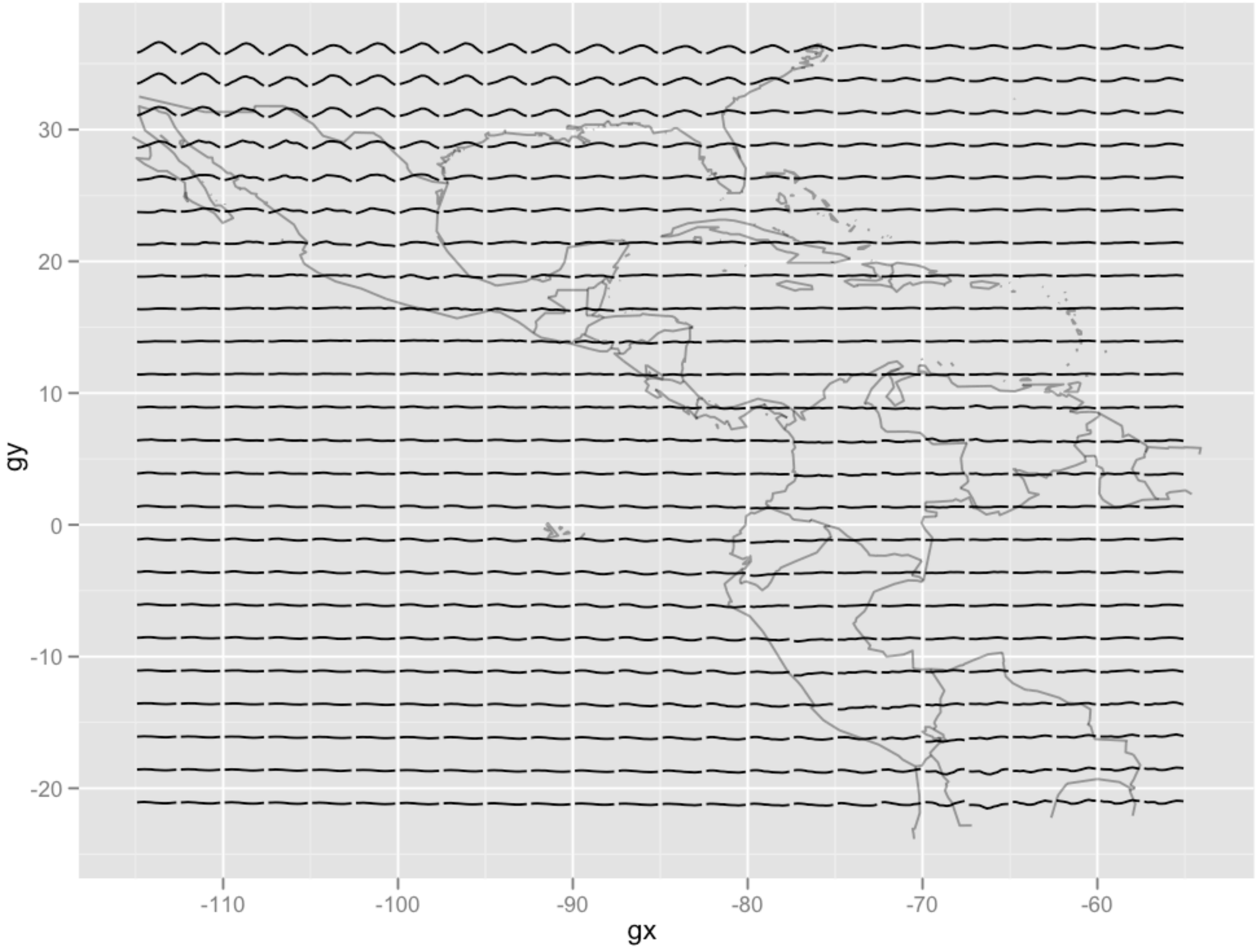
Your turn

Work through the code yourself and then visualise the results.

How does the seasonal pattern compare to the previous plots?

What do you learn about the long-term trend?


30
20
10
0
-10
-20
gy
-110
-100
-90
-80
-70
-60
gx


30
20
10
0
-10
-20
gy
-110
-100
-90
-80
-70
-60
gx

```
month_preds <- glyphs(month_preds, "long", "month",
  "lat", "pred")
year_preds <- glyphs(year_preds, "long", "year",
  "lat", "pred")

qplot(gx, gy, data = month_preds, geom = "path",
  group = gid) + map
qplot(gx, gy, data = year_preds, geom = "path",
  group = gid) + map
```

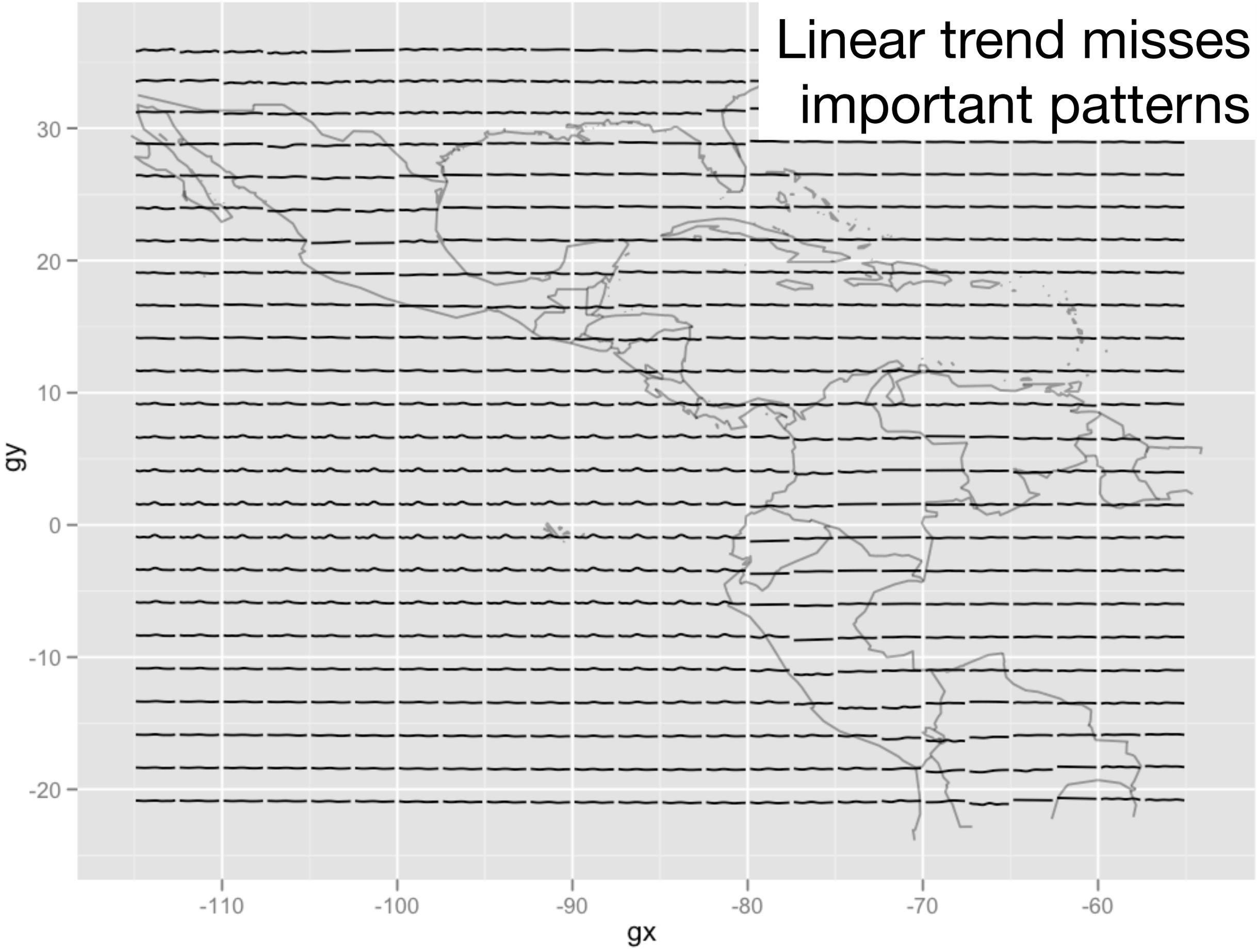
More models

This technique easily generalises to more sophisticated models.

Instead of a linear long-term trend, we can use a generalised additive model to fit a smooth long-term trend.

```
temp_smooth <- dlply(nasa, c("lat", "long"), function(df) {
  gam(surftemp ~ s(day) + factor(month), data = df)
})
day_grid <- expand.grid(day = seq(0, 2161, length = 50), month = 1)
day_preds <- ldply(temp_smooth, function(mod) {
  day_grid$pred <- predict(mod, newdata = day_grid)
  day_grid
})
day_preds <- glyphs(day_preds, "long", "day", "lat", "pred")
qplot(gx, gy, data = day_preds, geom = "path", group = gid) + map
```

Linear trend misses important patterns



Residuals

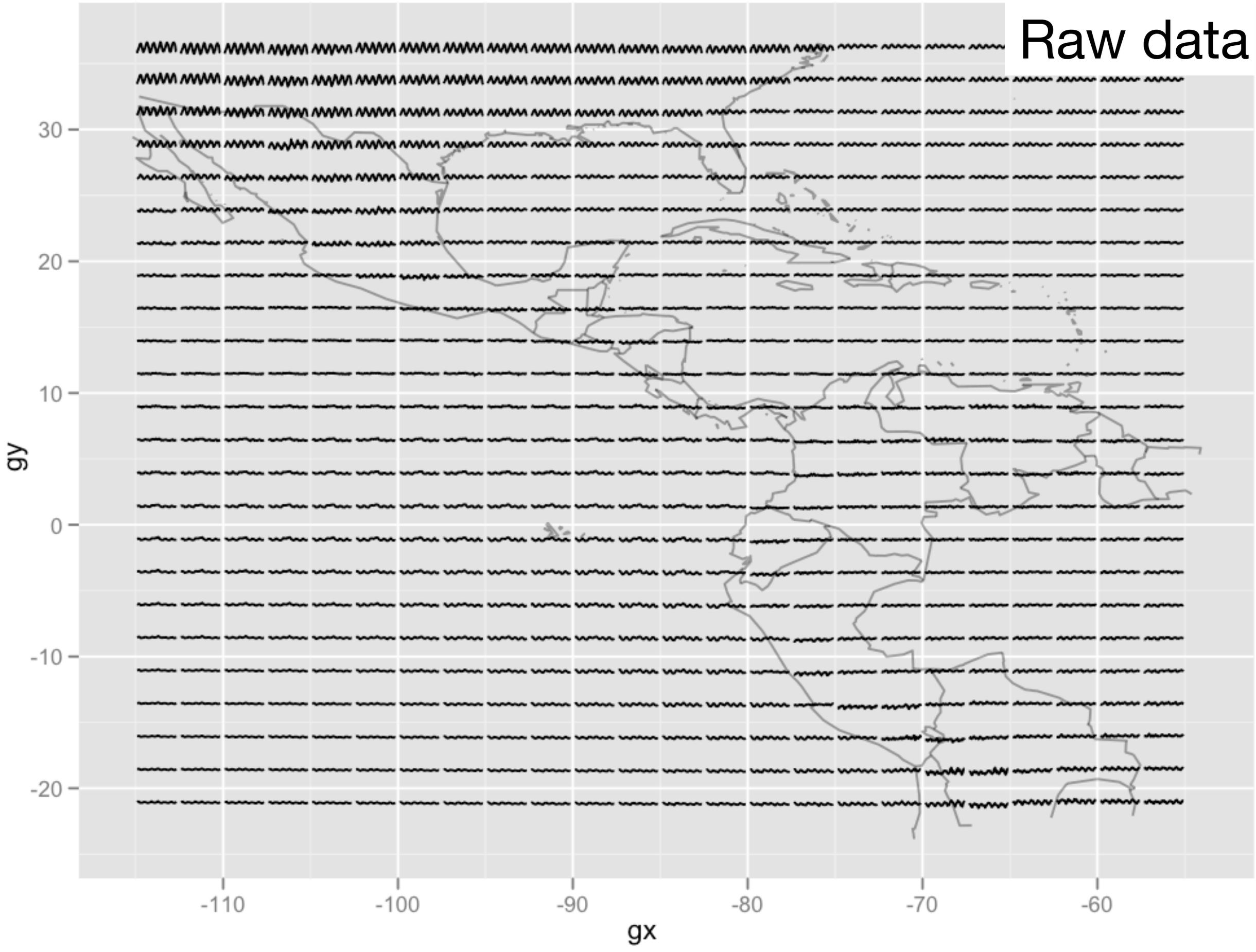
Model checking

Instead of fitting a more complicated model, we could have detected a problem by looking at the residuals for each model in temp_smooth.

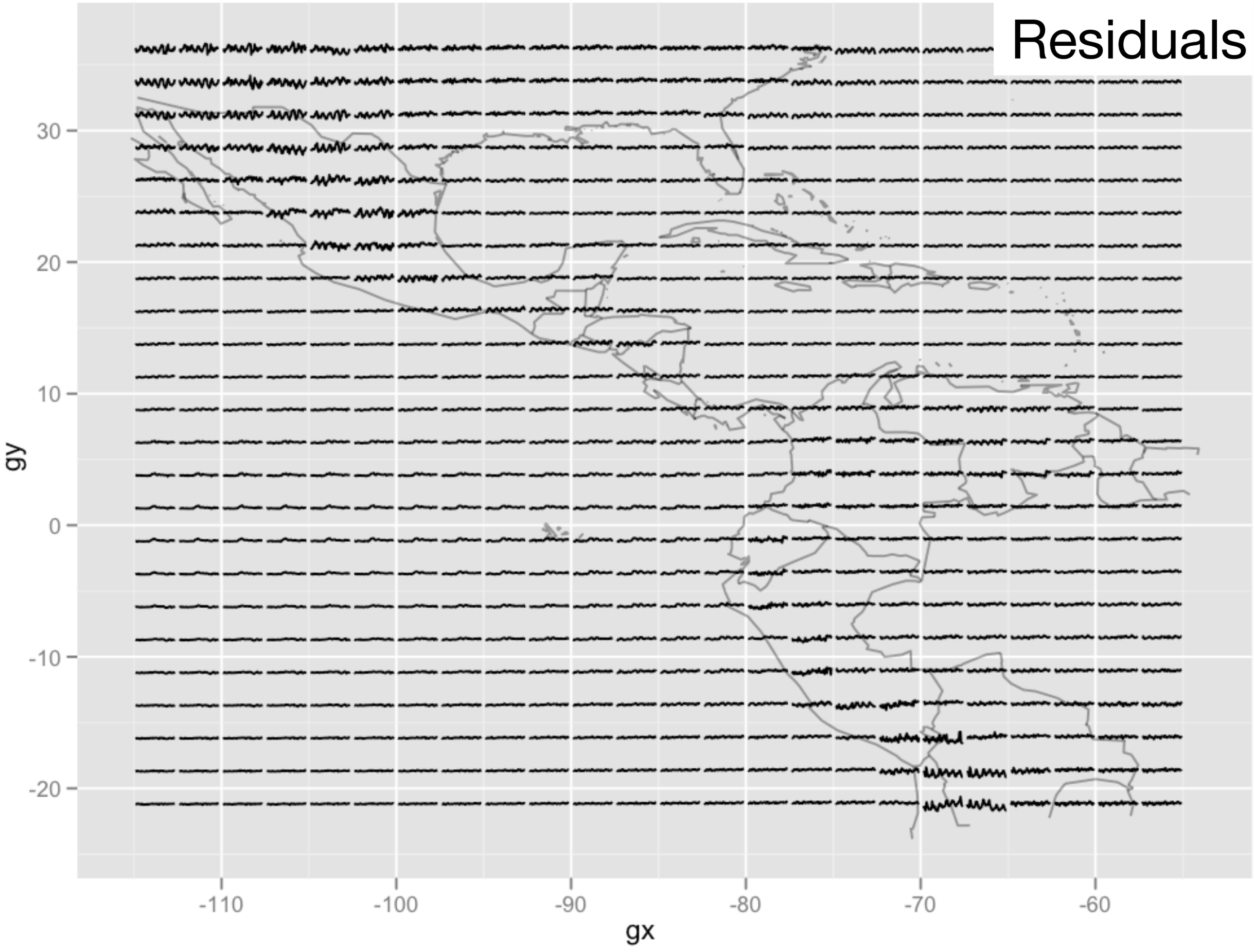
Extracting residuals is more complicated to extract because you need pair up the original data with the corresponding model

```
locs <- dlply(nasa, c("lat", "long"))

# mdply allows us to supply multiple arguments to
# the summary function - pairing data and model
resids <- mdply(cbind(d = locs, m = temp_models), function(d, m) {
  d$temp_resid <- d$temp - predict(m, newdata = d)
  d
})

resids <- glyphs(resids, "long", "day", "lat", "temp_resid")
qplot(gx, gy, data = resids, geom = "line", group = gid) + map
```

Raw data



Residuals



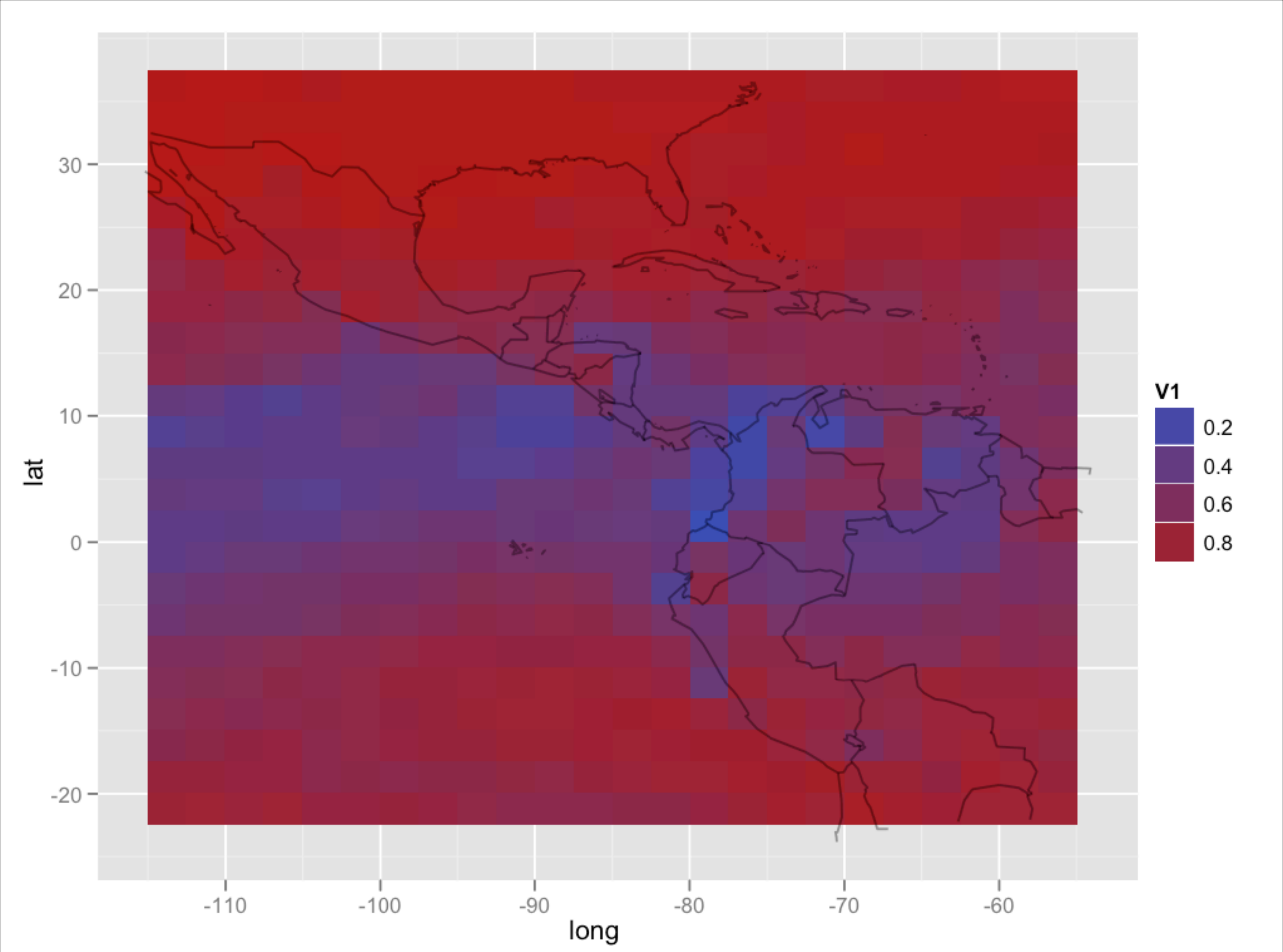
Summary statistics

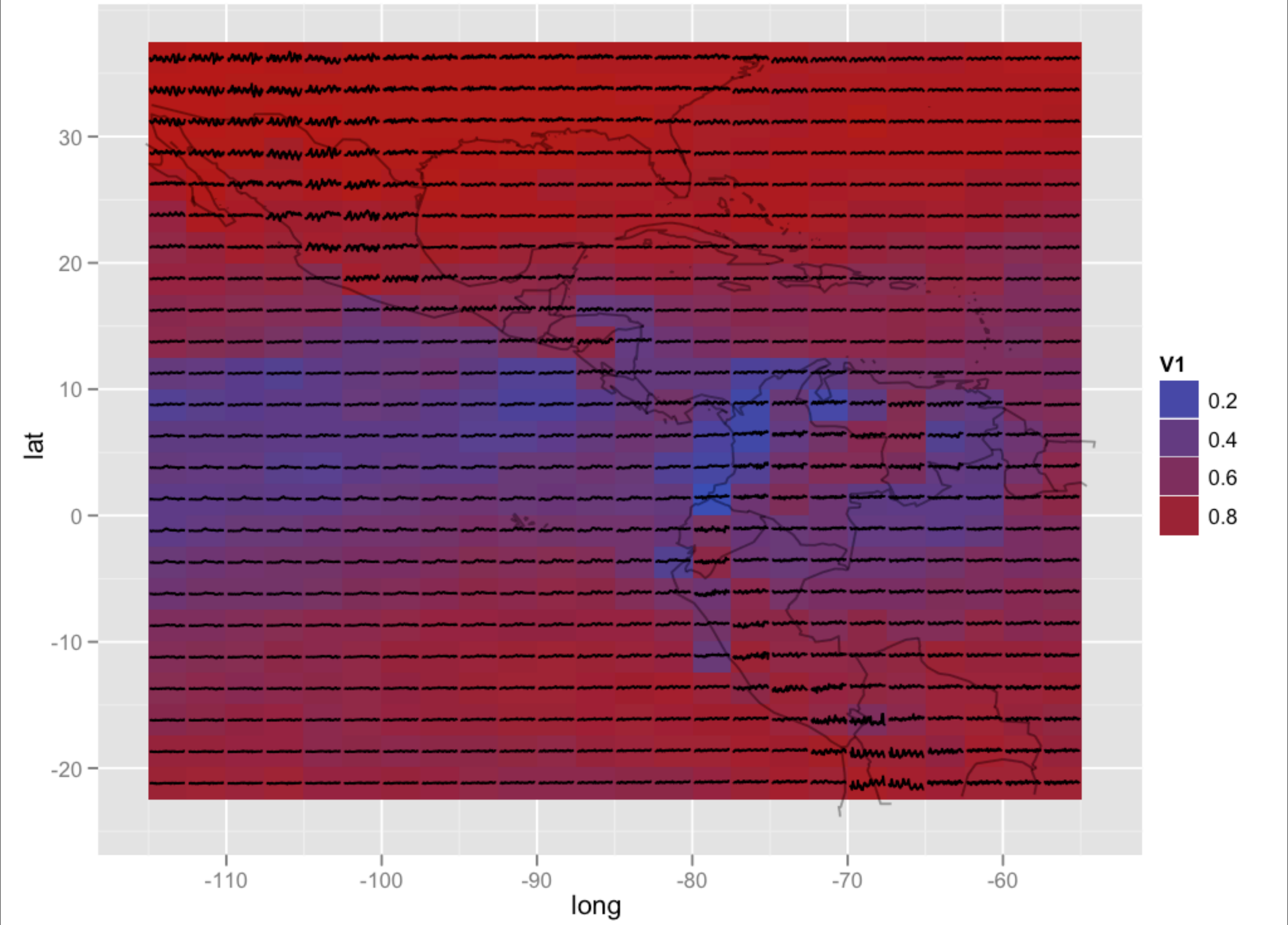
Your turn

How can you extract the r-squared from a single model? How can you extract it from all models and store in a data frame?

(Hint: use `summary` + `str`)

```
rsq <- function(mod) summary(mod)$r.squared
rsqs <- ldply(temp_models, rsq)

qplot(long, lat, data = rsqs, fill = V1, geom =
"tile") + map
```


30
20
10
0
-10
-20
lat
-110
-100
-90
-80
-70
-60
long
V1
0.2
0.4
0.6
0.8


30
20
10
0
-10
-20
lat
-110
-100
-90
-80
-70
-60
long
V1
0.2
0.4
0.6
0.8