SPLIT SETUP

for users 1:11398
assign to folds 1:5 in the cycle
1 2 3 4 5 1 2 3 4 5 .....

the last 10 elements of the vector are
> tail(folds,10)
 [1] 4 5 1 2 3 4 5 1 2 3

fold 4 is devset, with 1:3 dev training
fold 5 is reserved as test set

i have not evaluated anything on test set yet because this seems fairer at this point?

==========
MODEL SETUP

two completely separate models for latitude and longitude.

so we optimize, on devset, two different objectives for two different models:

argmin_betalat { sum_i (predlat_i - reallat_i)^2 + lambda1*sum_j | betalat_j | }
argmin_betalong { sum_i (predlong_i - reallong_i)^2 + lambda2*sum_j | betalong_j | }

pick each regularizer lambda1,lambda2 completely separately to do best on those single-dimension objectives.

Then evaluate the average 2-d euclidean distance. that is, RMSE in the 2d space. So the model that comes out of the above doesn't target this exactly correctly.

loss = (1/D) sum_i sqrt[(predlat_i - reallat_i)^2 + (predlong_i - reallong_i)^2]

==========
TEXT FEATURES

using TF. probabilities within the user. that is,

score(word i, user u) = (#tokens of word_i for user u) / (#tokens total from user)

=====

FITTING GLMNET MODELS

always use dfmax=2000 or 3000

ALPHA TUNING

alpha=.01 and alpha=1 have different #df for the dev-best model
but latitude MSE is nearly identical, ~ 16.9-17.1

longitude MSE is much higher and more sensitive, ~154 mse
alpha=.01 is worse than lasso, at 163 mse

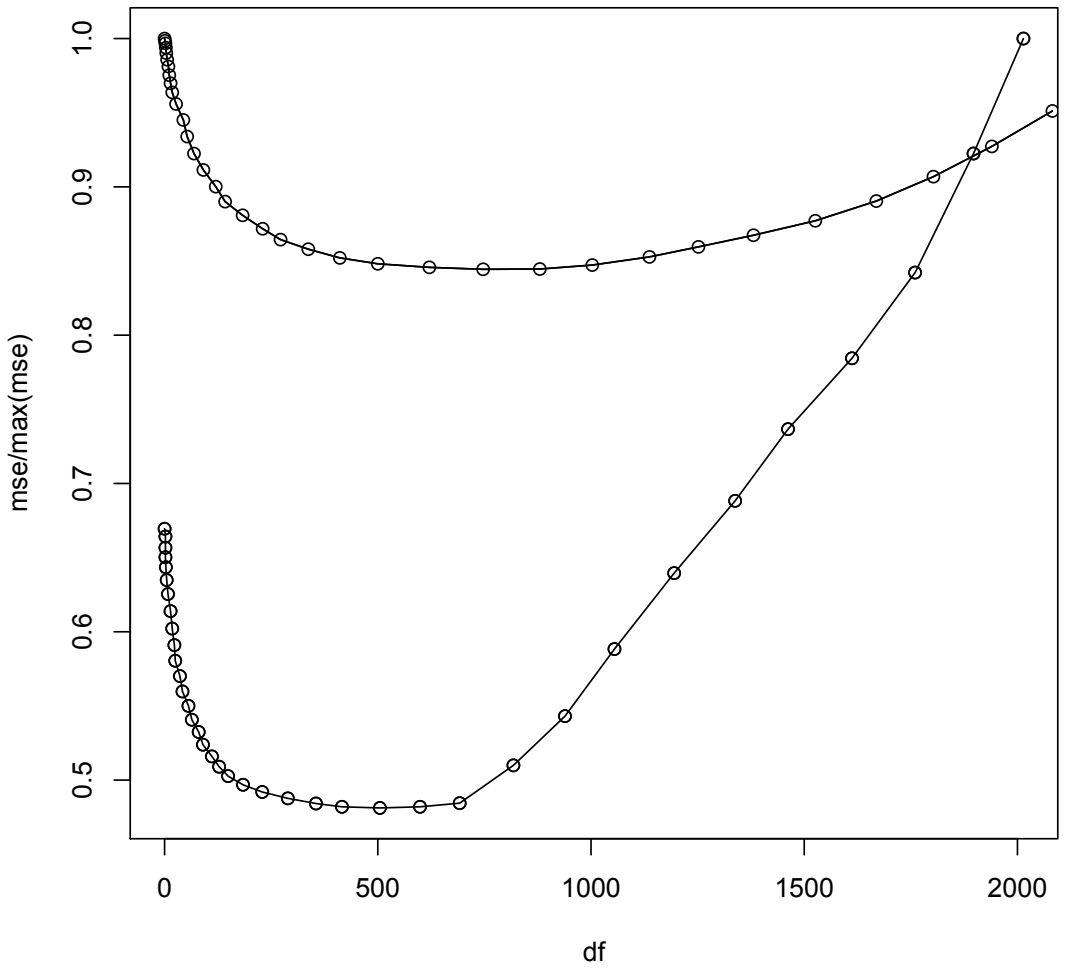
alpha=.50 is ~154 mse

conclusion: just use alpha=1 lasso

LAMBDA TUNING

best lambdas on dev set
longitude 0.274452 (df 505)
latitude 0.07078667 (df 747)

latutude curve is the higher one (less sensitive, right)
note that two completely separate models are learned, the model sizes are comparable only in size



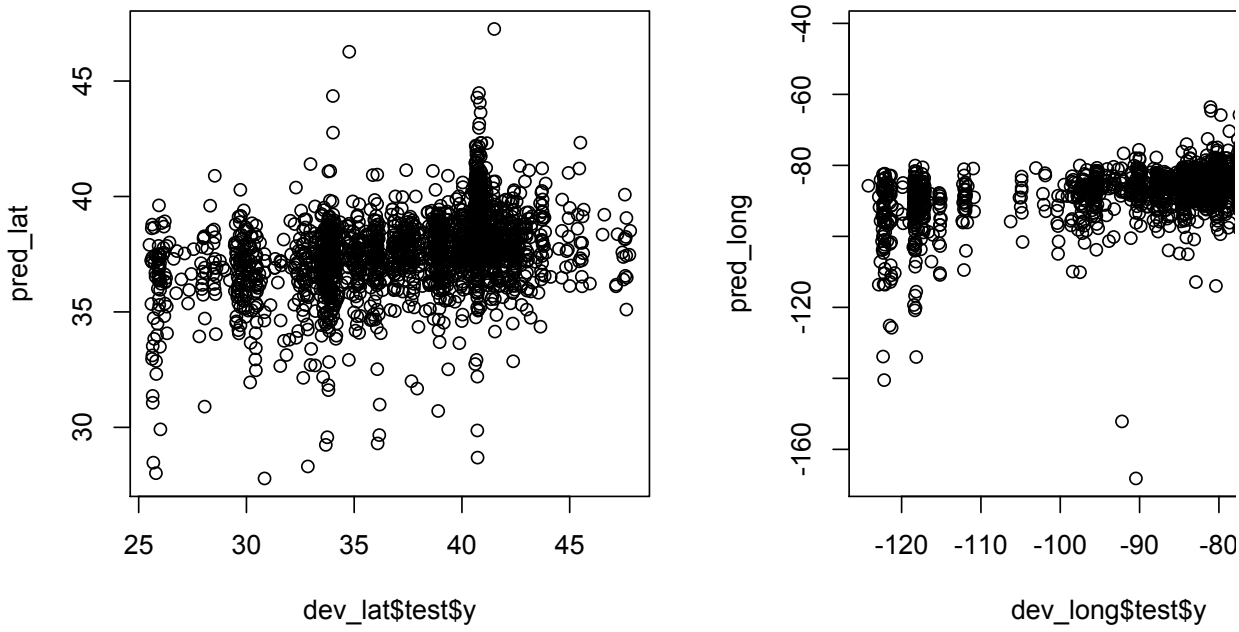
==========

EVALUATION RESULTS

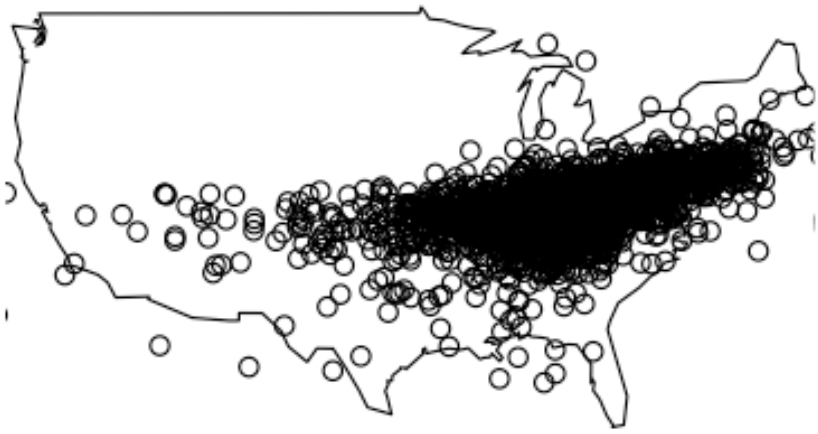
on the devset. using the optimal lambdas (that were directly optimized on the devset).

[[potential confound: should do L2 refits]]

real vs. predicted values are pretty crappy.



and oh god the actual predictions look bad straight-up. if you think about it, if the distribution is bimodal (west and east coasts), then for most people, say there's some indication of the east coast but not enough information to make a confident decision. then the MSE-minimizing solution (best inference for the linear function plus gaussian noise), is to put it like only halfway towards the east coast.



2D Numbers (degree "lengths")
the average 2d euclidean distance, e.g., RMSE-like

12.45439 = from baseline: training center guess
10.18707 = from this model

multiply by 100 to get a rough stupid conversion to km

==========
KEY R HISTORY FRAGMENTS

```
> load_stuff()

> r_lat=run_split(dev_lat,dfmax=2000)
> r_long=run_split(dev_long,dfmax=2000)
# also iterate on 'alpha' param
> eval_run(r_lat)
> eval_run(r_long)

> data.frame(beta=sort(active(r_long$m$beta[,20])))

> z_lat=eval_run(r_lat)
> z_long=eval_run(r_long)
> with(z_long,plot(df,mse/max(mse)))
> with(z_lat,points(df,mse/max(mse)))


> pred=predict_2d(dev_lat,dev_long,r_lat$m,r_long$m,0.07078667,0.274452)

> map('usa')
> with(pred,points(long,lat))

> eval_2d(pred, dev_lat$test$y, dev_long$test$y)
[1] 10.18707
> eval_center_guess(dev_lat,dev_long)
[1] 12.45439
```