This week

- Ensemble methods
  - Bagging
  - Random forests

Today

- Bagging
  - code: ants_bag.R, ants_bag.py
  - inference algorithm (predictive performance)
  - tuning
- Parallel processing

Ensemble methods

- Train many models (ensemble)
- Average the models to predict
- Averaging reduces prediction variance

e.g. $\mathrm{Var}(\bar{y}) = \frac{\sigma_y^2}{n}$ (for independent y)

Takeaway:
variance of the mean of y is
less than the variance of y

Bagging

- Bootstrap
  - form many new datasets by resampling from the data
  - sample with replacement
  - train model on each dataset
- Aggregate
  - average over trained models

Bagging algorithm

```
for many repetitions
    resample the data with replacement
    train the base model
    record prediction
final prediction = mean of predictions
```

Base model: can be any type of model

Bagged regression tree

Algorithm pseudocode to R

setup

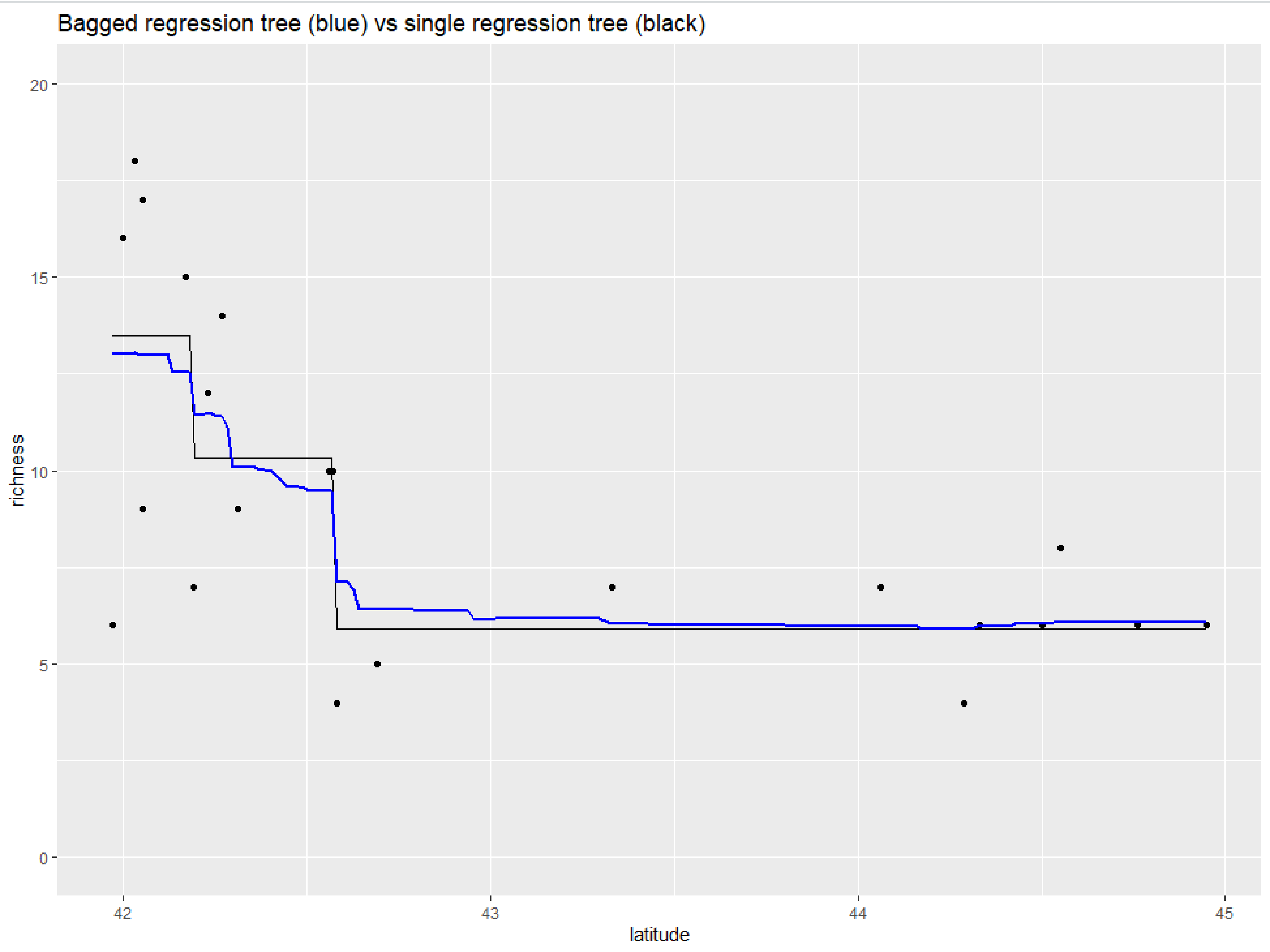
```
# Bagging algorithm
boot_reps <- 500
n <- nrow(forest_ants)
nx <- nrow(grid_data)
boot_preds <- matrix(rep(NA, nx*boot_reps), nrow=nx, ncol=boot_reps)
# for many repetitions
for ( i in 1:boot_reps ) {
#   resample the data (rows) with replacement
    boot_indices <- sample(1:n, n, replace=TRUE)
    boot_data <- forest_ants[boot_indices,]
#   train the base model
    boot_train <- tree(richness ~ latitude, data=boot_data)
#   record prediction
    boot_preds[,i] <- predict(boot_train, newdata=grid_data)
}
bagged_preds <- rowMeans(boot_preds)
```

Bagged regression tree

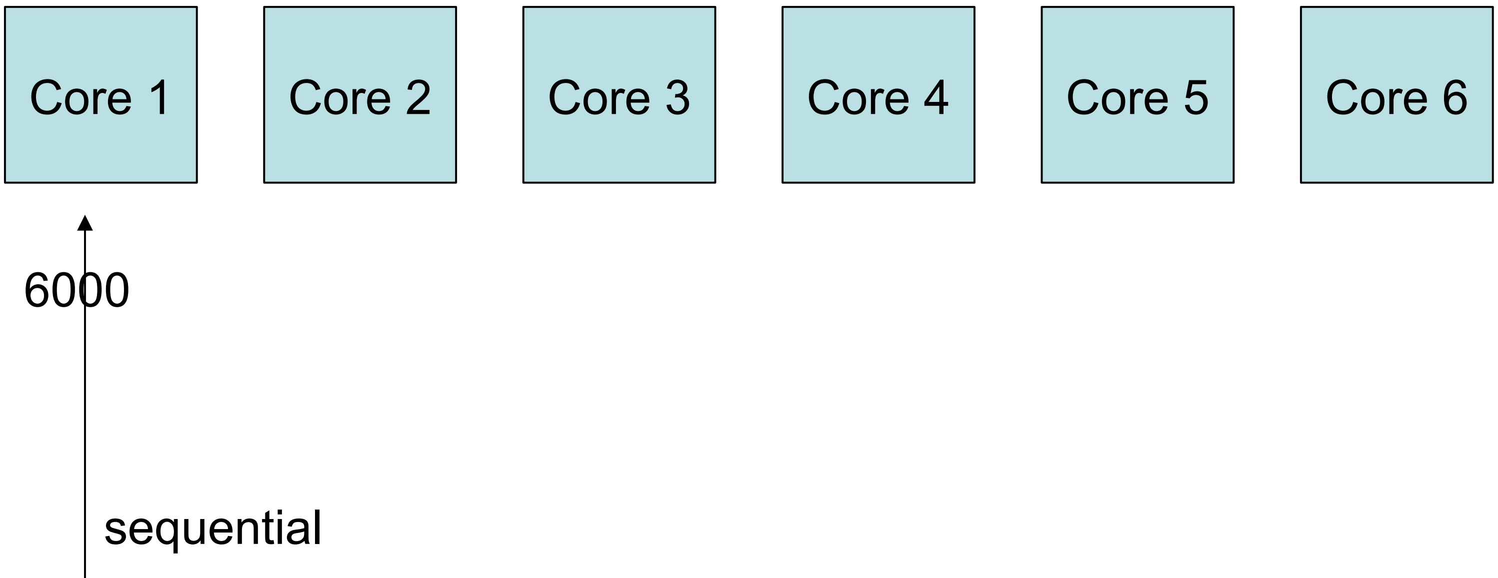
Algorithm pseudocode to Python

setup

```
# Bagging algorithm
boot_reps = 500
dt = tree.DecisionTreeRegressor(max_depth=2) #define base model
n = len(forest_ants)
nx = len(grid_data)
boot_preds = np.full((nx, boot_reps), np.nan)
# for many repetitions
for i in range(boot_reps):
    # resample the data (rows) with replacement
    boot_indices = rng.choice(range(n), n, replace=True)
    boot_data = forest_ants.iloc[boot_indices]
    # train the base model
    boot_train = dt.fit(boot_data[["latitude"]], boot_data["richness"])
    # record prediction
    boot_preds[:,i] = boot_train.predict(grid_data)
# mean of predictions
bagged_preds = np.mean(boot_preds, axis=1)
```

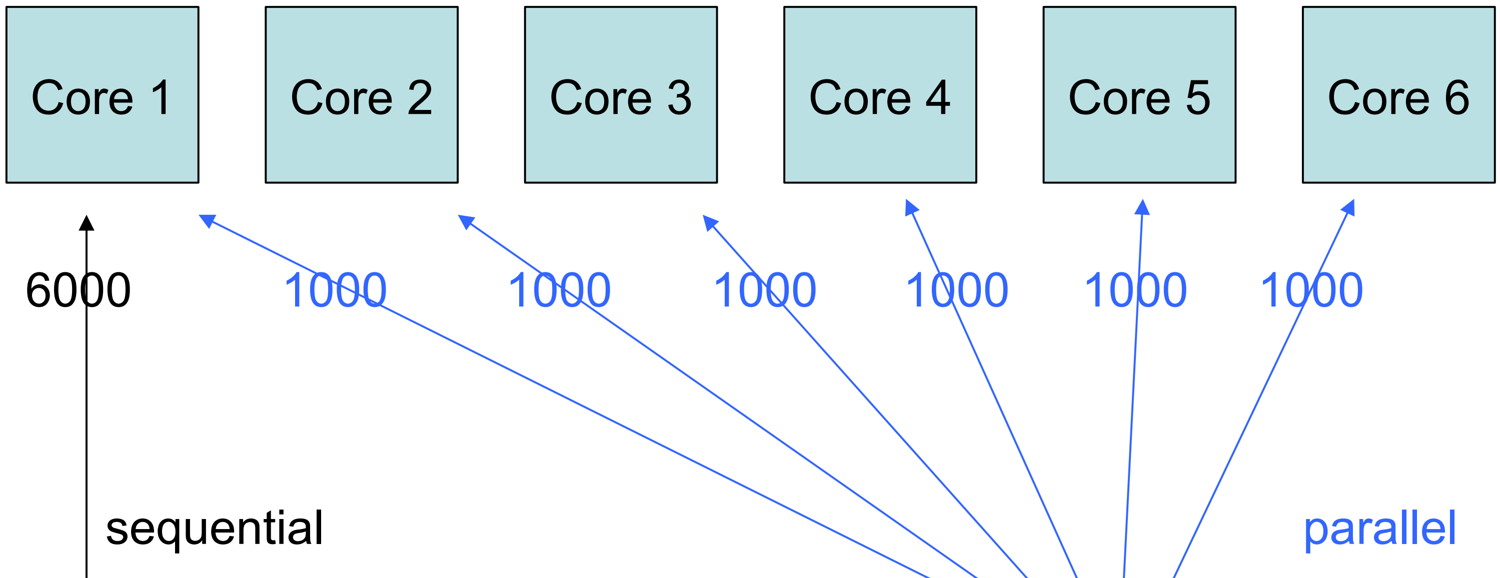

Bagged regression tree (blue) vs single regression tree (black)
richness
0
5
10
15
20
latitude
42
43
44
45

Parallel processing




```
for ( i in 1:6000 ) {
   y[i] <- long_job()
}
```

Parallel processing




```
for ( i in 1:6000 ) {
   y[i] <- long_job()
}
```

```
y <- foreach ( i=1:6000 ) %doRNG% {
   long_job()
}
```

Setup

```
library(doFuture)
library(doRNG)
registerDoFuture()
```

```
library(future)
library(foreach)
```

```
availableCores()
plan(multisession, workers=8)


# Handy timing function:
system.time( some_function() )

# Saving/loading long jobs
save(myresult1, myresult2, file="/saved/myresult.Rdata")
load("/saved/myresult.Rdata")
```

Inference algorithm

- 5-fold CV, 500 splits
- Ants: mean prediction error (MSE)

    12.93 +/- 0.07

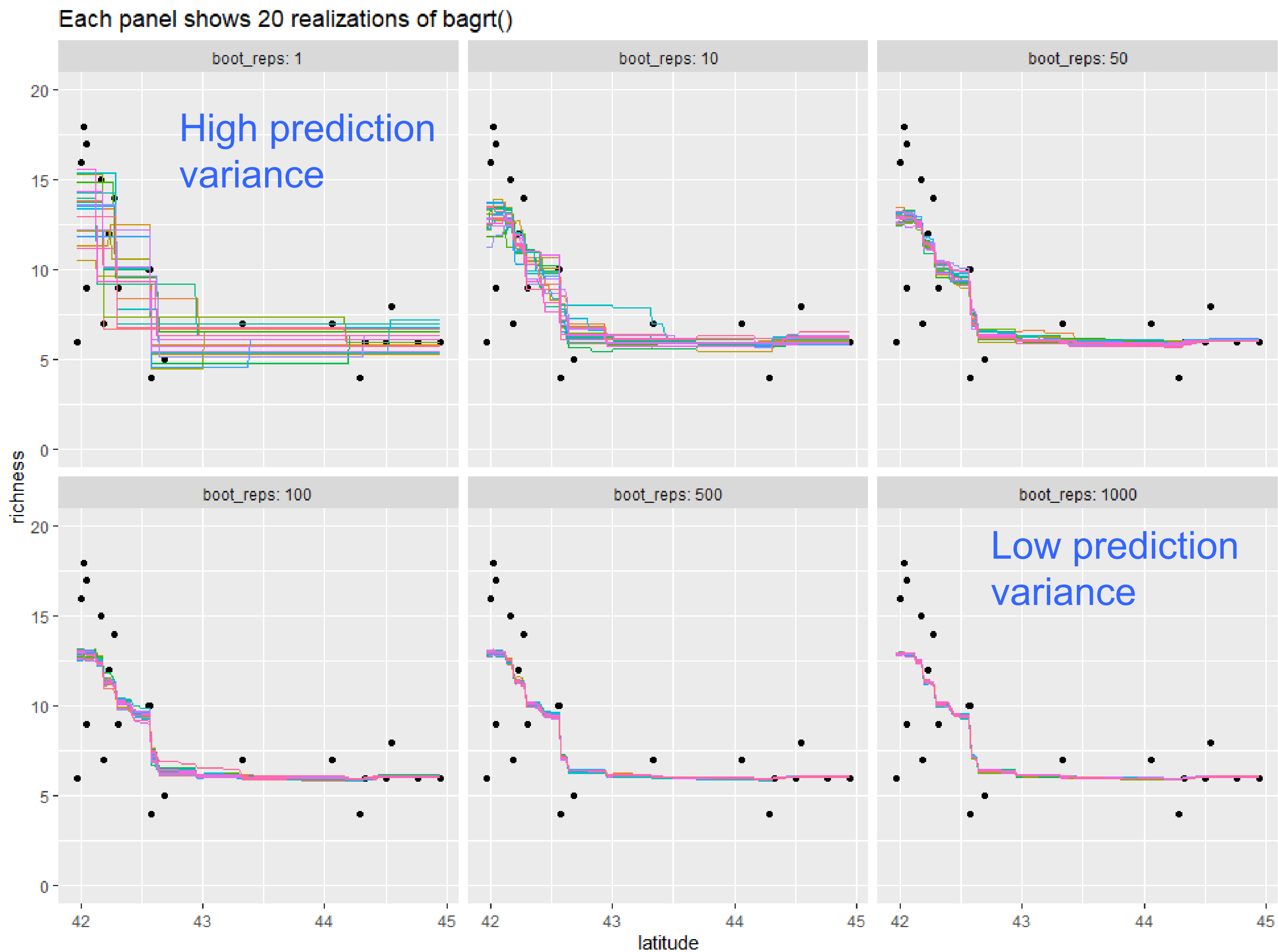
Model	LOOCV	5-fold CV
Polynomial 2	12.88	13.51
Single reg tree	12.68	13.15
KNN 7	12.63	13.03
KNN 6	12.95	13.01
Bagged reg tree	13.23*	12.93
Smoothing spline 3	12.52	12.77

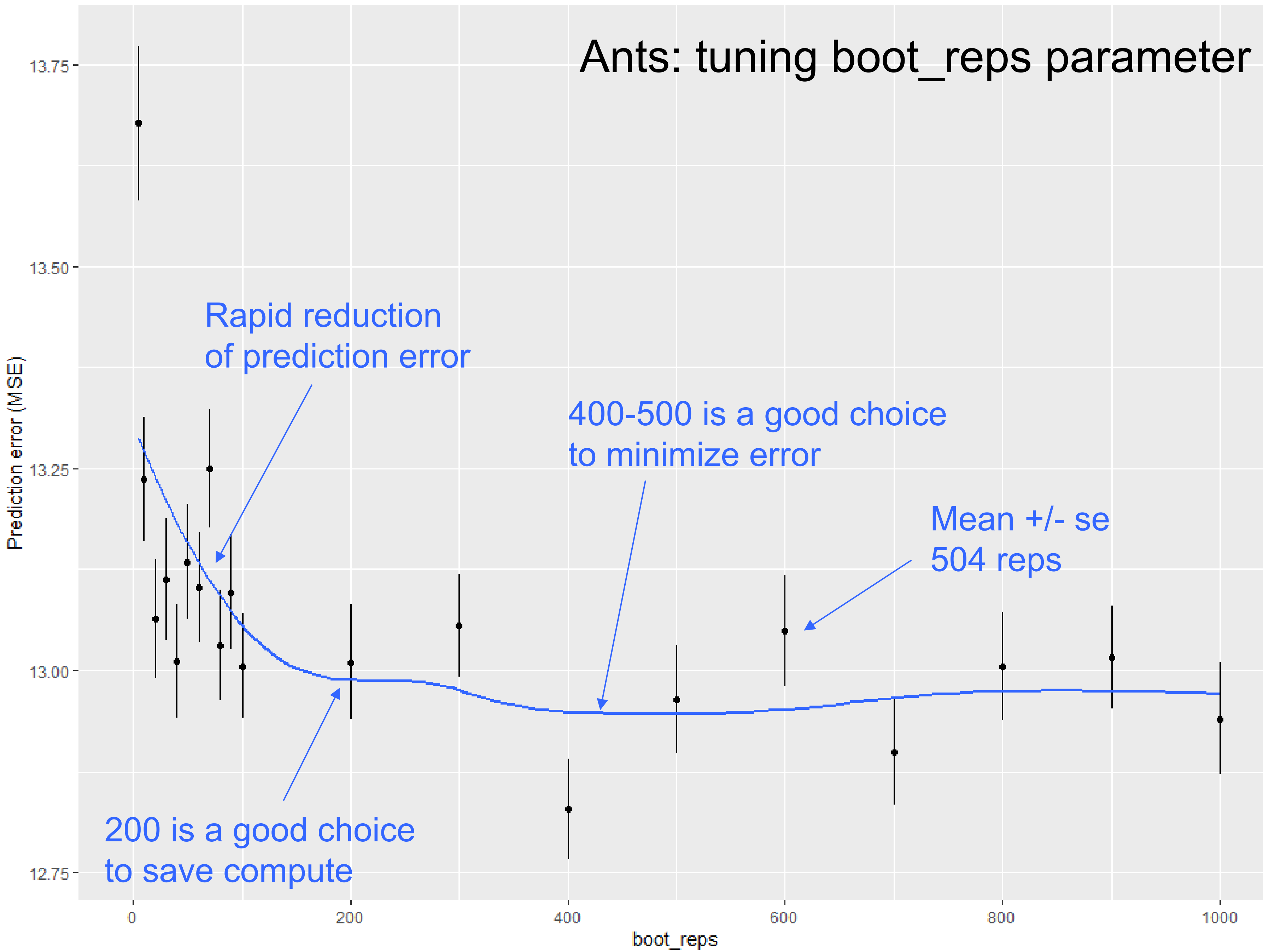
2nd best on 5-fold CV:
generally good predictive function

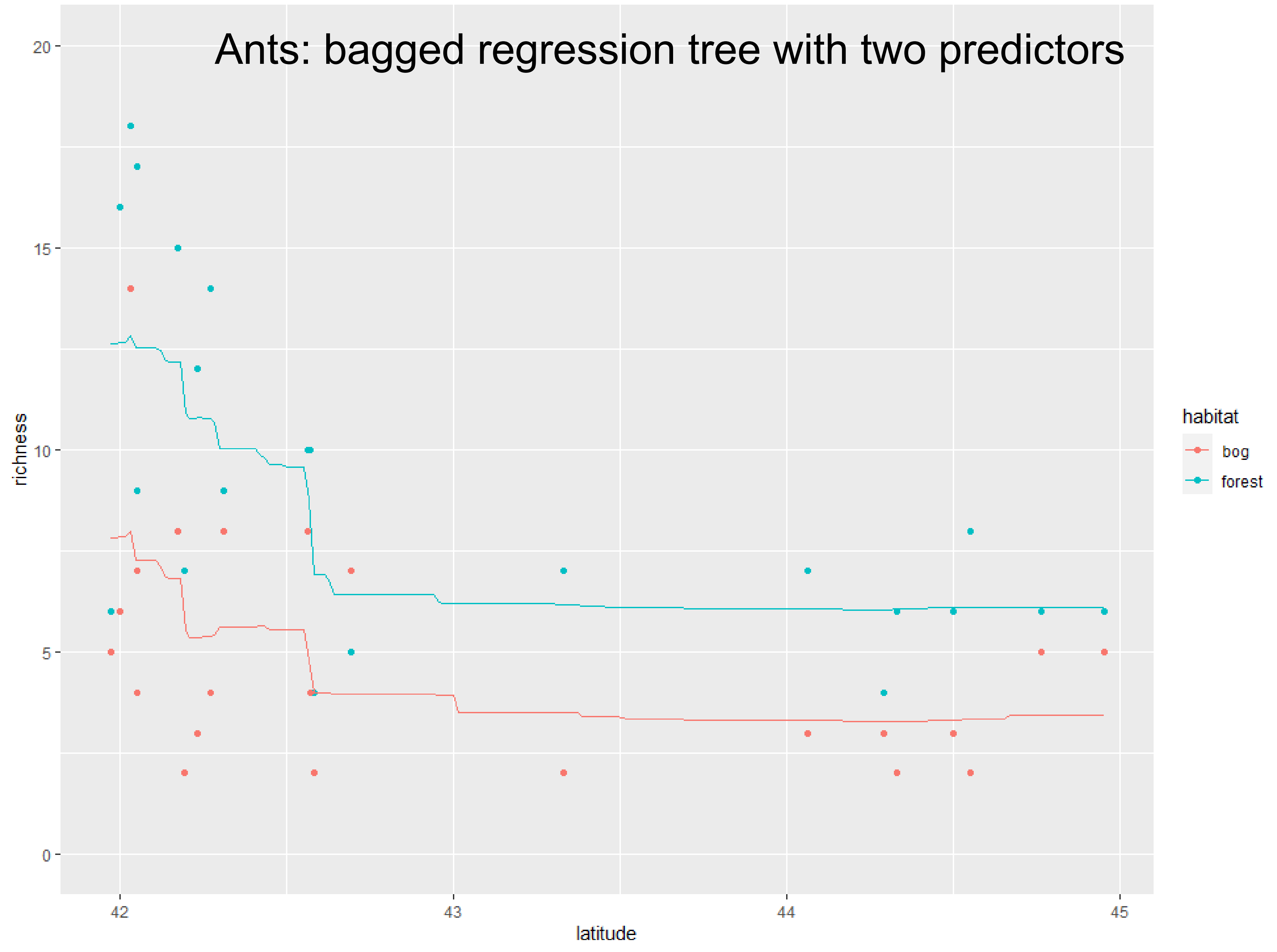
Worst on LOOCV:
perhaps an influential data point

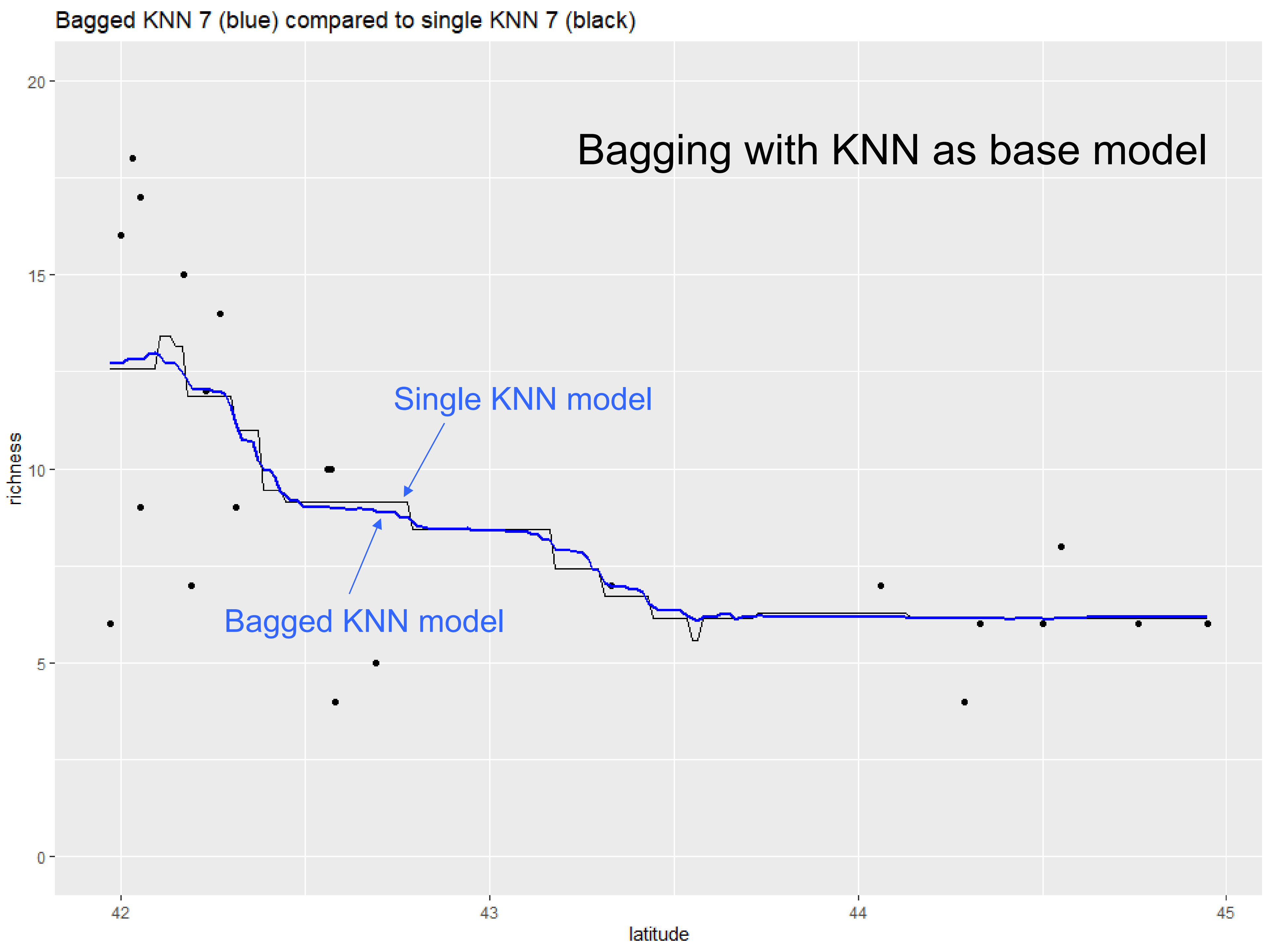
Prediction is always tenuous
with a small dataset

Bagging reduces prediction variance




Ants: tuning boot_reps parameter
Rapid reduction
of prediction error
400-500 is a good choice
to minimize error
Mean +/- se
504 reps
200 is a good choice
to save compute
Prediction error (MSE)
13.75
13.50
13.25
13.00
12.75
0
200
400
600
800
1000
boot_reps


Ants: bagged regression tree with two predictors
richness
latitude
habitat
bog
forest
0
5
10
15
20
42
43
44
45


Bagged KNN 7 (blue) compared to single KNN 7 (black)
Bagging with KNN as base model
Single KNN model
Bagged KNN model
richness
latitude
20
15
10
5
0
42
43
44
45