Machine learning for ecology with R

Brett Melbourne
Associate Professor, EBIO
brett.melbourne@colorado.edu
Pronouns: he, him, his

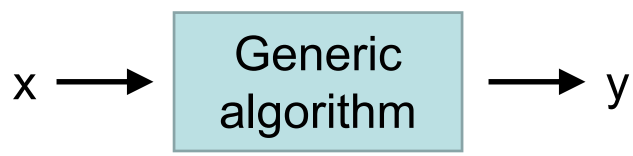
GitHub

- Code for this presentation
  - github.com/melbourne-lab/ml4e-nutshell

- Machine learning for ecology graduate course
  - github.com/EBIO5460Spring2022

What is machine learning?

- Using generic algorithms to predict outputs y from inputs x
- Emphasis: predictive skill

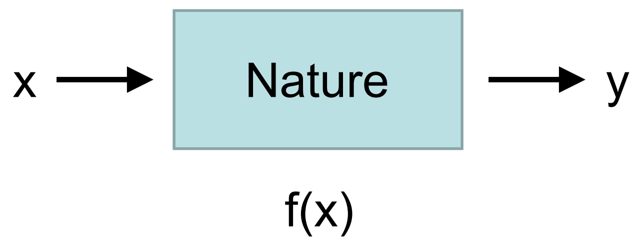


Examples in ecology

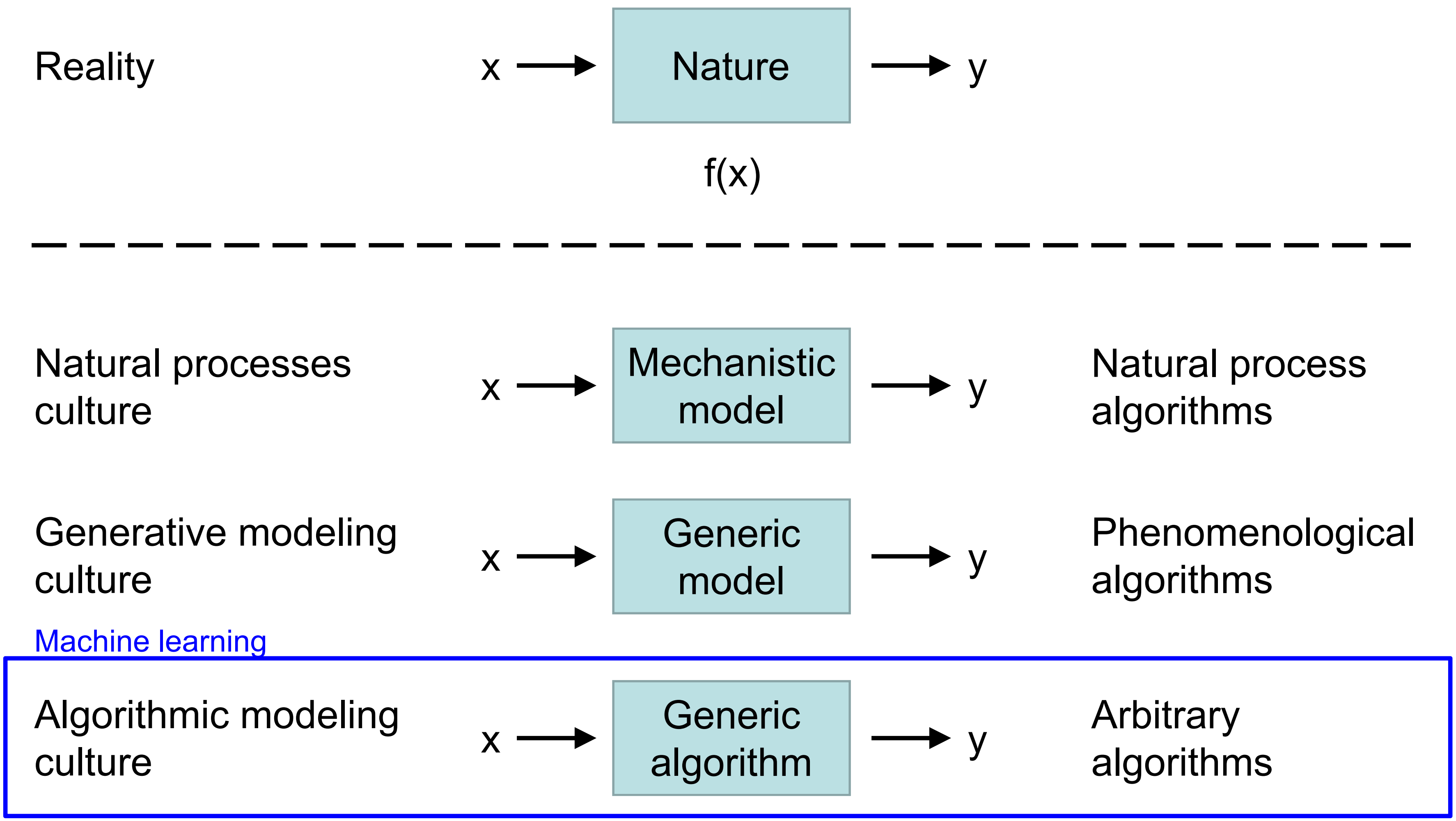
- Species distribution models (SDMs)
  - predicting the spatial distribution of a species from explanatory variables
- Counting the number of penguins in Antarctica from satellite imagery
- Identifying species in camera trap images
- Identifying bird species from audio recordings

Trying to learn a function f

Reality



Trying to learn a function f



f can mean different things in different data science cultures

Goal of prediction

Use data to find a function $\hat{f}$ that has good predictive performance given X

That is, $\hat{f}$ is accurate on new observations

Goal of machine learning

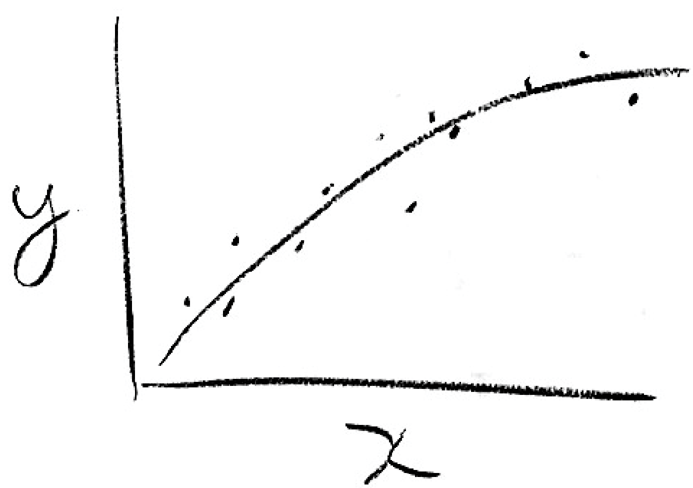
To predict accurately!

- Species distribution
  - map
  - predict accurately for places we won’t visit
- Climate change forecast
  - predict accurately for the future
- Antelopes in camera trap images
  - hand over the identification task to a machine so we don’t have to look at images!
  - predict accurately for images that we’ll never look at

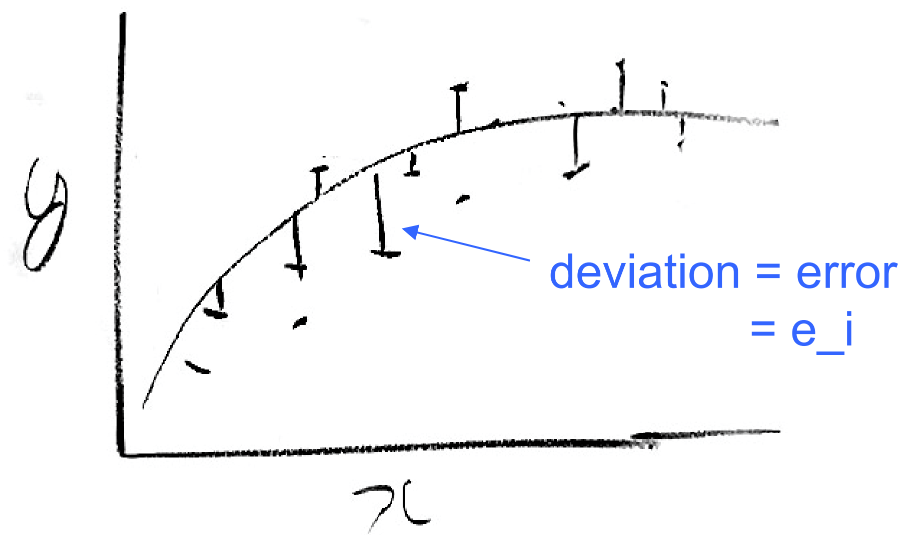
Predictive skill

Basic idea: out-of-sample accuracy

$\hat{f}$ fitted on training data



$\hat{f}$ predicting new data



e.g. mean square error (MSE) $\frac{1}{n}\sum_{i}^{n} e_i^2$

Basic full ML setup

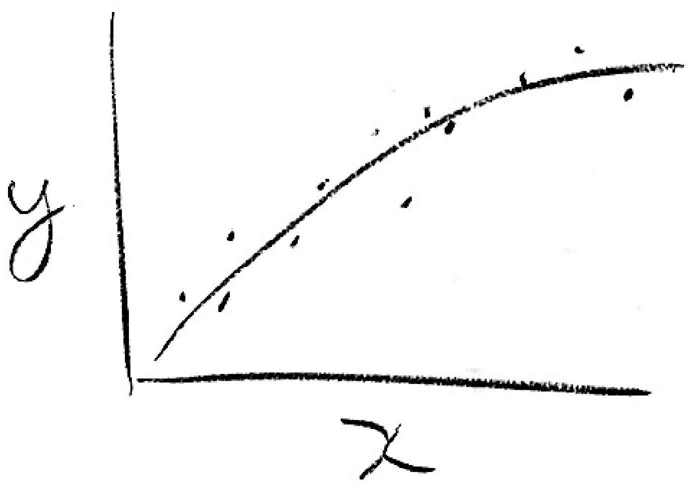
Overall algorithm:

1. Create a model algorithm for $\hat{f}(x)$
2. Use a training algorithm to find parameter values of $\hat{f}(x)$
3. Use an inference algorithm to compare predictive skill among models (model families, tuning parameters, x sets, etc).

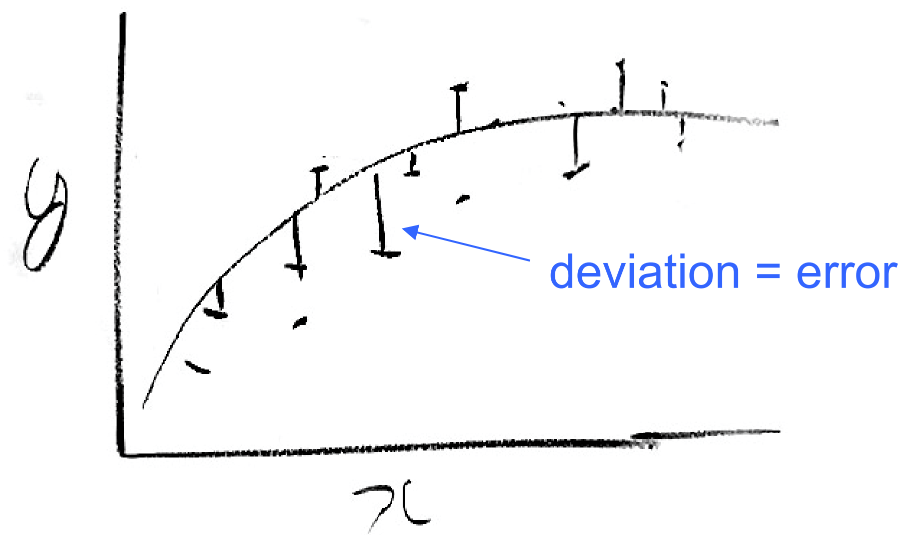
Inference algorithm

Basic idea: out-of-sample validation

Fit model to training dataset



Test model on validation dataset



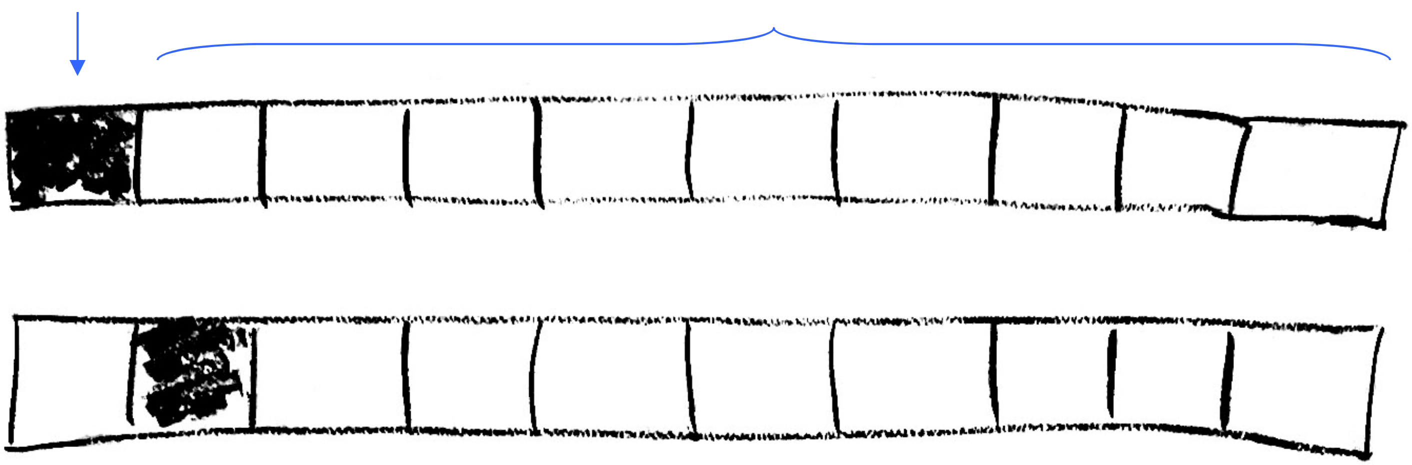
e.g. mean square error (MSE)

k-fold cross validation (CV)

Divide dataset into k parts (preferably randomly)

test (validation) data

training data



repeat with next test subset

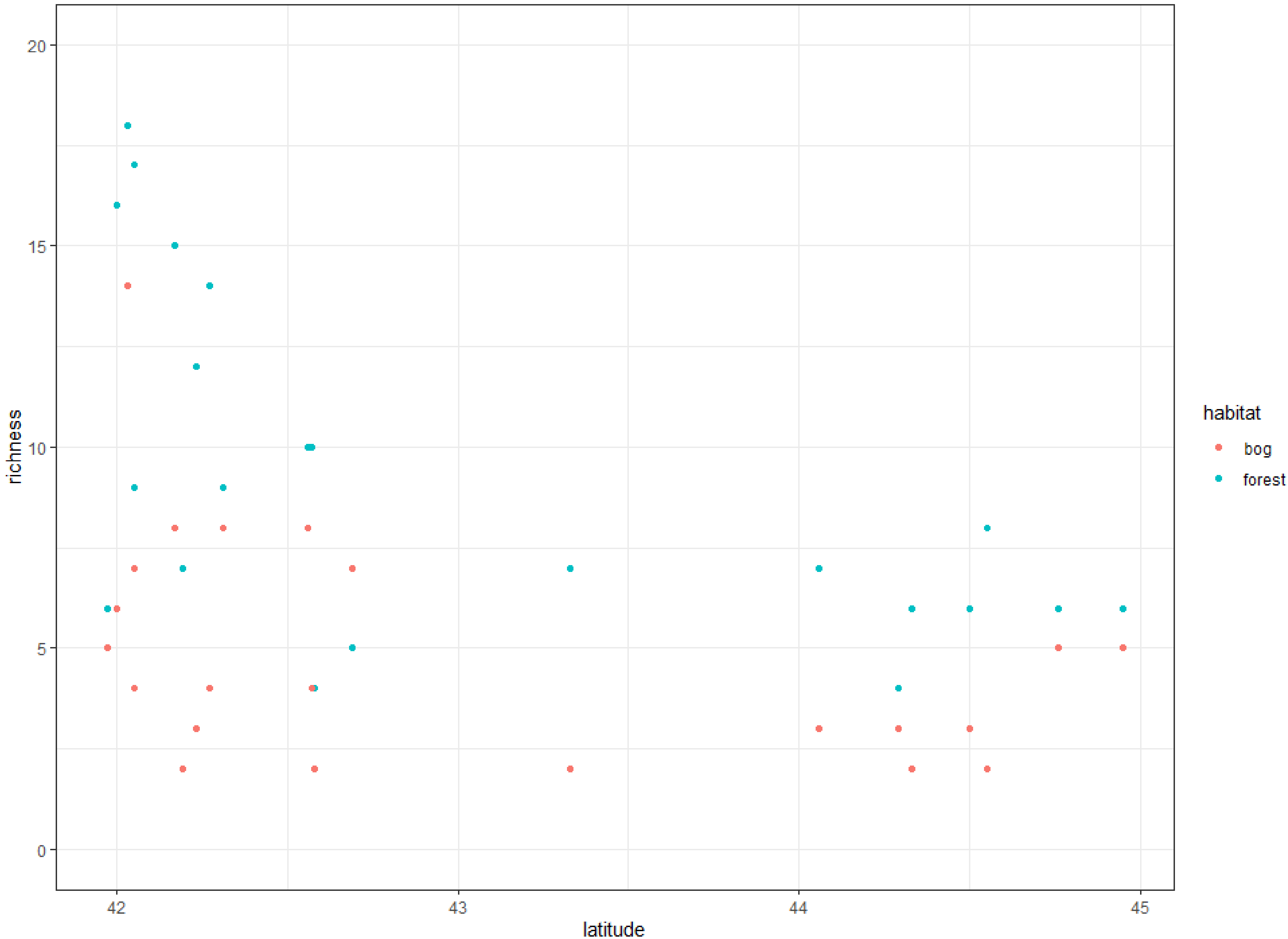
... repeat with each test subset

Regression & classification

- Regression:
  - numerical response variable
  - predict a numerical value given x
  - e.g. number of species given latitude
- Classification:
  - categorical response variable
  - predict the category given x
  - e.g. is it a bird, deer, tree, or mountain lion?
  - e.g. is it dead or alive?; present or absent?

Ants data

```
> head(ants)
  site habitat latitude elevation richness
1  TPB  forest    41.97       389        6
2  HBC  forest    42.00         8       16
3  CKB  forest    42.03       152       18
4  SKP  forest    42.05         1       17
5   CB  forest    42.05       210        9
6   RP  forest    42.17        78       15
```


richness
20
15
10
5
0
42
43
44
45
latitude
habitat
bog
forest

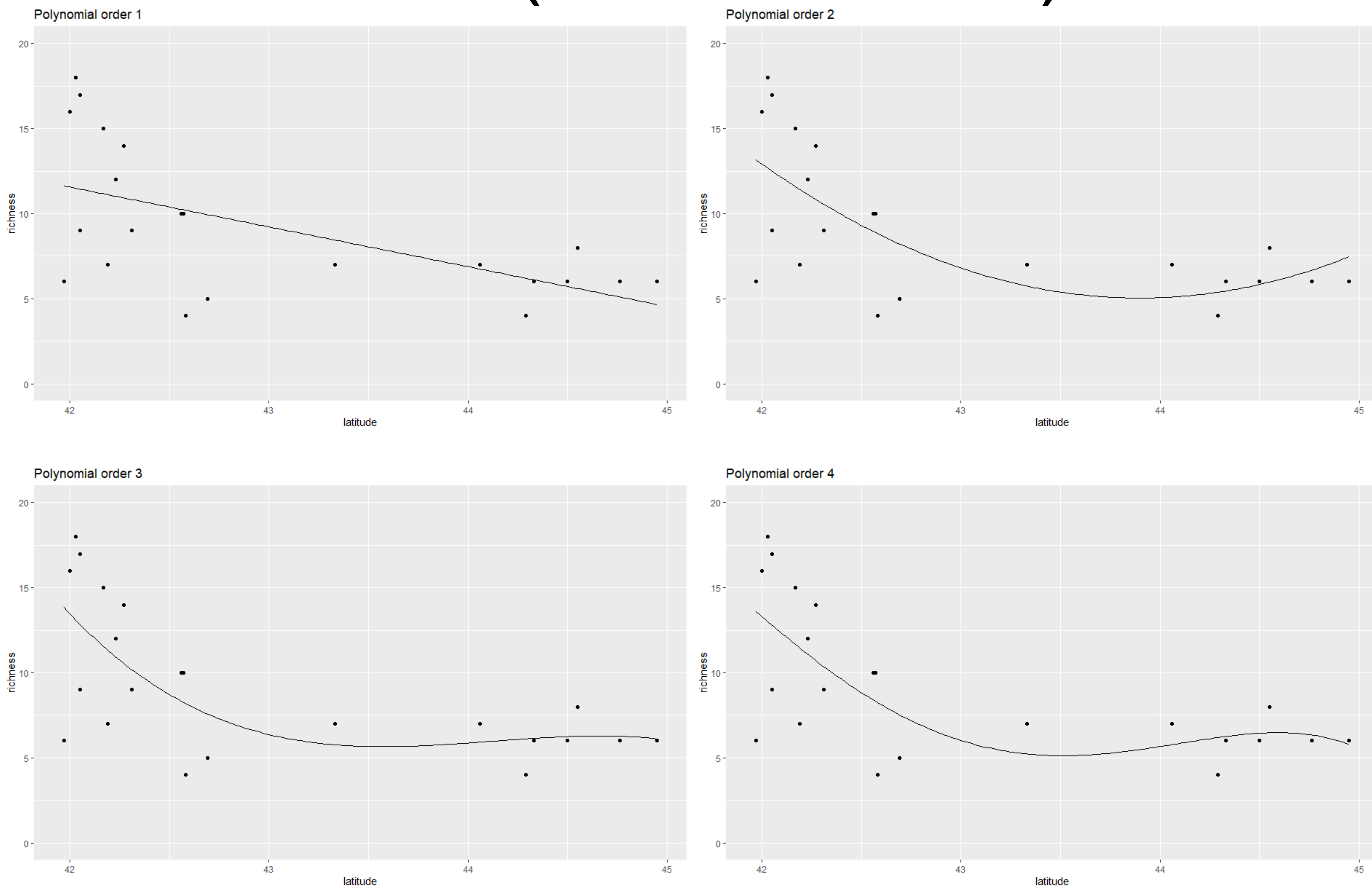
Basic full ML setup

- Polynomial example, 3 algorithms:
  - model: flexible function $\hat{f}(x)$; polynomial linear model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ \ldots + \beta_m x^m$$

m=order

Ants (forest habitat)



Basic full ML setup

- Polynomial example, 3 algorithms:
  - model: flexible function $\hat{f}(x)$; polynomial linear model

    $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots + \beta_m x^m$ m=order

  - training: optimize least squares objective function
  - minimize $SSQ = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ for training data

```
lm(richness ~ poly(latitude, order), data=forest_ants)
```

Basic full ML setup

- Polynomial example, 3 algorithms:
  - model: flexible function $\hat{f}(x)$; polynomial linear model

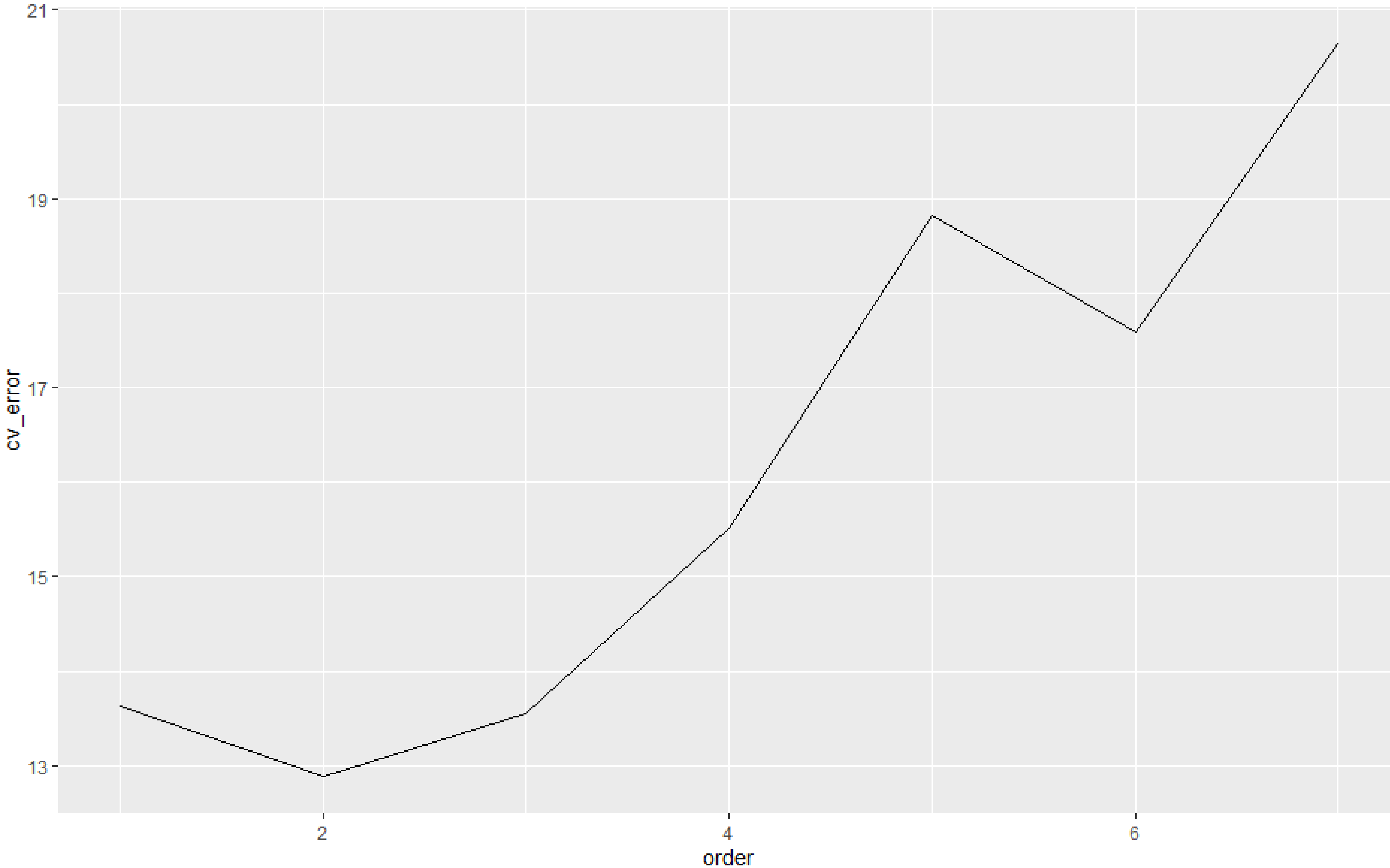
    $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ ... + \beta_m x^m$ m=order
  - training: optimize least squares objective function
  - minimize $SSQ = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ for training data
  - inference: tuning parameter (order of poly); k-fold cross validation

Inference algorithm

k-fold cross validation

```
cv_ants <- function(k, order) {
    forest_ants$fold <- random_folds(nrow(forest_ants), k)
    e <- rep(NA, k)
    for ( i in 1:k ) {
        test_data <- forest_ants %>% filter(fold == i)
        train_data <- forest_ants %>% filter(fold != i)
        poly_trained <- lm(richness ~ poly(latitude, order), data=train_data)
        pred_richness <- predict(poly_trained, newdata=test_data)
        e[i] <- mean((test_data$richness - pred_richness) ^ 2)
    }
    cv_error <- mean(e)
    return(cv_error)
}
```

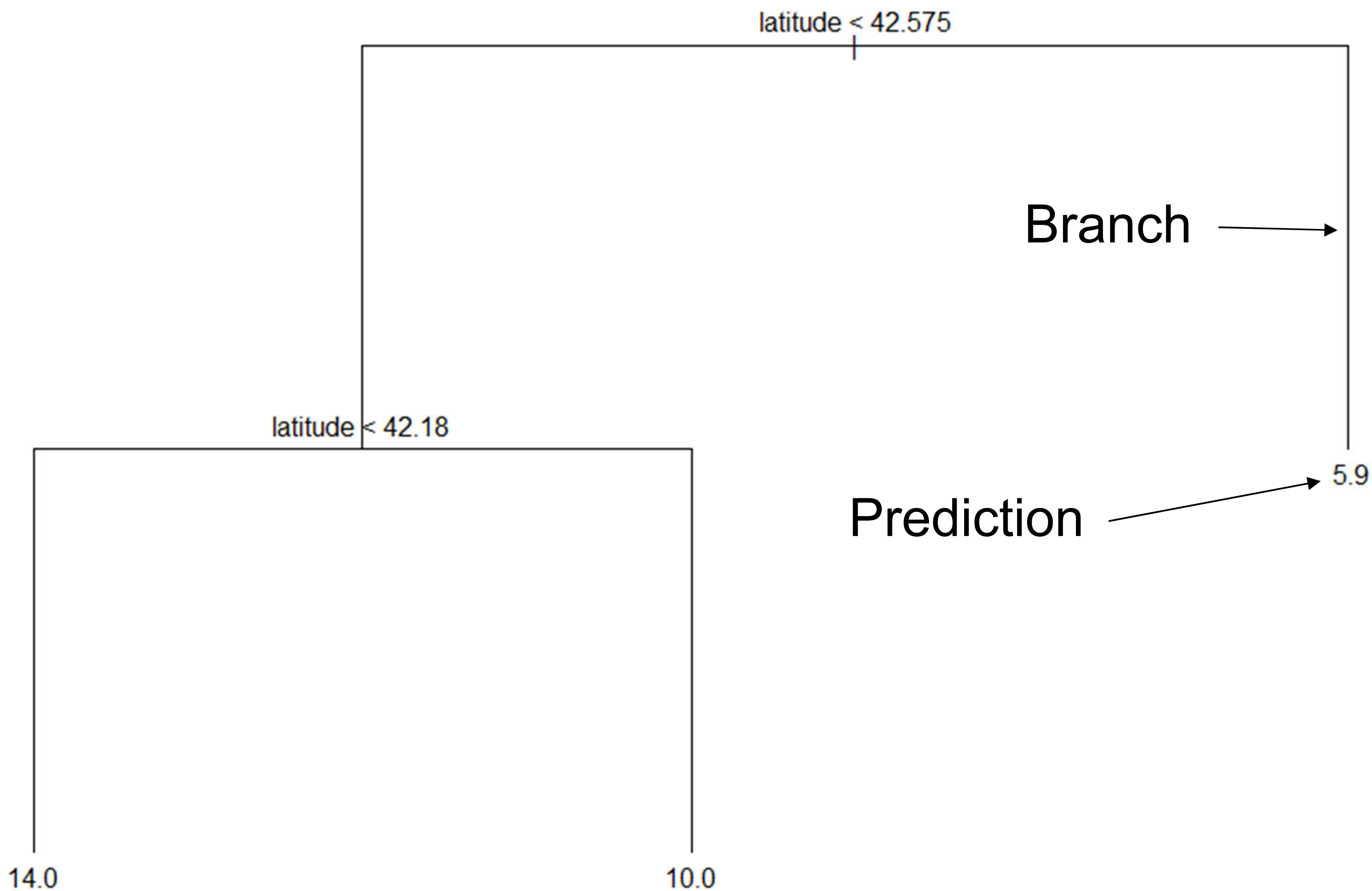
Leave one out CV (k-fold CV, k=n)

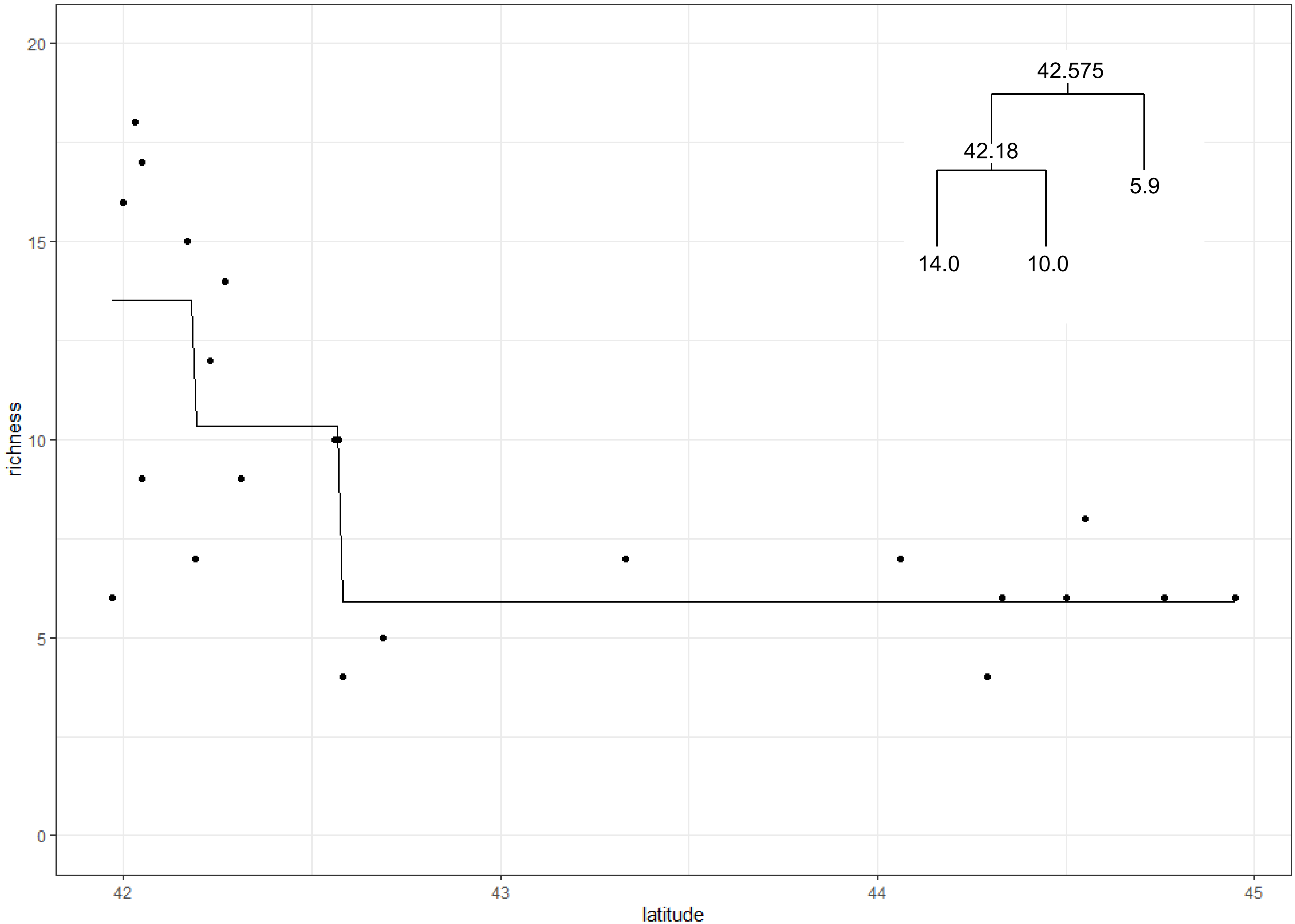


Basic ML algorithms

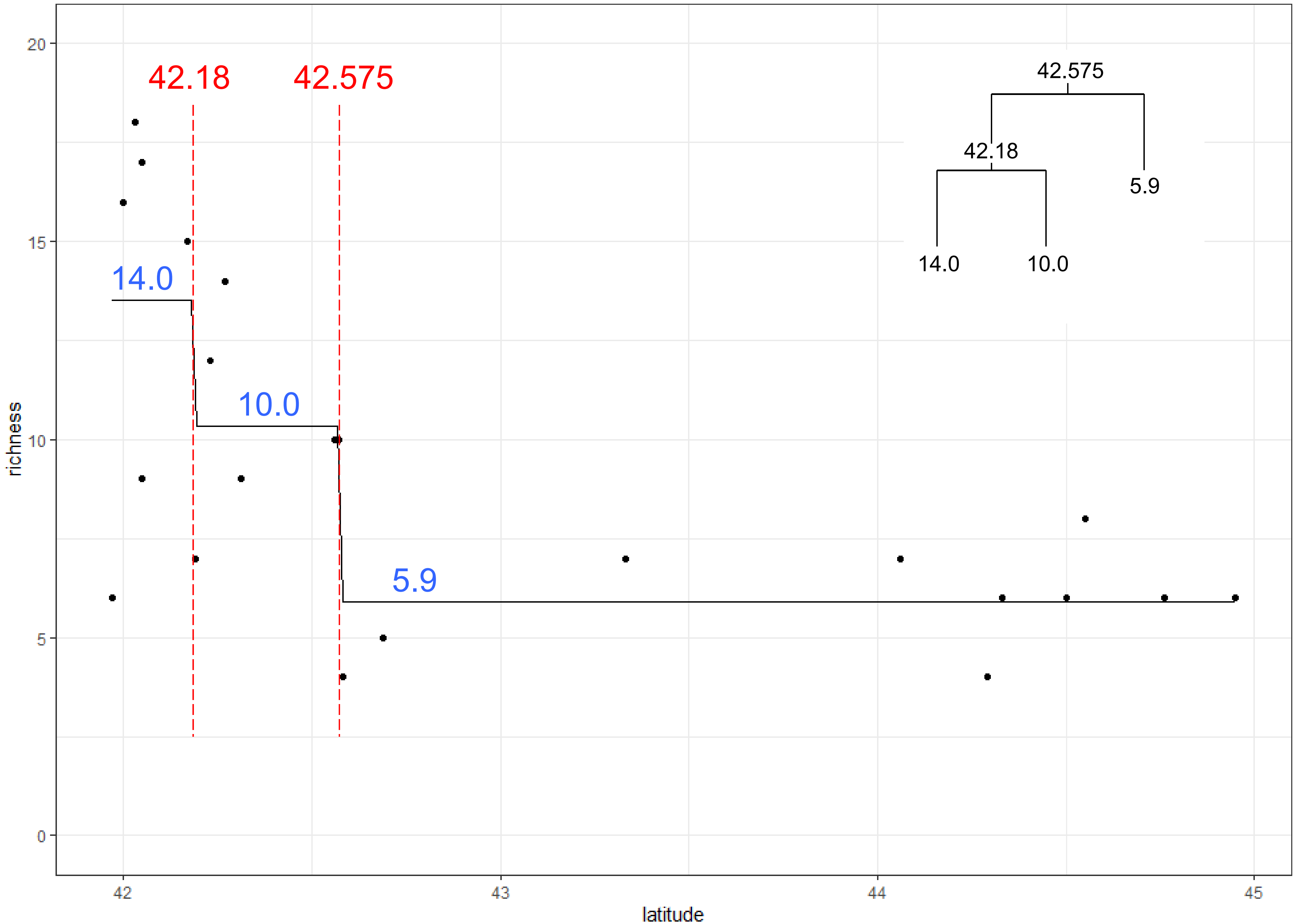
- Ensemble methods
  - Bagging
  - Random forest
  - Boosting
- Neural networks

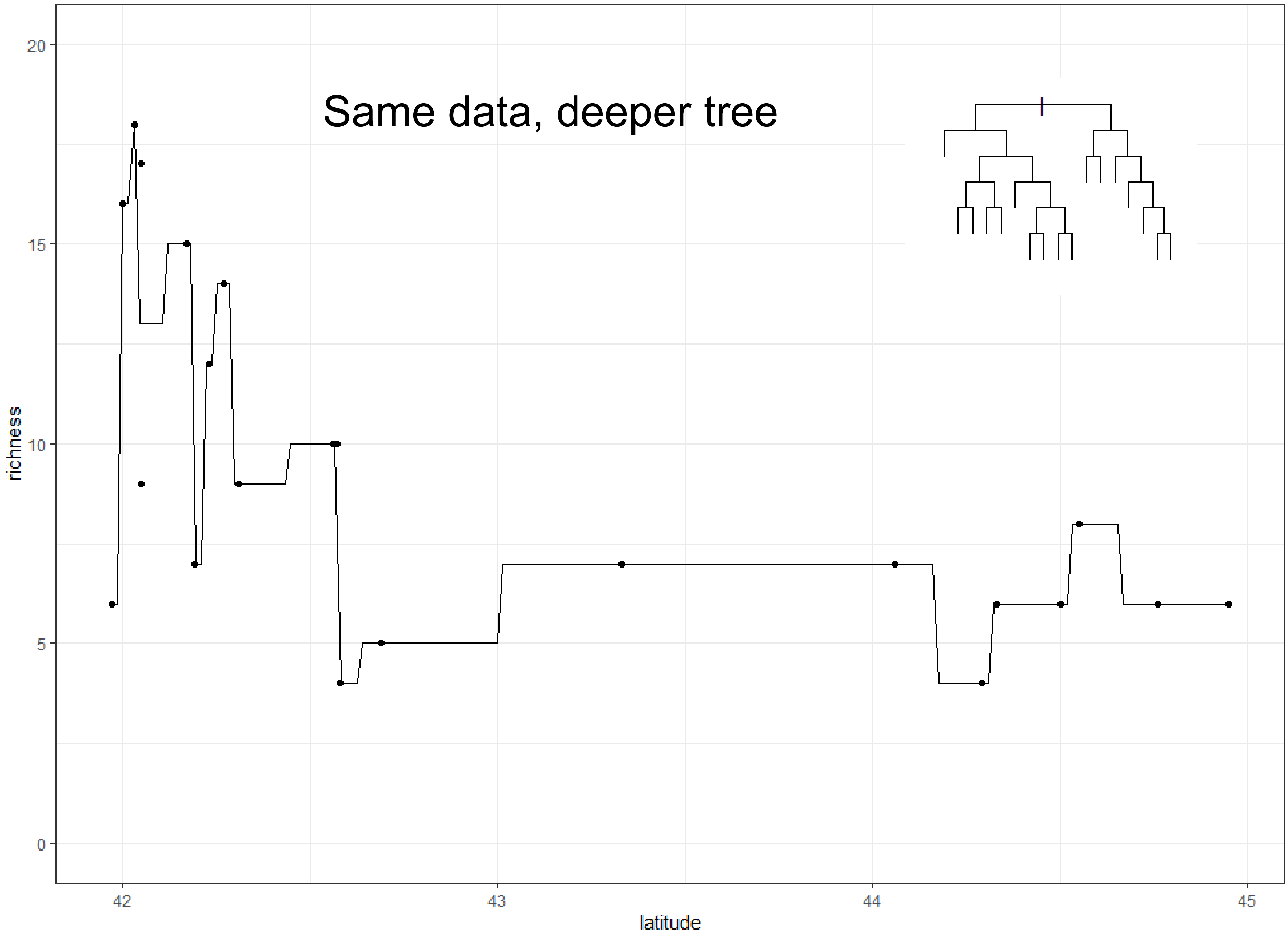
Regression tree base model

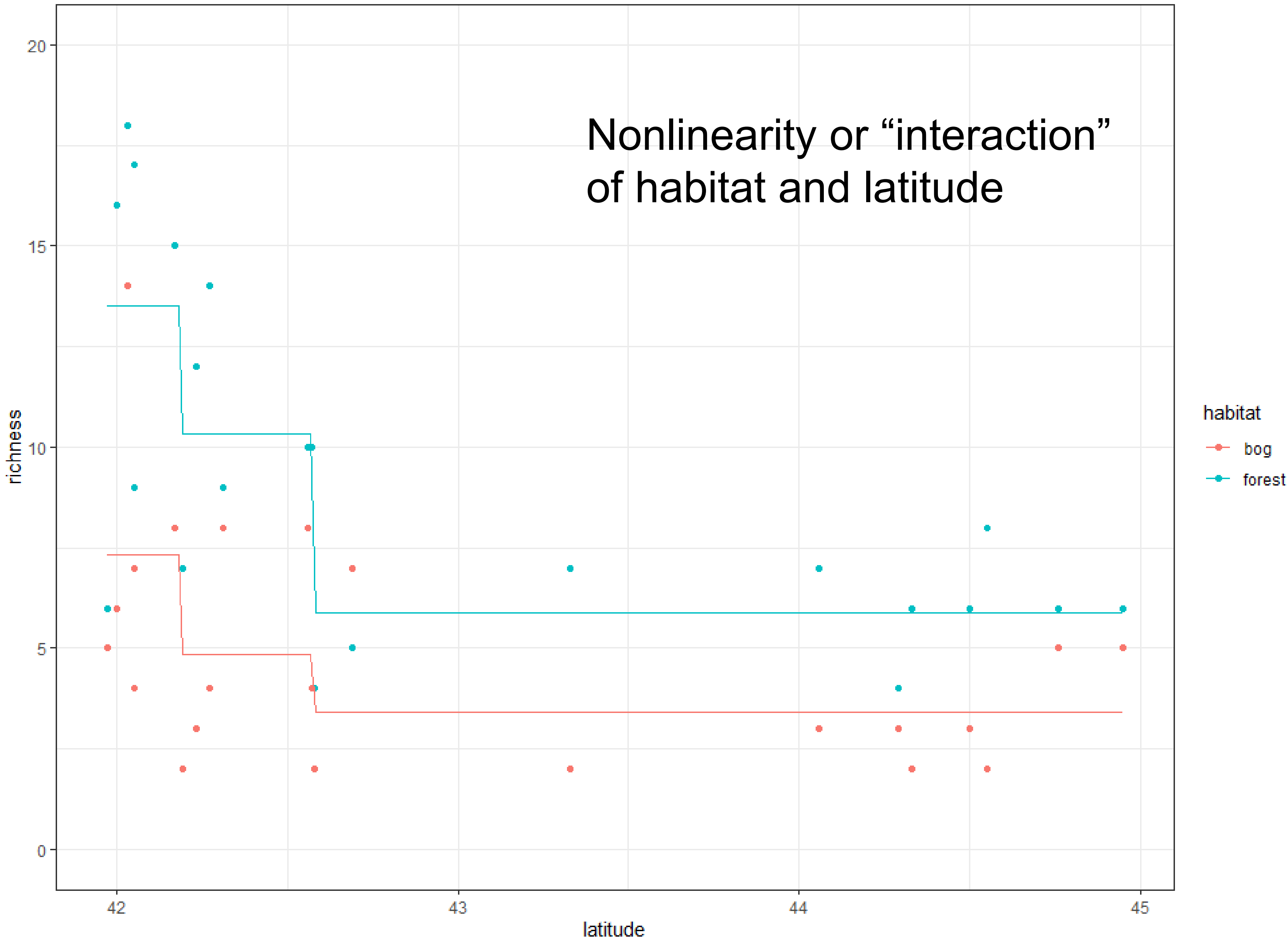


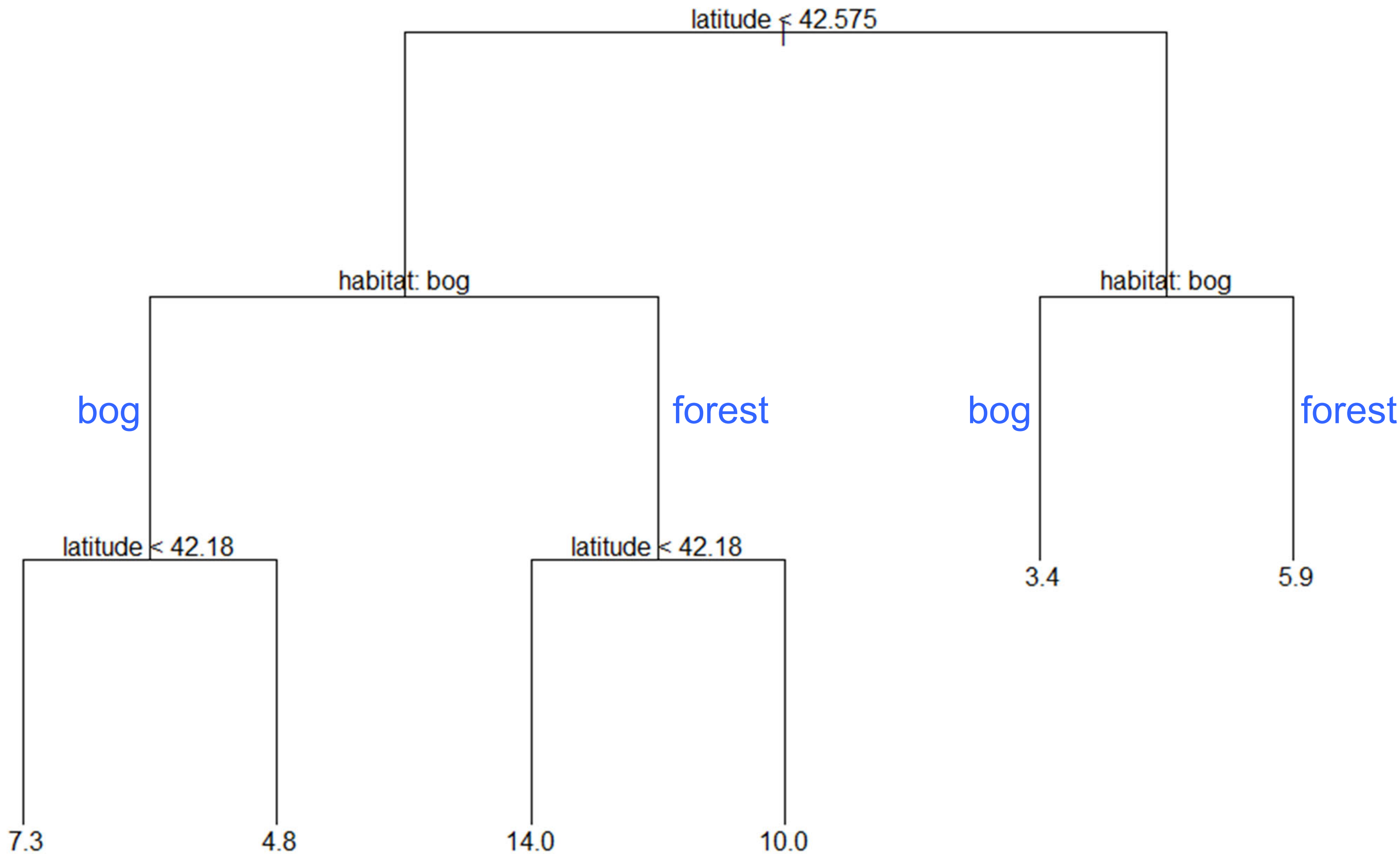


42.575
42.18
5.9
14.0
10.0
20
15
10
5
0
richness
42
43
44
45
latitude


42.18
42.575
14.0
10.0
5.9
42.575
42.18
5.9
14.0
10.0
richness
latitude
0
5
10
15
20
42
43
44
45


Same data, deeper tree
20
15
10
5
0
richness
42
43
44
45
latitude


Nonlinearity or “interaction”
of habitat and latitude
richness
latitude
habitat
bog
forest
0
5
10
15
20
42
43
44
45



latitude < 42.575
habitat: bog
habitat: bog
bog
forest
bog
forest
latitude < 42.18
latitude < 42.18
3.4
5.9
7.3
4.8
14.0
10.0

Ensemble methods

- Train many models
- Average the models to predict
- Averaging reduces variance

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

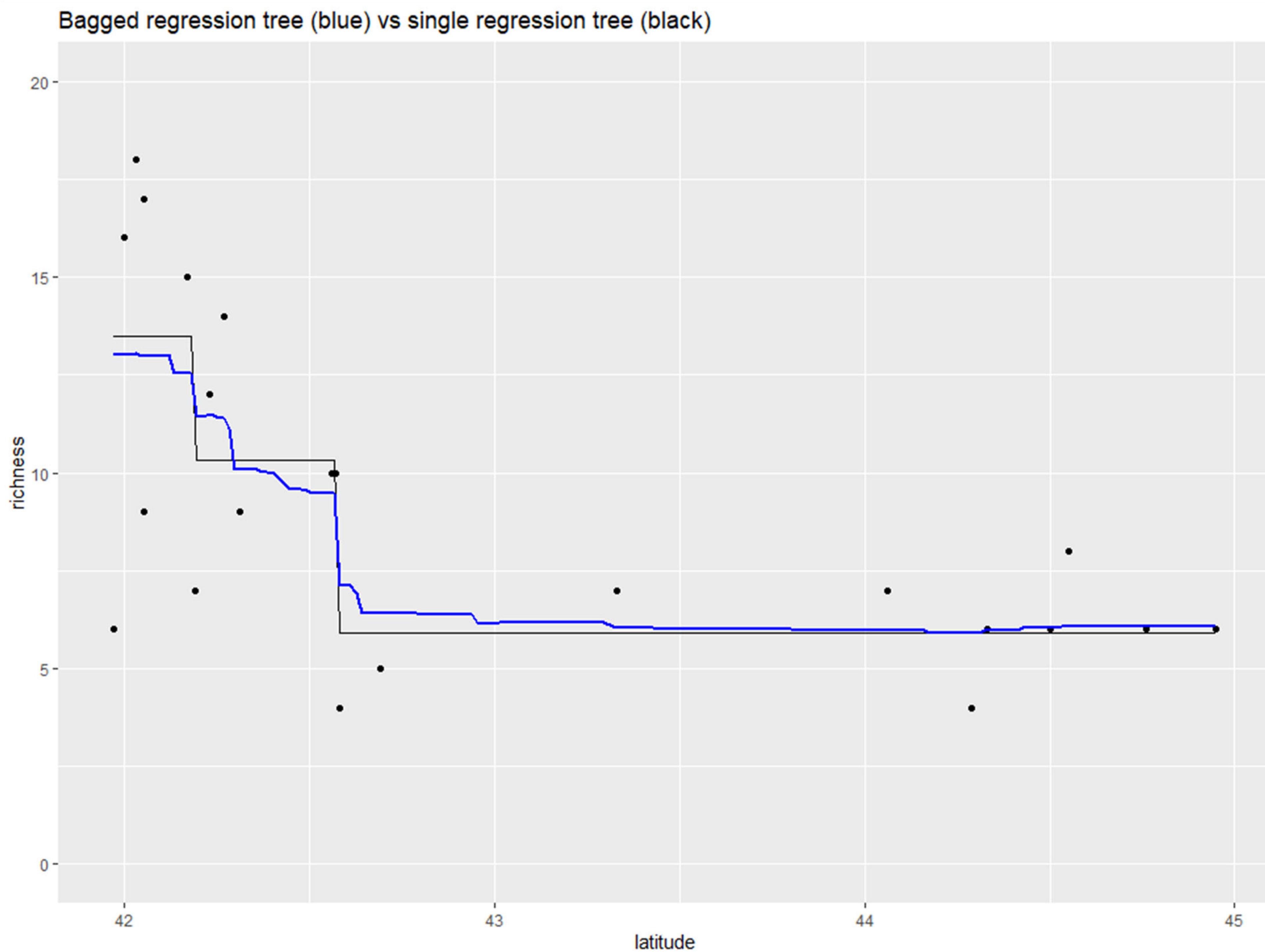
Bagging

- Bootstrap
  - form new datasets by resampling from the data
- Aggregate
  - average over bootstrapped model fits

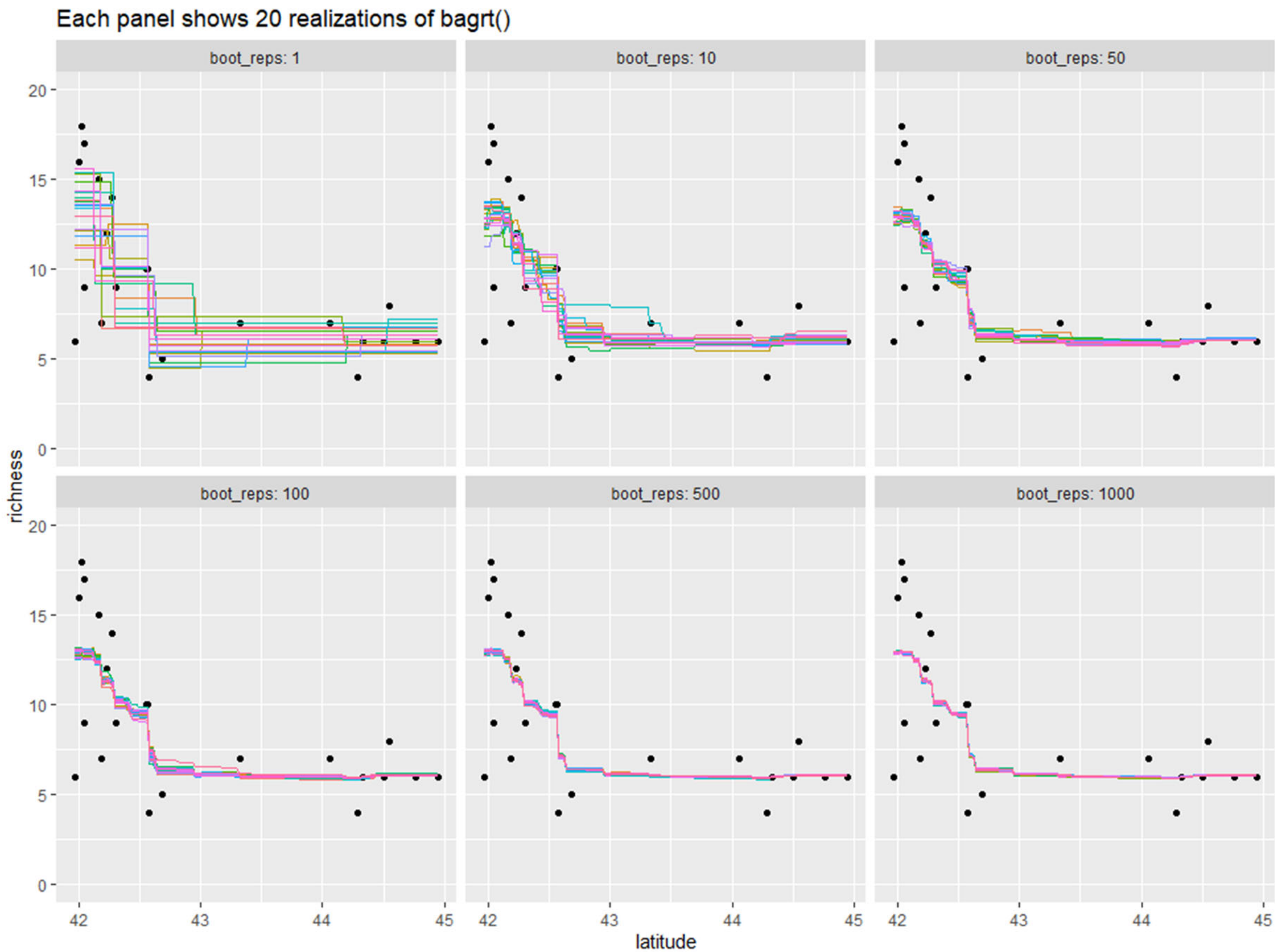
Bagging algorithm

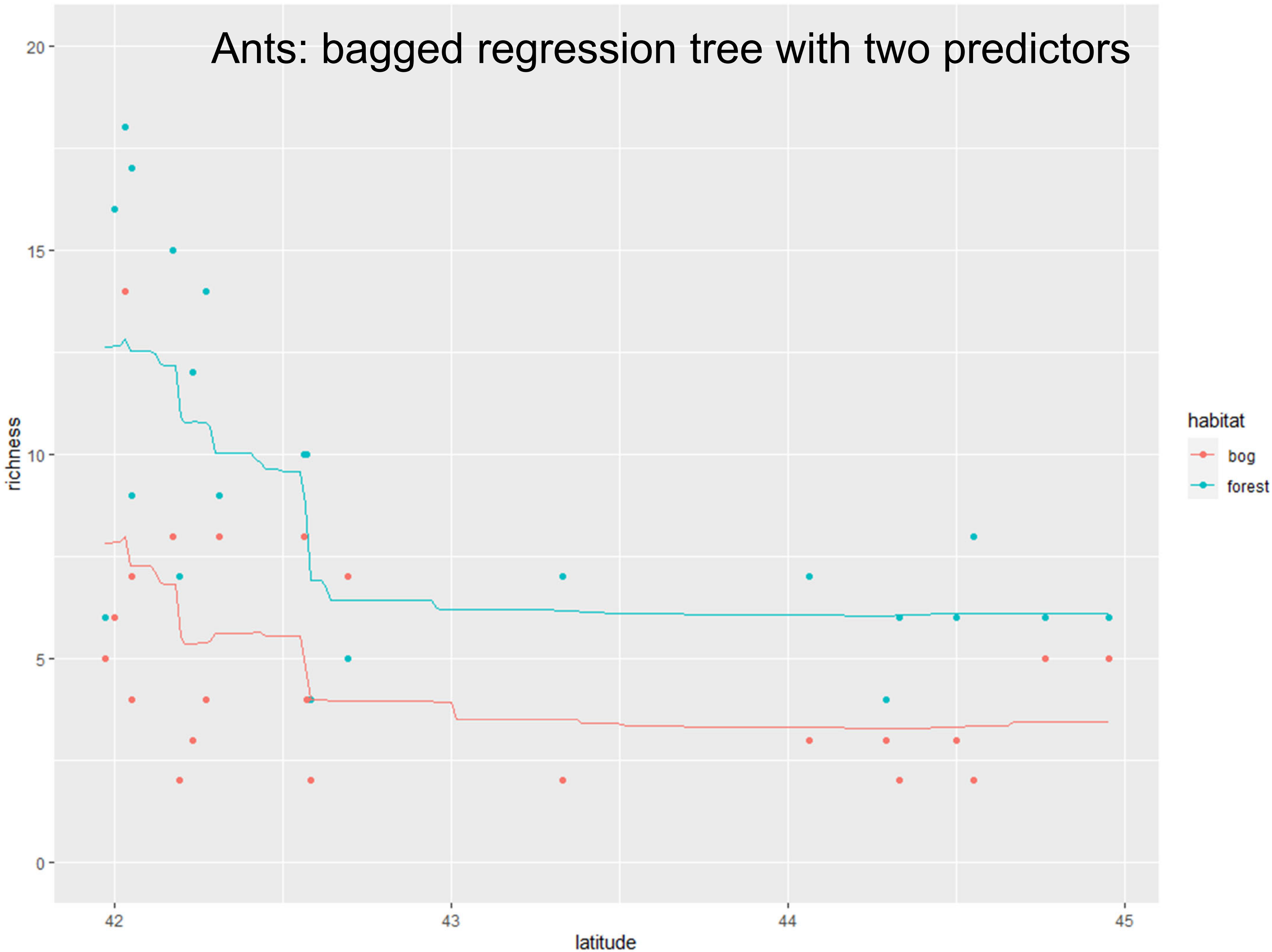
for many repetitions
    resample the data with replacement
    train the base model
    record prediction
final prediction = mean of predictions

Base model: can be any type of model


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

Bagging reduces prediction variance





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

Random forest

Algorithm

for many repetitions

  randomly select m predictor variables

  resample the data (rows) with replacement

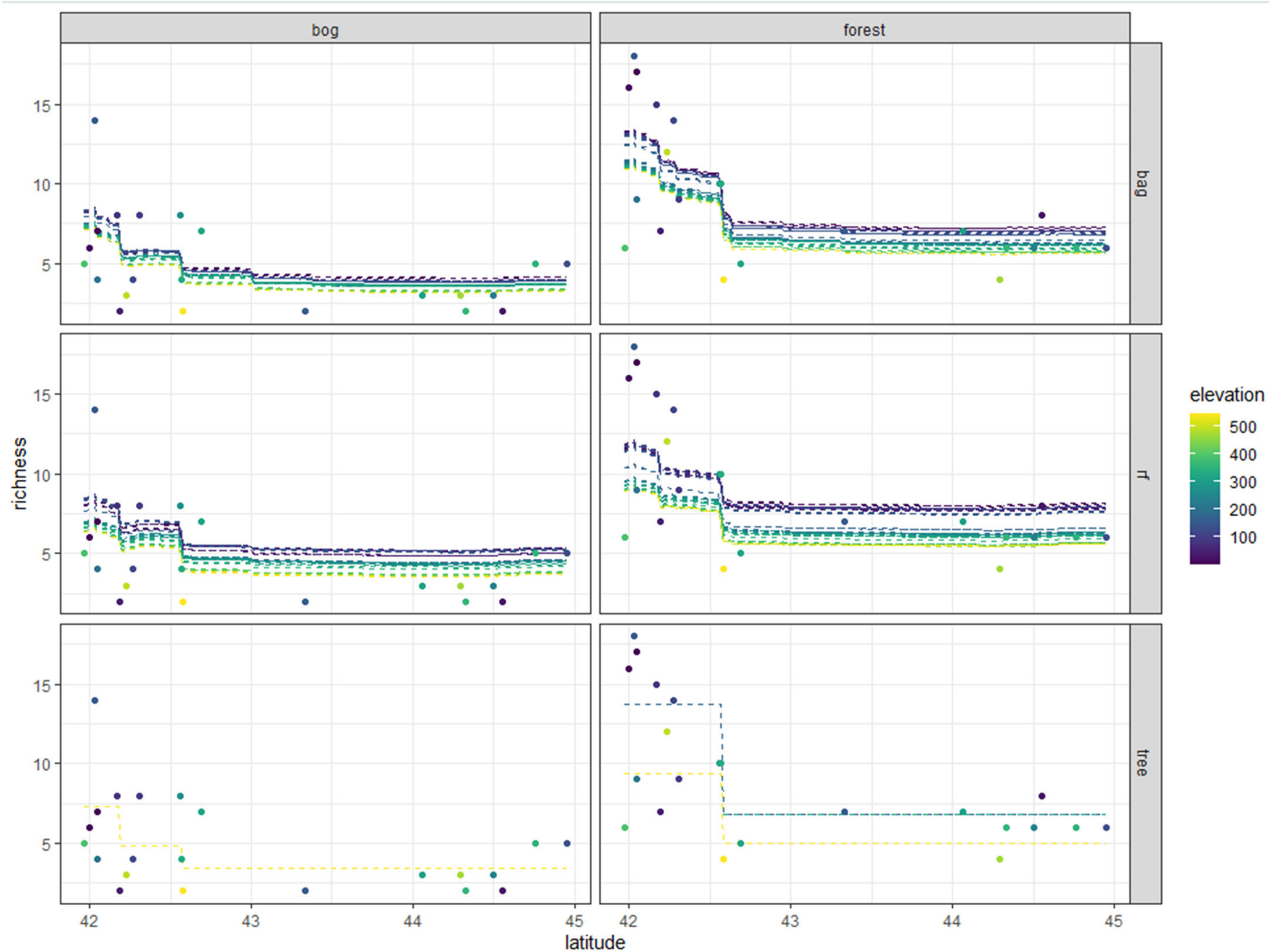
  train the tree model

  record prediction

final prediction = mean of predictions

R packages

- randomForest
  - original Breimen (2001) algorithm
  - Fortran
- ranger
  - fast implementation
  - C++

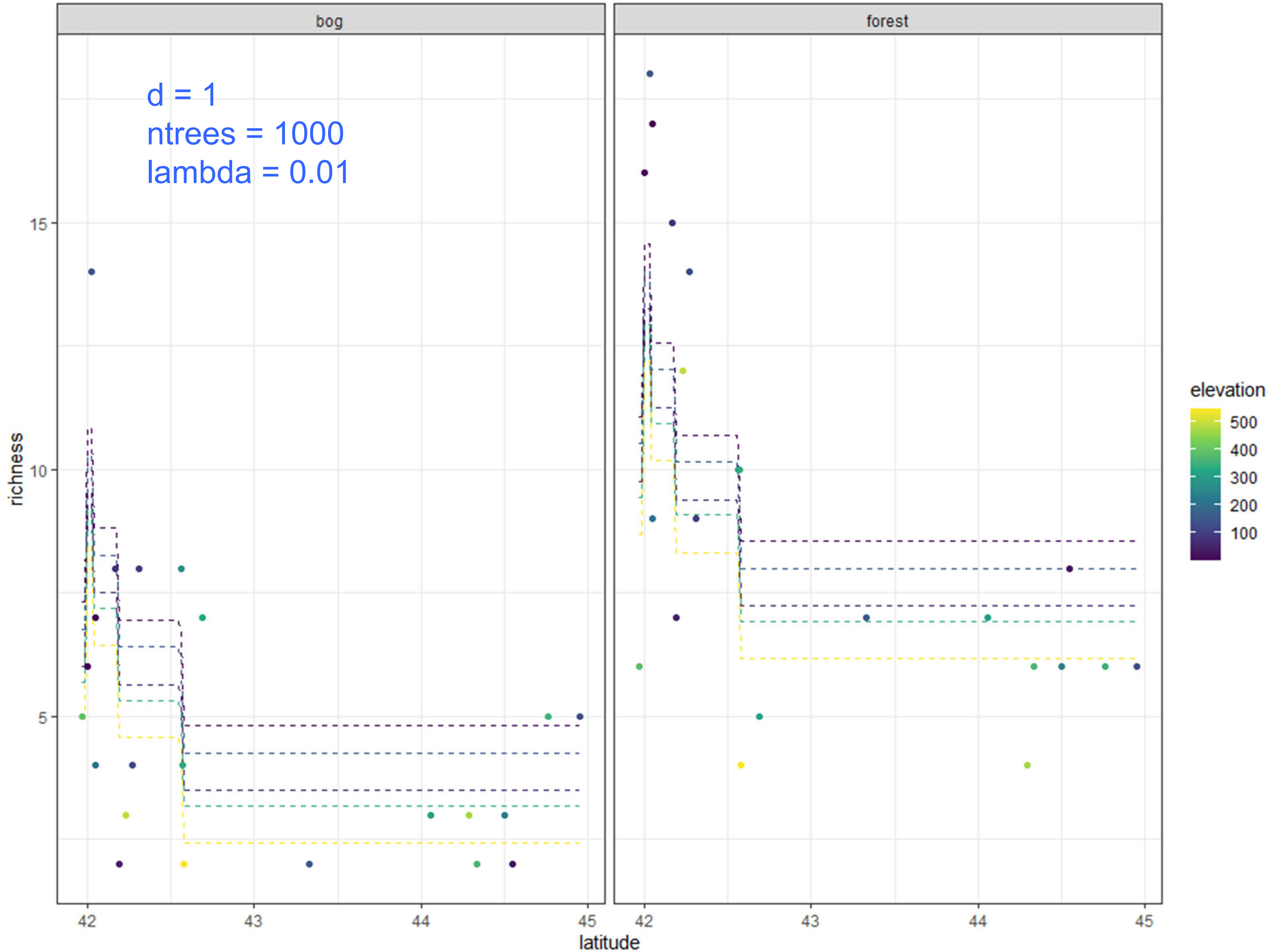

bog
forest
bag
rf
tree
richness
latitude
elevation
500
400
300
200
100
15
10
5
42
43
44
45

Boosting algorithm

- Too complex for this quick overview
- Basic idea: learn slowly by building up an ensemble iteratively from many models, each with small weight
- Key tuning parameter: learning rate
- Can include aspects of bagging and RF
- Training algorithm: gradient descent

Boosting packages in R

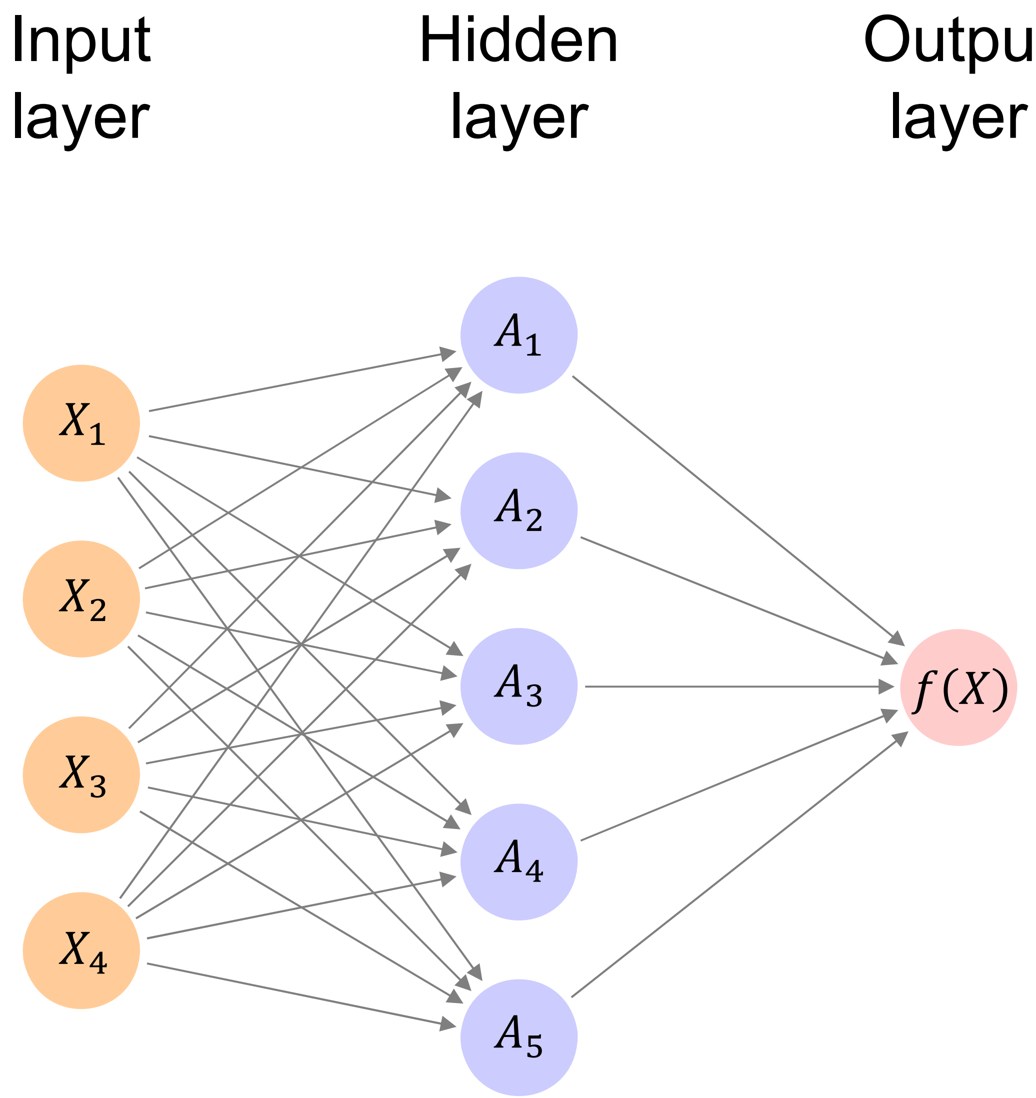
- gbm: gradient boosting machines
  - boosted decision trees
  - good, stable, maintained
  - retired (no new features)
- xgboost: extreme gradient boosting
  - R interface to very fast C++ library
  - additional algorithm innovations
  - current industry standard


bog
forest
d = 1
ntrees = 1000
lambda = 0.01
richness
15
10
5
42
43
44
45
latitude
elevation
500
400
300
200
100

Neural Networks in R

- keras
  - currently easiest to use
  - tensorflow (Google)
- torch
  - coming along (v 0.9)
  - PyTorch (Facebook, now Linux Foundation)

Single layer NN



Single layer NN

Input layer

Hidden layer

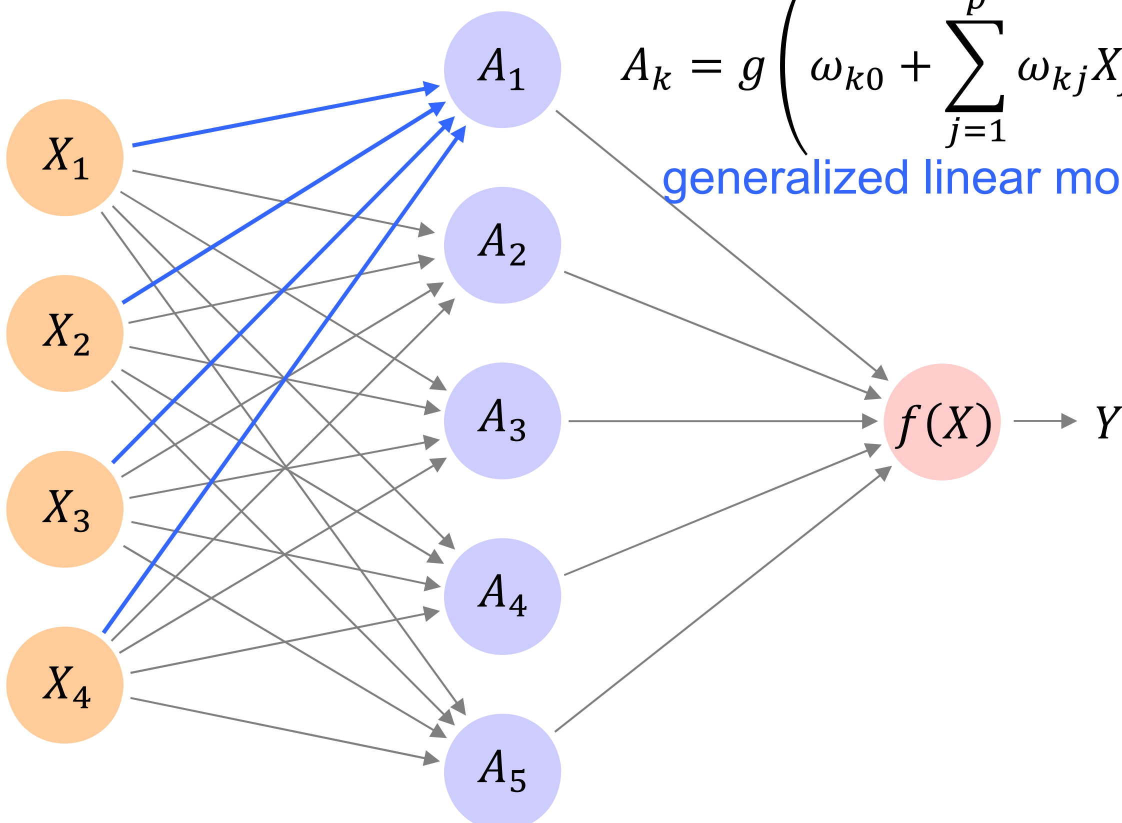
Output layer

Activation function

$$A_k = g\left(\omega_{k0} + \sum_{j=1}^{p} \omega_{kj} X_j\right)$$

generalized linear model

Transform from X to A



Single layer NN

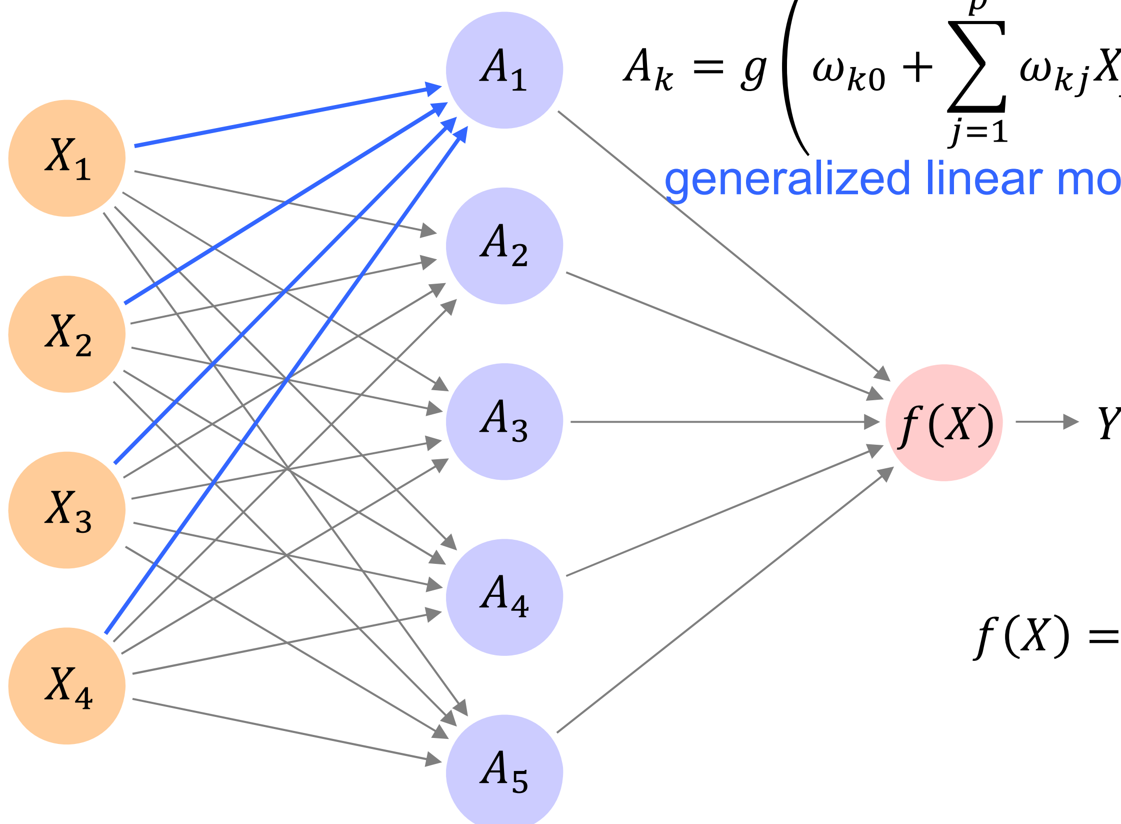
Input layer | Hidden layer | Output layer

Activation function

$$A_k = g\left(\omega_{k0} + \sum_{j=1}^{p} \omega_{kj} X_j\right)$$

generalized linear model

Transform from X to A



$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$

multiple linear regression

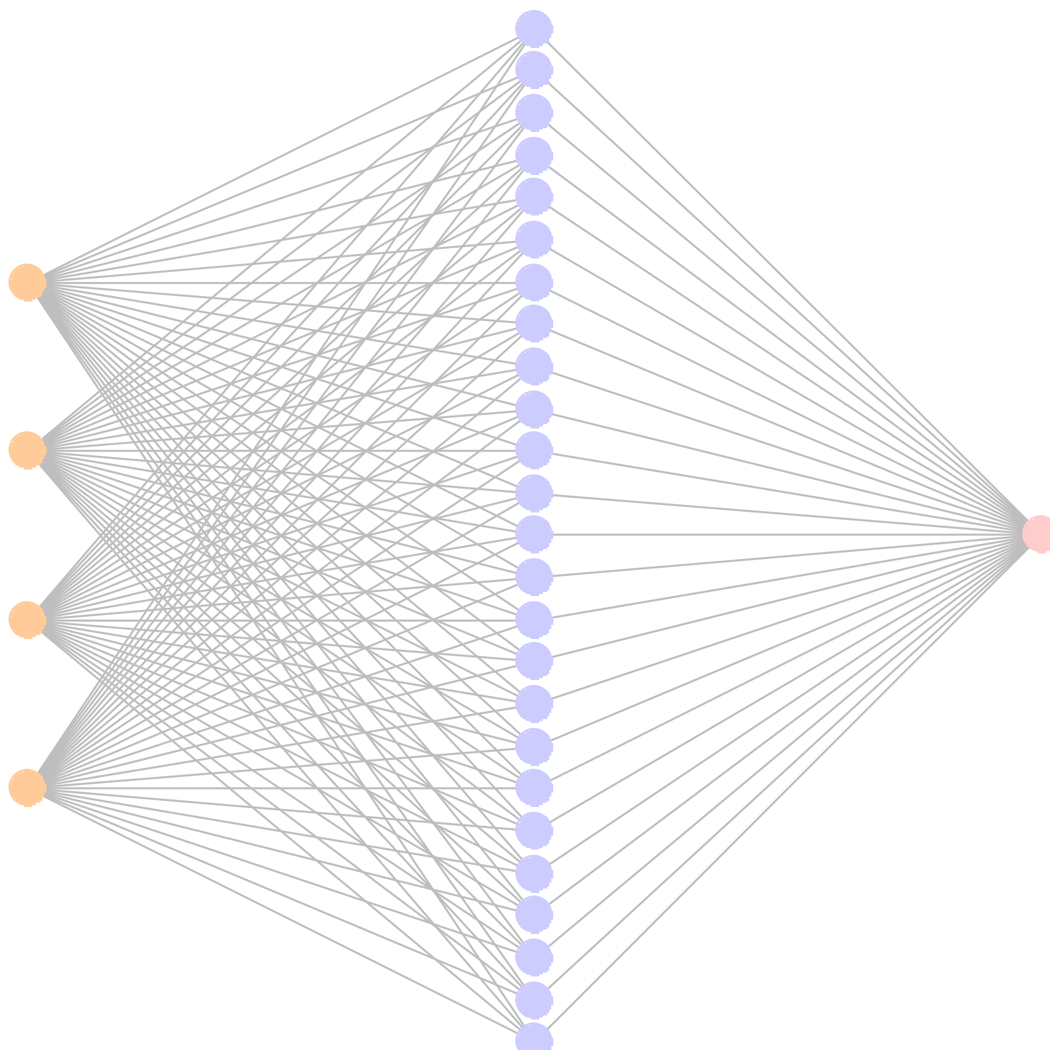
Training algorithm

- Stochastic gradient descent
  - with back propagation

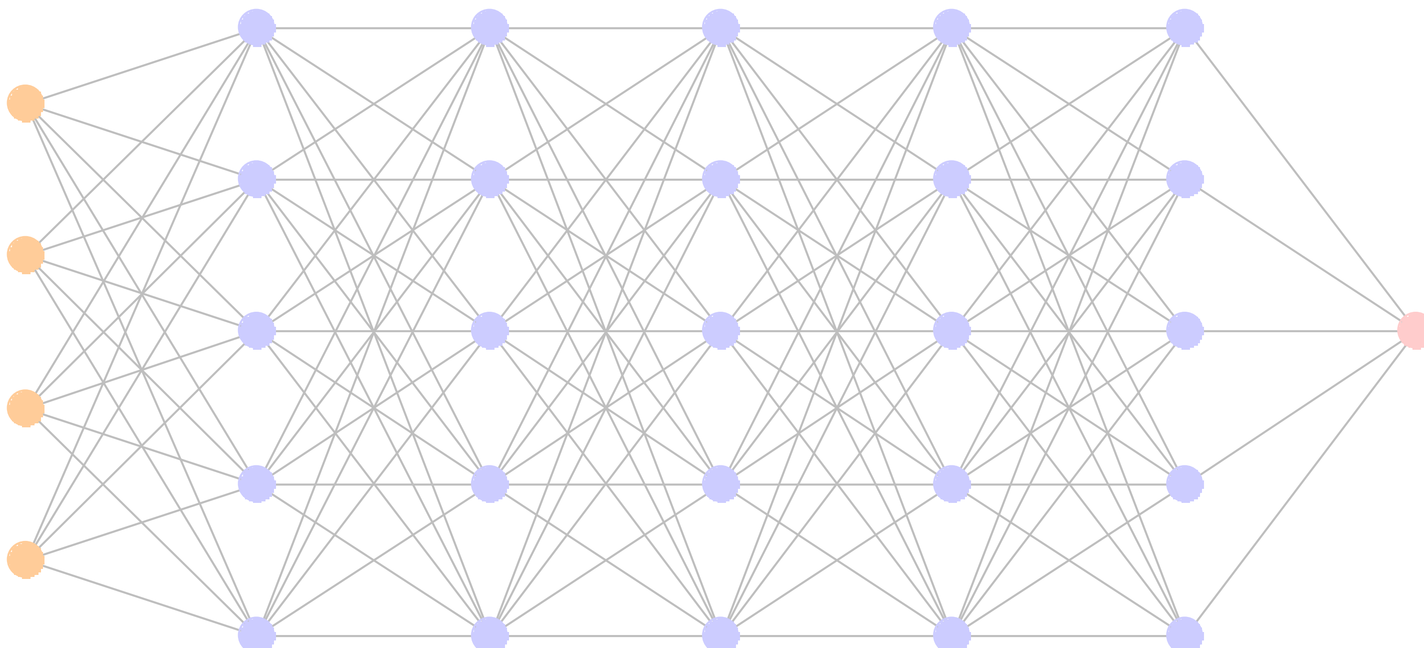
Deep learning

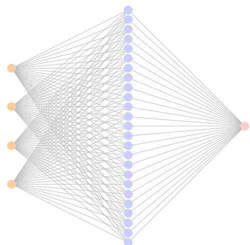
- Multilayer neural networks
  - aka deep feedforward networks
  - aka multilayer perceptrons (MLP)
- Model algorithm
  - expressiveness
    - ability to approximate complex nonlinearity
  - architecture: width versus depth

Wide: 25 hidden units

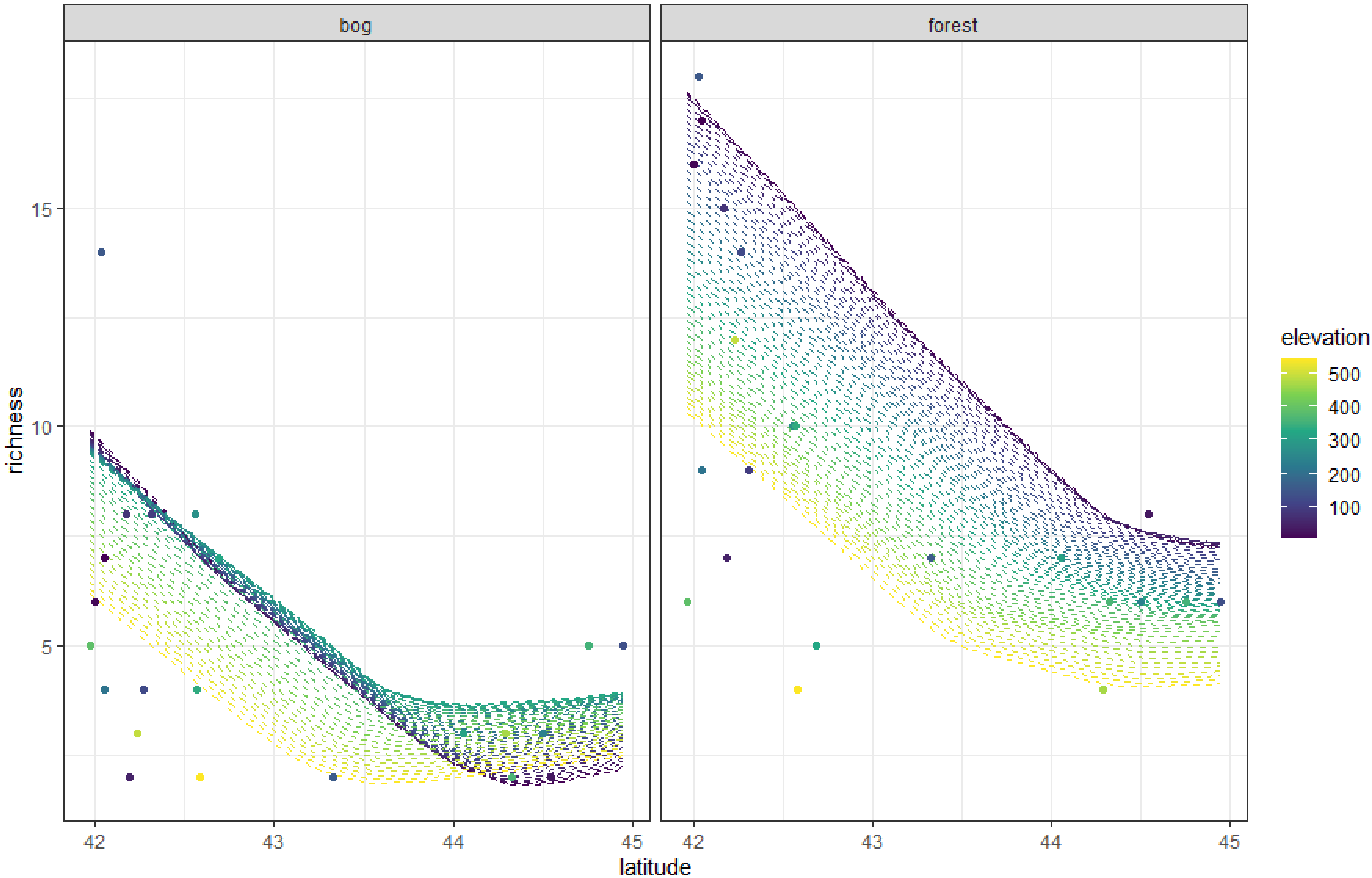


Deep: 25 hidden units

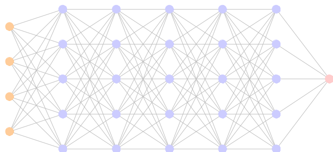


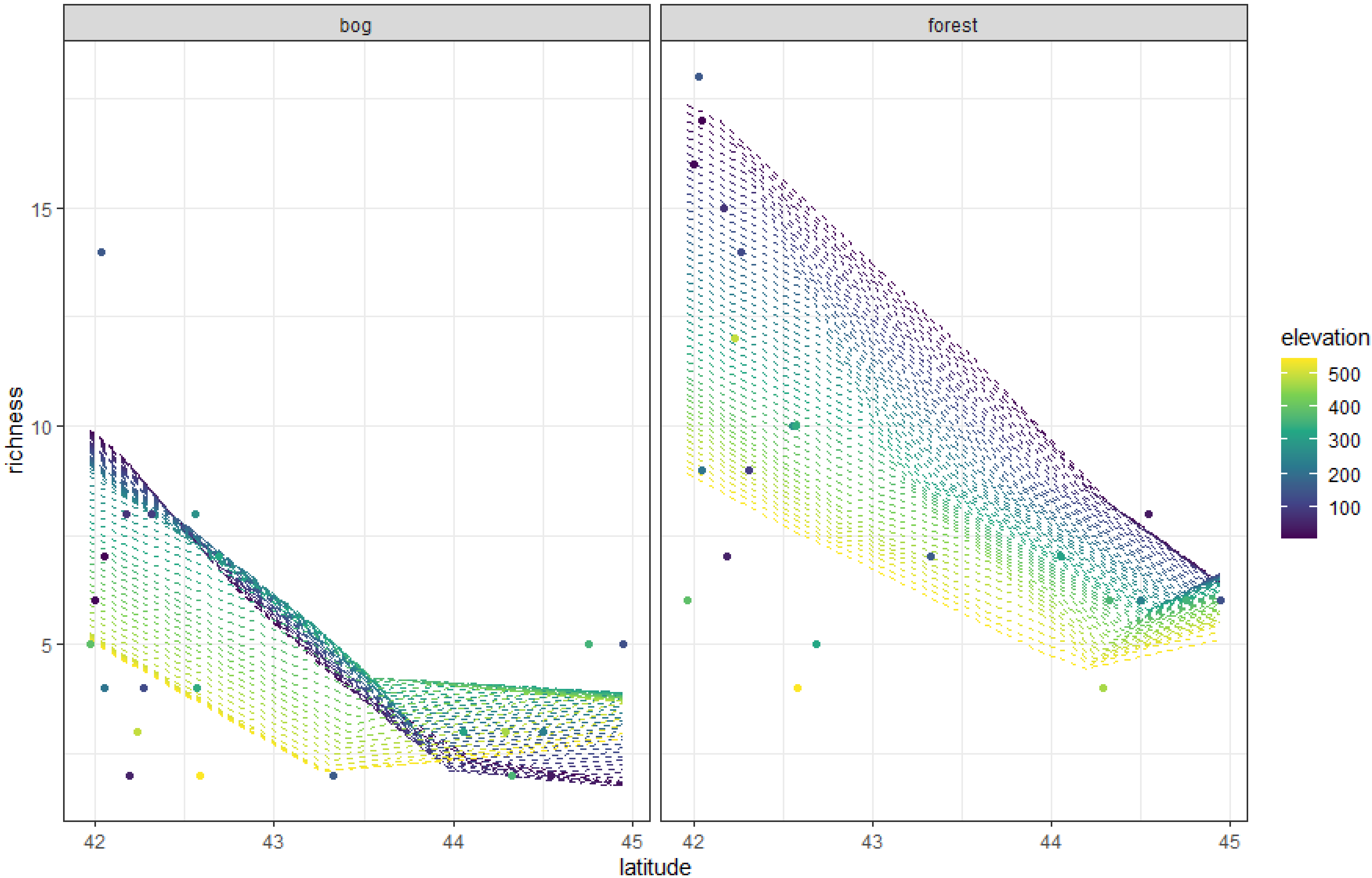


Ants data: wide



Ants data: deep

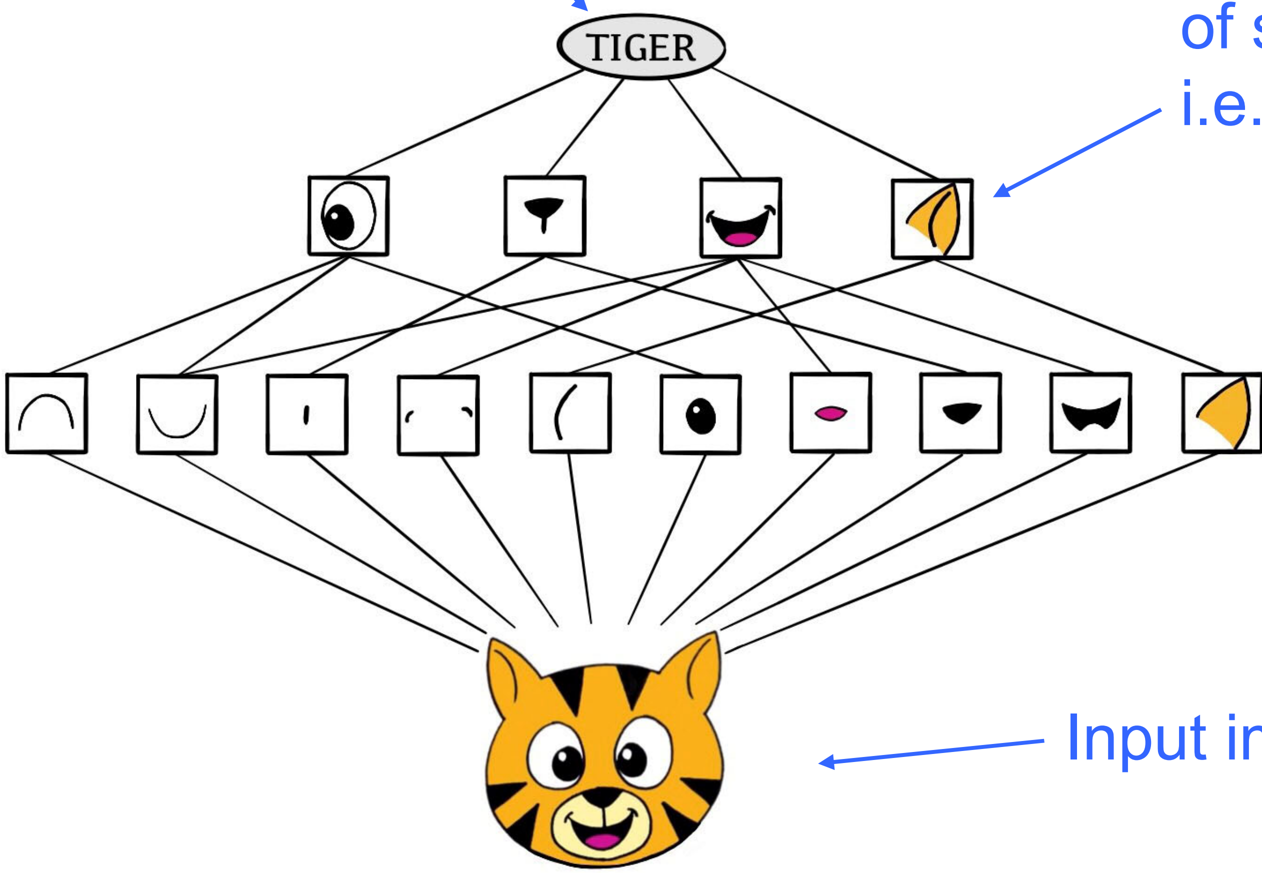




Convolutional NNs

Output is high-level concept

Conv layer 2
learns combinations
of small features
i.e. objects



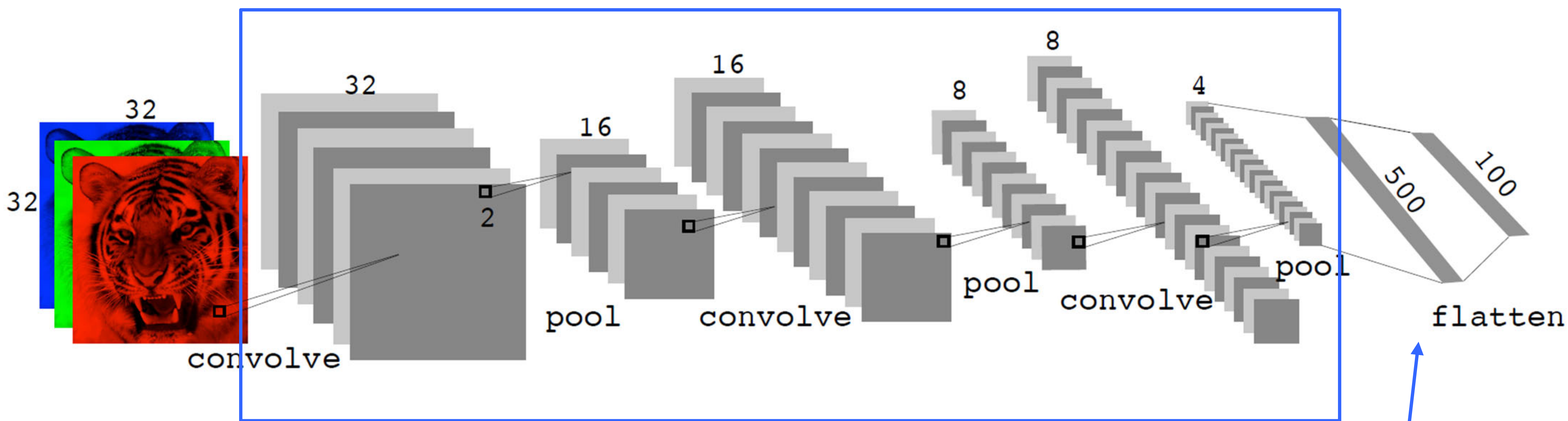
Conv layer 1
learns small
features
e.g. edges

Input image

ISLR 10.6

Transfer learning

Pretrained model on related big data
Retrain last 1-2 layers on specialized little data



Freeze convolve-pool layers (architecture, weights etc)

Reconfigure
and/or retrain
dense layers

ISLR 10.8

Rapid innovation

- Architectures
- Algorithms
- Recent example: transformer

Outlook

Automated data collection

+

machine learning

=

revolutionary