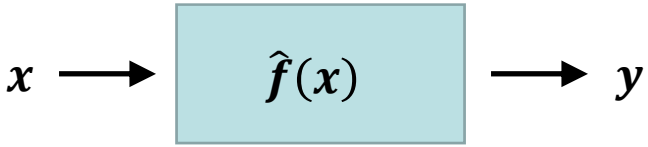
Prediction



Goal: find function $\hat{f}$ that has good predictive performance

Accurate on new observations of y

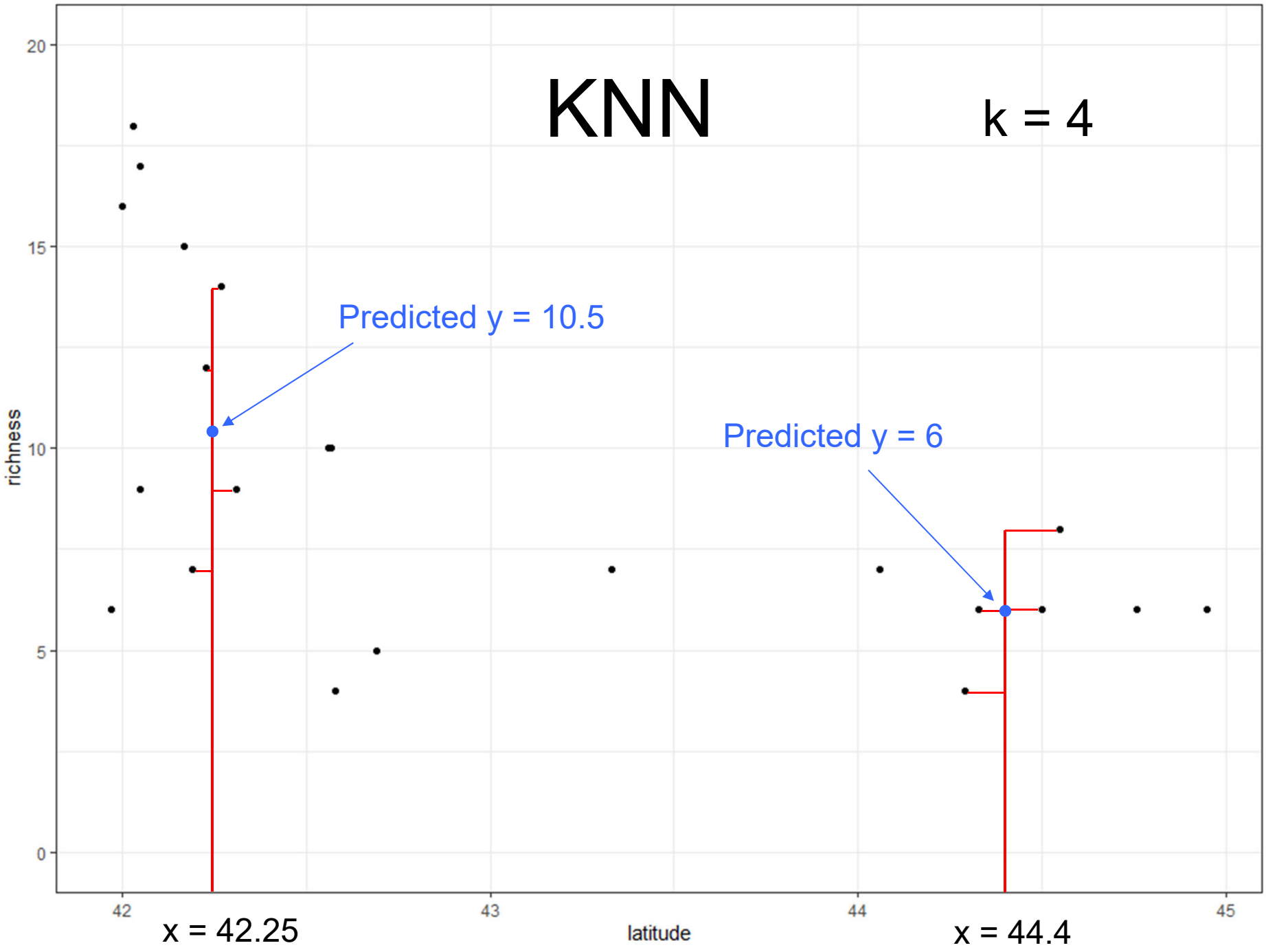
Out-of-sample accuracy
Bias-variance tradeoff
Cross validation and k-fold CV algorithm

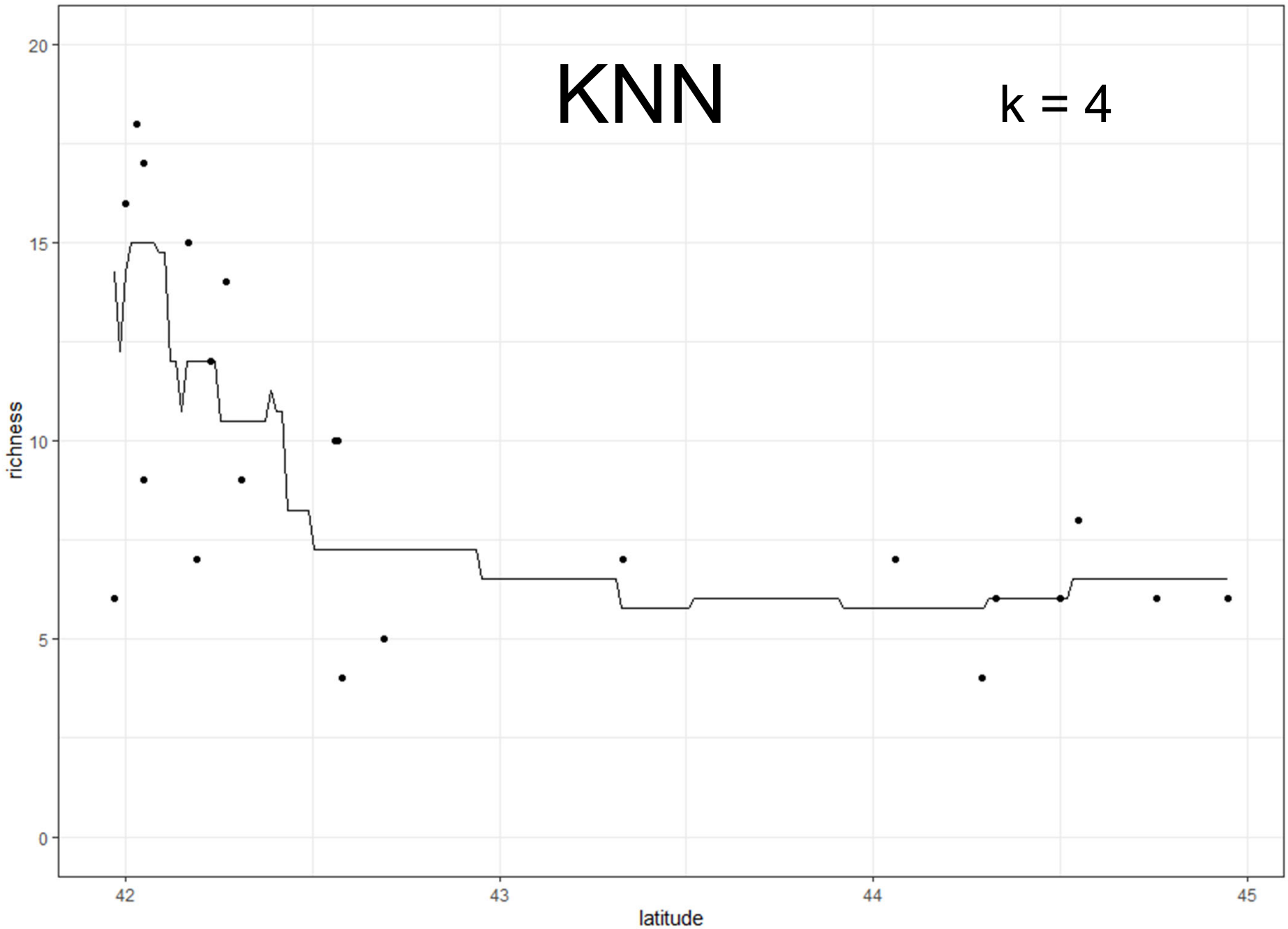
Today

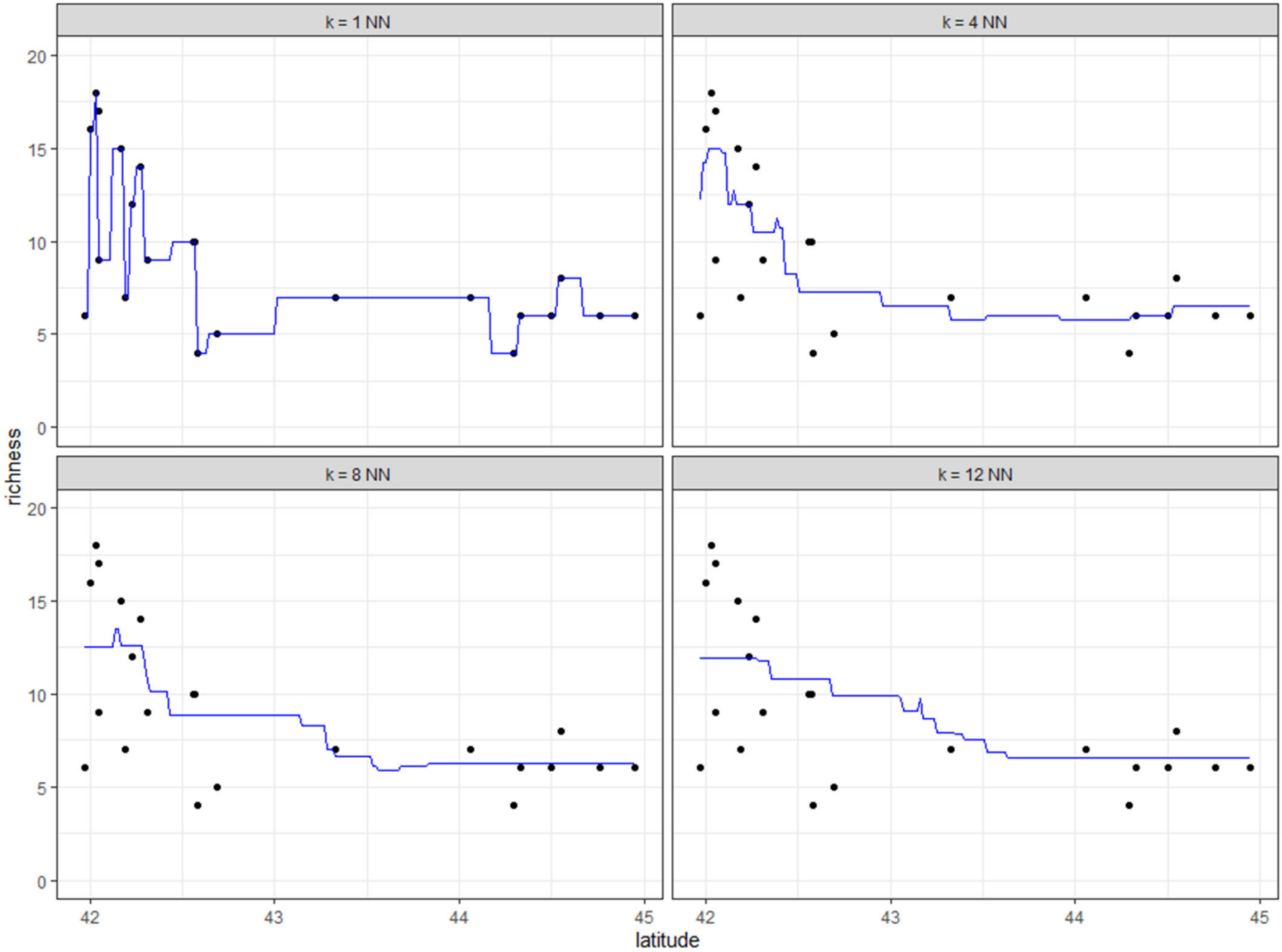
- Explore CV code for ants with smoothing spline models
- KNN models (k nearest neighbors)
- KNN code for ants

KNN

- k nearest neighbors
- e.g of a more “algorithmic” model algorithm
- no parameters to be trained
- one tuning parameter


KNN
k = 4
Predicted y = 10.5
Predicted y = 6
richness
latitude
x = 42.25
x = 44.4
20
15
10
5
0
42
43
44
45


KNN
k = 4
richness
latitude
20
15
10
5
0
42
43
44
45


k = 1 NN
k = 4 NN
k = 8 NN
k = 12 NN
richness
latitude
0
5
10
15
20
42
43
44
45

KNN

Algorithm

Set k = number of nearest neighbors
Input (x, y) = x, y data pairs
Input x_new = x value at which to predict y_new
Calculate d = distance of x_new to other x
Sort y data ascending by d; break ties randomly
Predict new y = mean of k nearest neighbors;
i.e. mean of first k values in y_sort

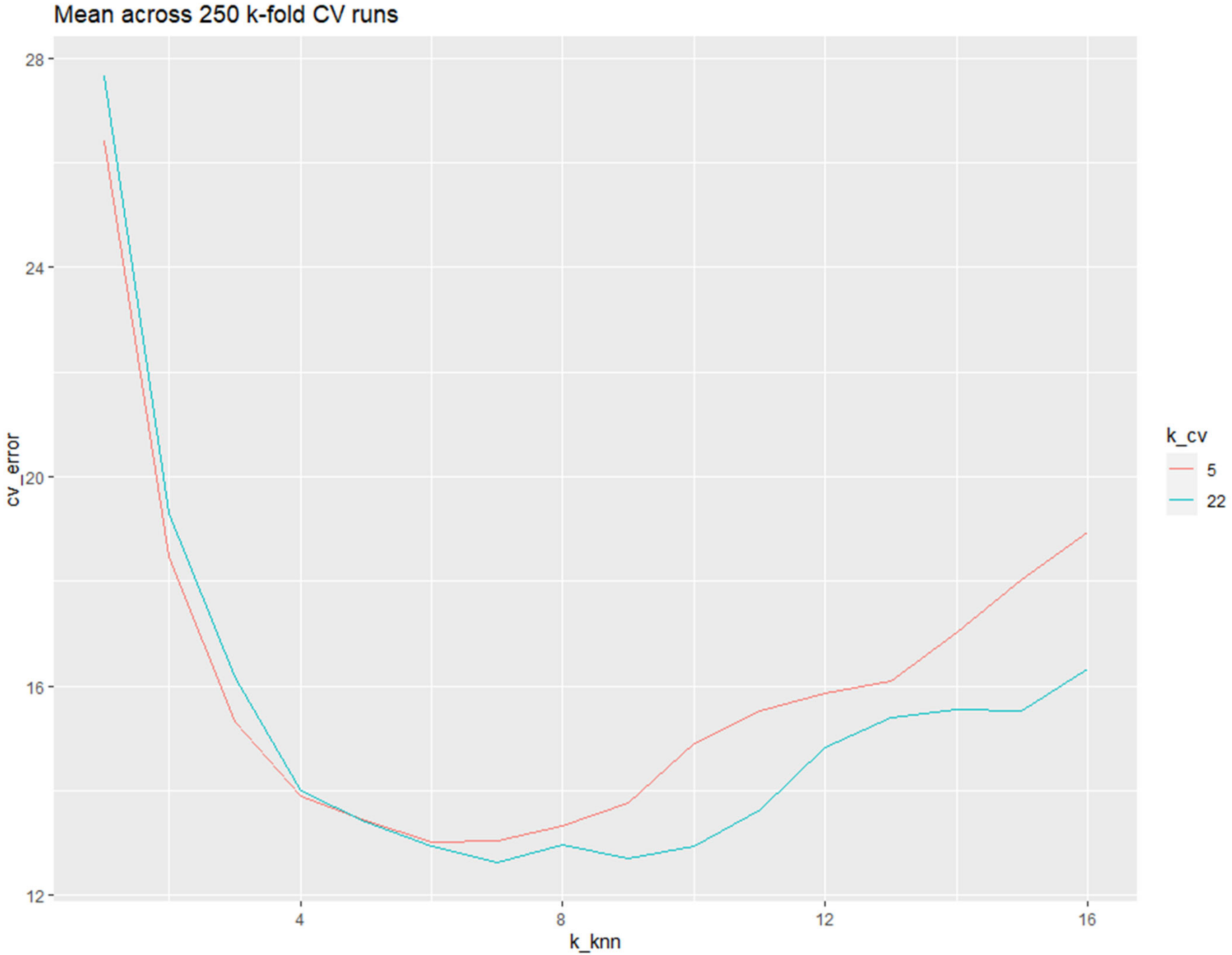
Code

- ants_knn.R
- k-fold CV for KNN models with different numbers of nearest neighbors

Pseudocode to R code

```r
#' K Nearest Neighbors (KNN) algorithm for 1 new value of x, translating our
#' pseudocode to R code.

# Set k = number of nearest neighbors
k <- 4

# Input (x, y) = x, y data pairs
x <- forest_ants$latitude
y <- forest_ants$richness

# Input x_new = x value at which to predict y_new
x_new <- 42.25

# Calculate d = distance of x_new to other x
d <- abs(x - x_new)

# Sort y data ascending by d; break ties randomly
y_sort <- y[order(d, sample(1:length(d)))]

# Predict new y = mean of k nearest y data
y_pred <- mean(y_sort[1:k])
```


30
25
20
15
cv_error
4
8
12
16
k_knn
k_cv
5
10
22


Mean across 250 k-fold CV runs
cv_error
28
24
20
16
12
4
8
12
16
k_knn
k_cv
5
22

Model	LOOCV	5-fold CV
KNN 6	12.95	13.01
KNN 7	12.63	13.03
Smoothing spline 3	12.52	12.77

... but single run LOOCV identifies KNN 7 with LOOCV=12.48
i.e. be aware of randomness in algorithms