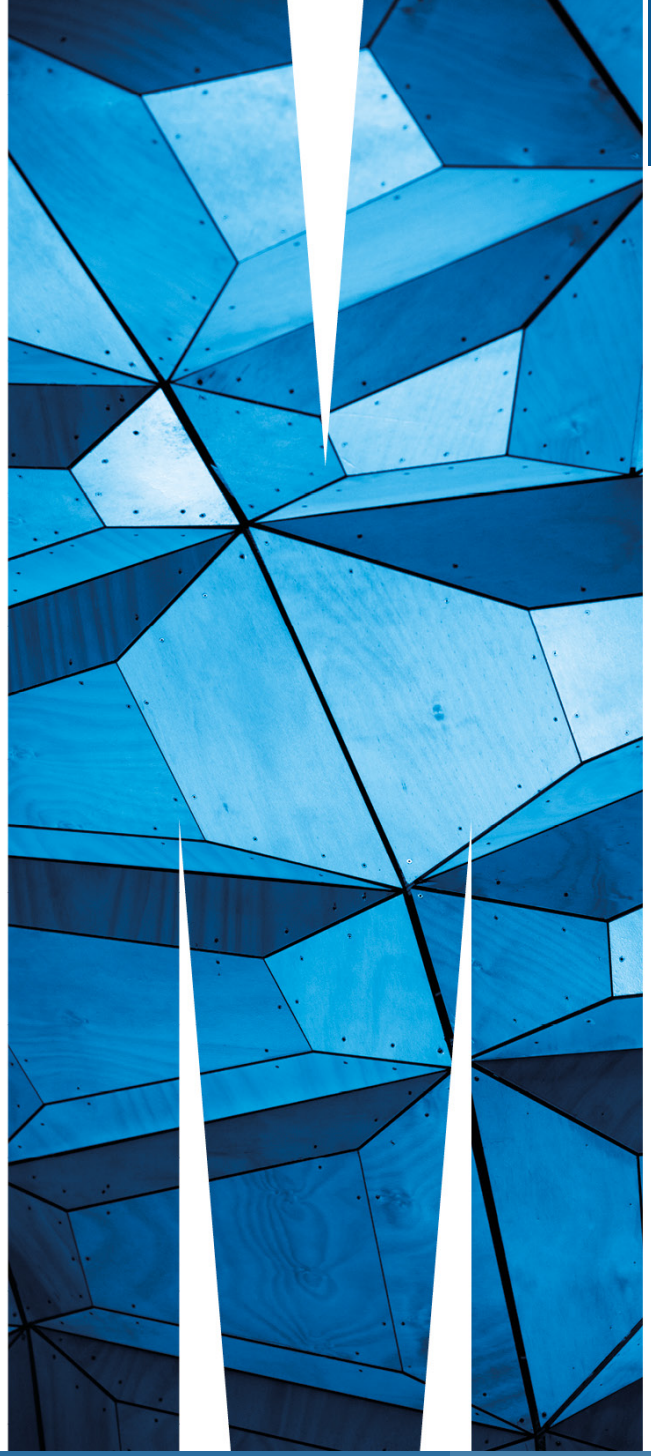
MONASH University

MONASH BUSINESS SCHOOL

ETC3250: Classification with Trees & Forests

Week 9, class 1

Professor Di Cook, Econometrics and Business Statistics



GROUP OF EIGHT AUSTRALIA

AACSB ACCREDITED EFMD EQUIS ACCREDITED ASSOCIATION OF MBAS ACCREDITED

Decision trees

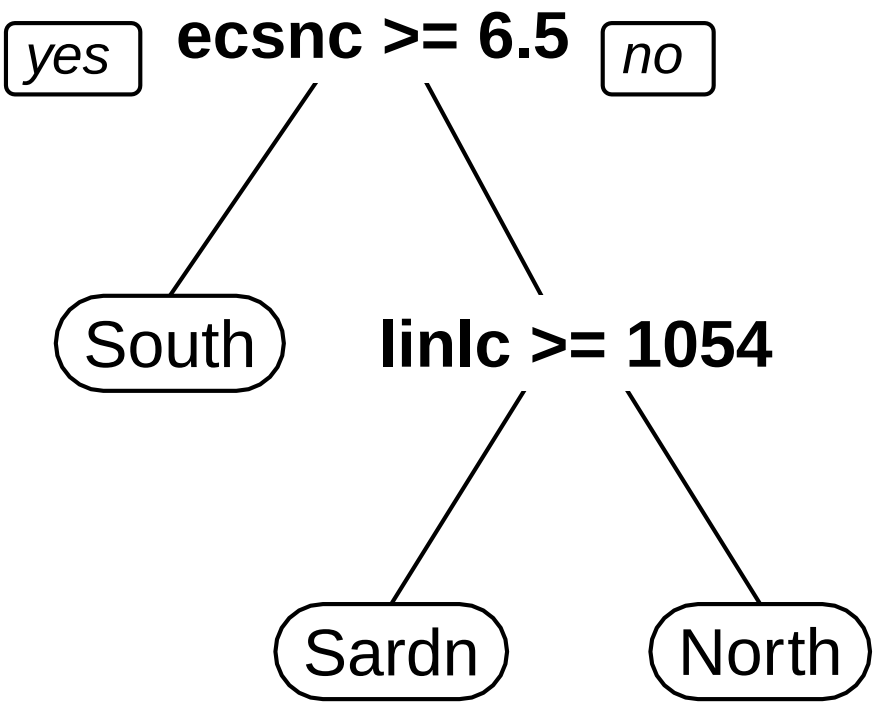
- Recursive binary splitting
- Compute all possible splits $(n-1)$ on every variable (p)
- Choose the best split of the data, the one that separates it into two groups which are the most "pure"
- Continue to operate on each of the subsets until a stopping criteria is satisfied (e.g. all cases in the subset are of one class, there are less than m cases in a subset, ...)

Example: olive oils

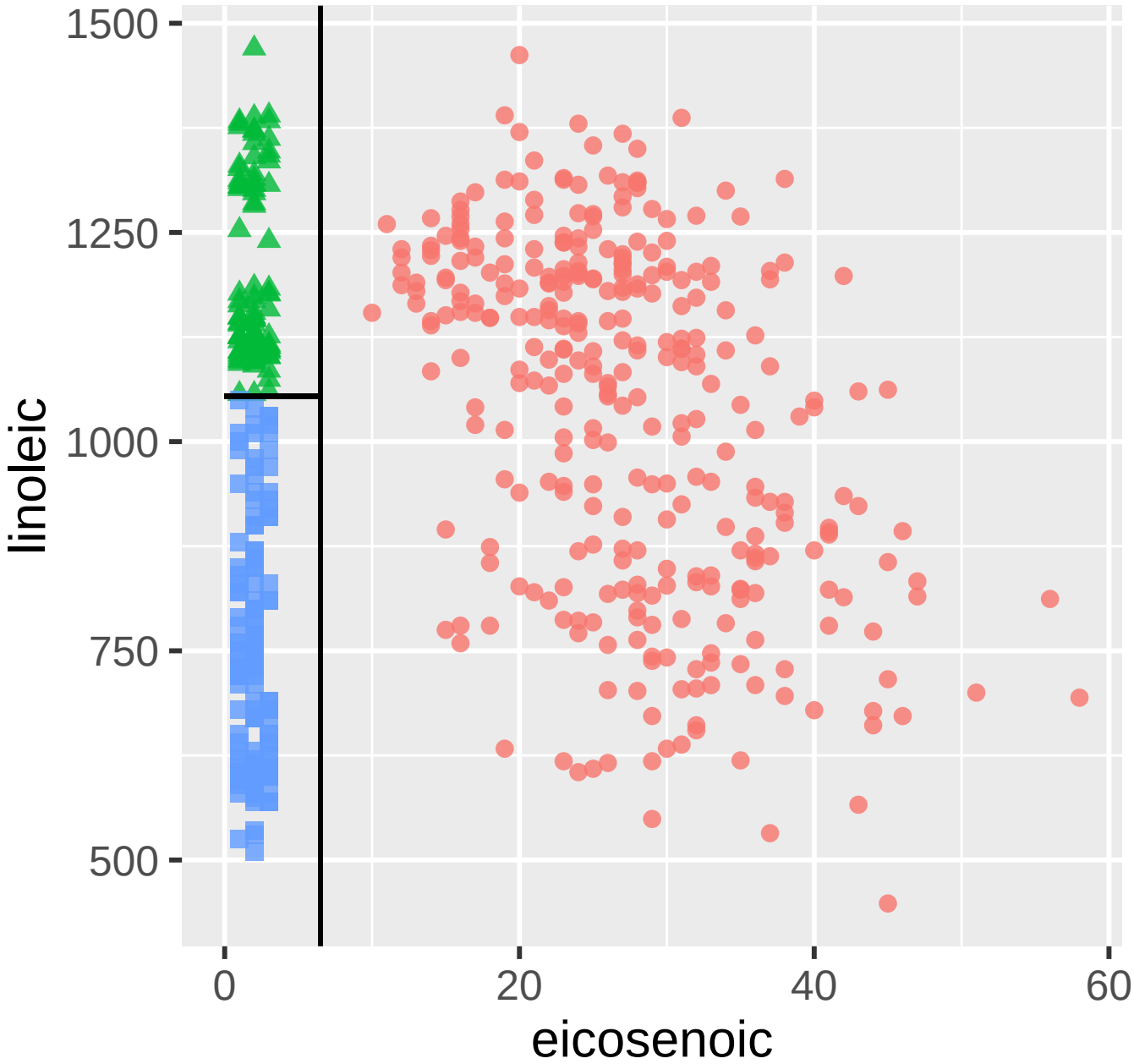
```
## n= 572
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 572 250 South (0.56 0.17 0.26)
##   2) eicosenoic>=6.5 323   0 South (1.00 0.00 0.00) *
##   3) eicosenoic< 6.5 249  98 North (0.00 0.39 0.61)
##     6) linoleic>=1.1e+03 98   0 Sardinia (0.00 1.00 0.00) *
##     7) linoleic< 1.1e+03 151   0 North (0.00 0.00 1.00) *

##
##              South Sardinia North
##   South        323        0     0
##   Sardinia       0       98     0
##   North          0        0   151
```

Example: olive oils



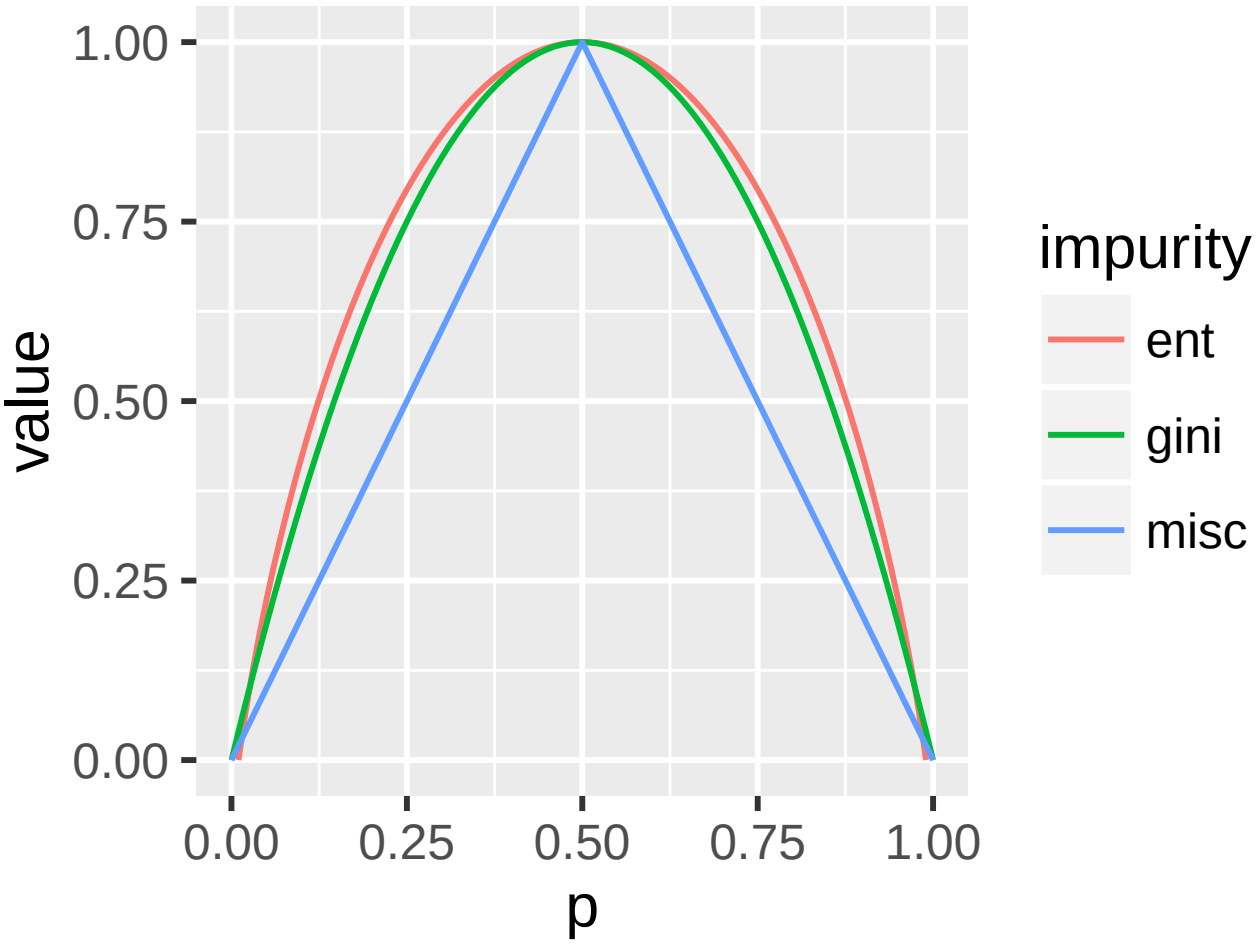
Example: olive oils



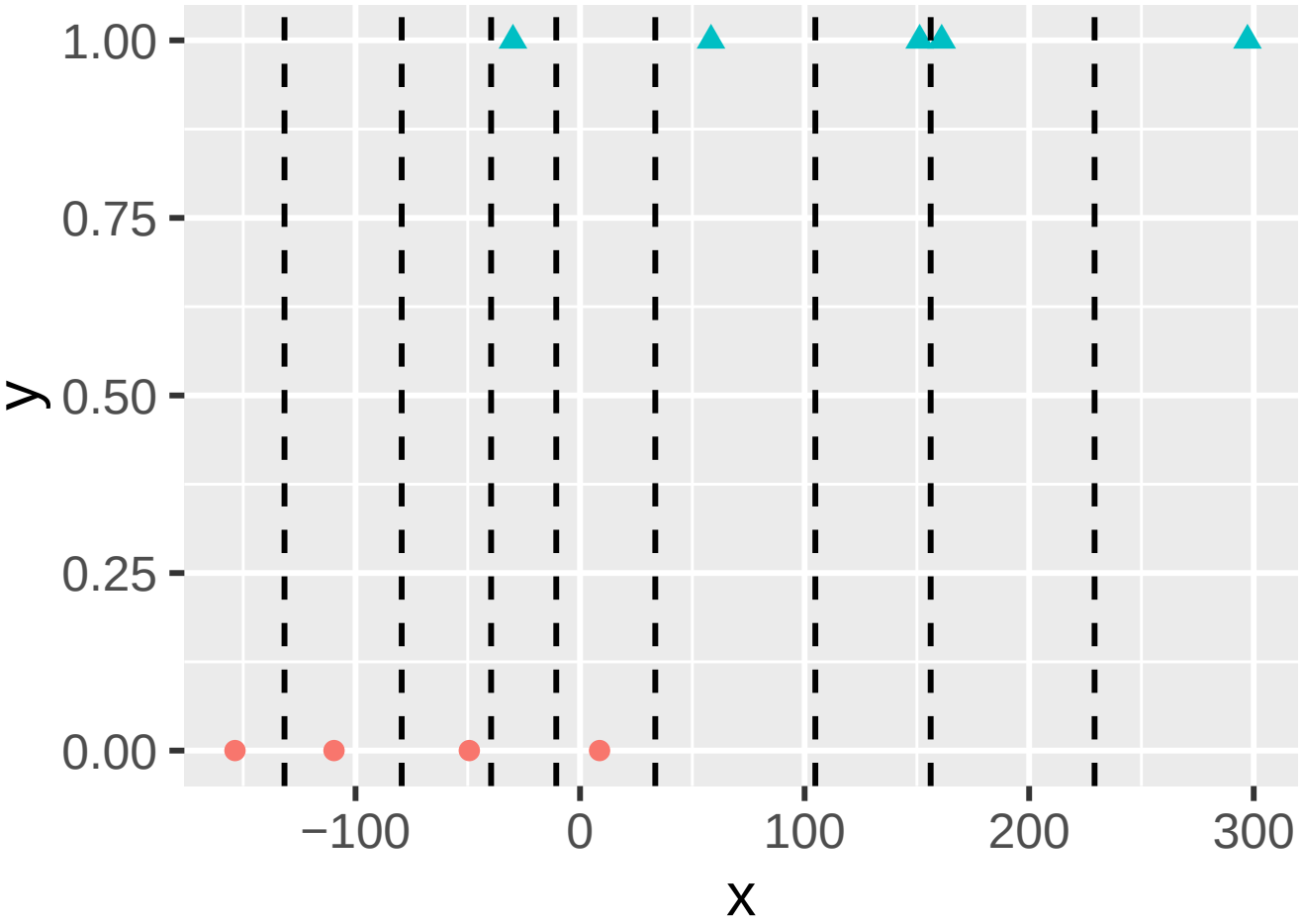
Measuring the quality of splits

- Explanation for two classes (0,1), and p=proportion in class 0
- Entropy: $-p(log_e p) - (1-p)log_e(1-p)$
- Gini: $2p(1-p)$
- Misclassification: $1 - 2|p - 0.5|$

What these look like



Choosing the best split

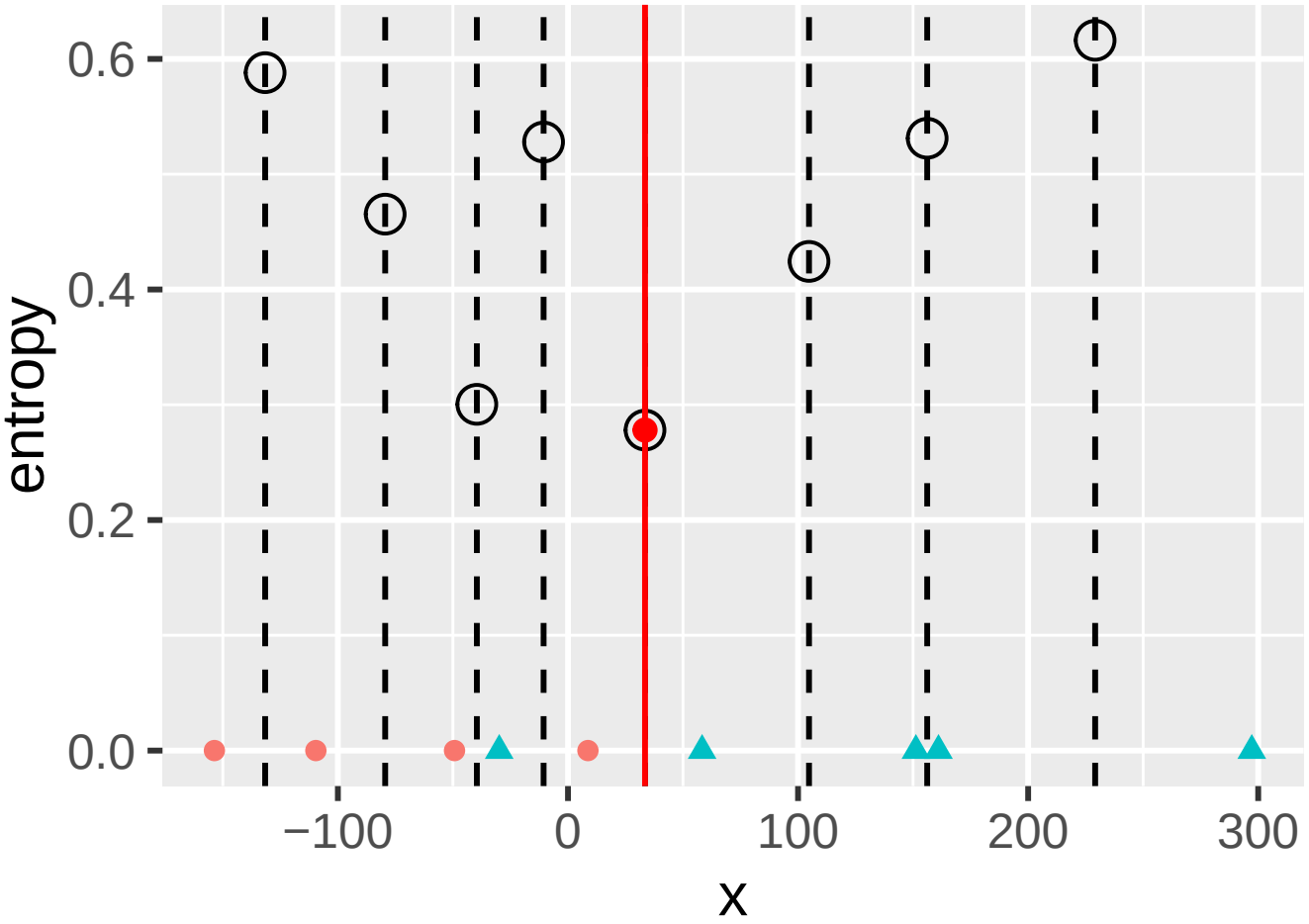


Choosing the best split

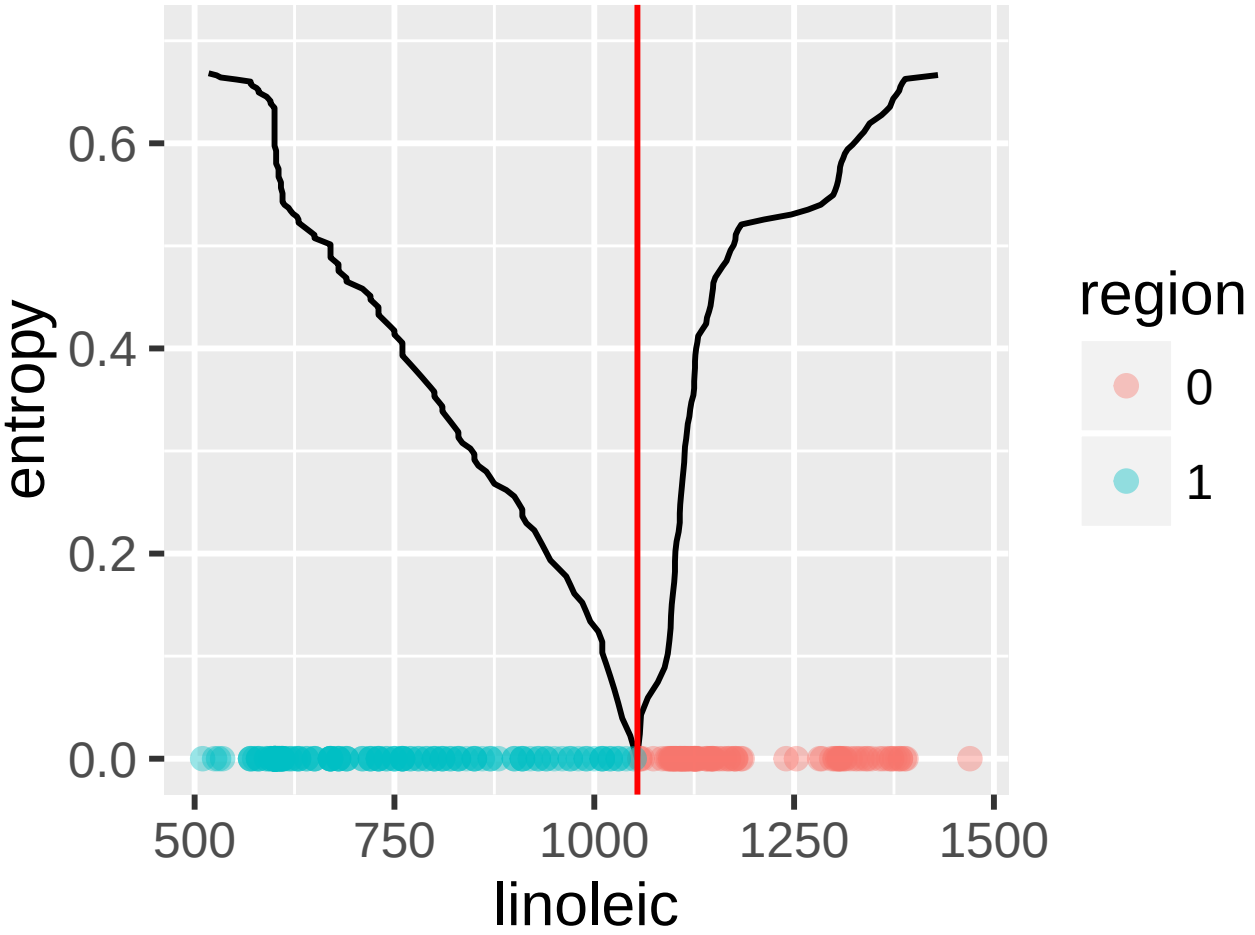
- Calculate the impurity for each subset, and combine
- p^L impurity $_L + p^R$ impurity$_R$

```
##      x  ent
## 1 -132 0.59
## 2  -79 0.47
## 3  -40 0.30
## 4  -11 0.53
## 5   33 0.28
## 6  105 0.42
## 7  156 0.53
## 8  229 0.62
```

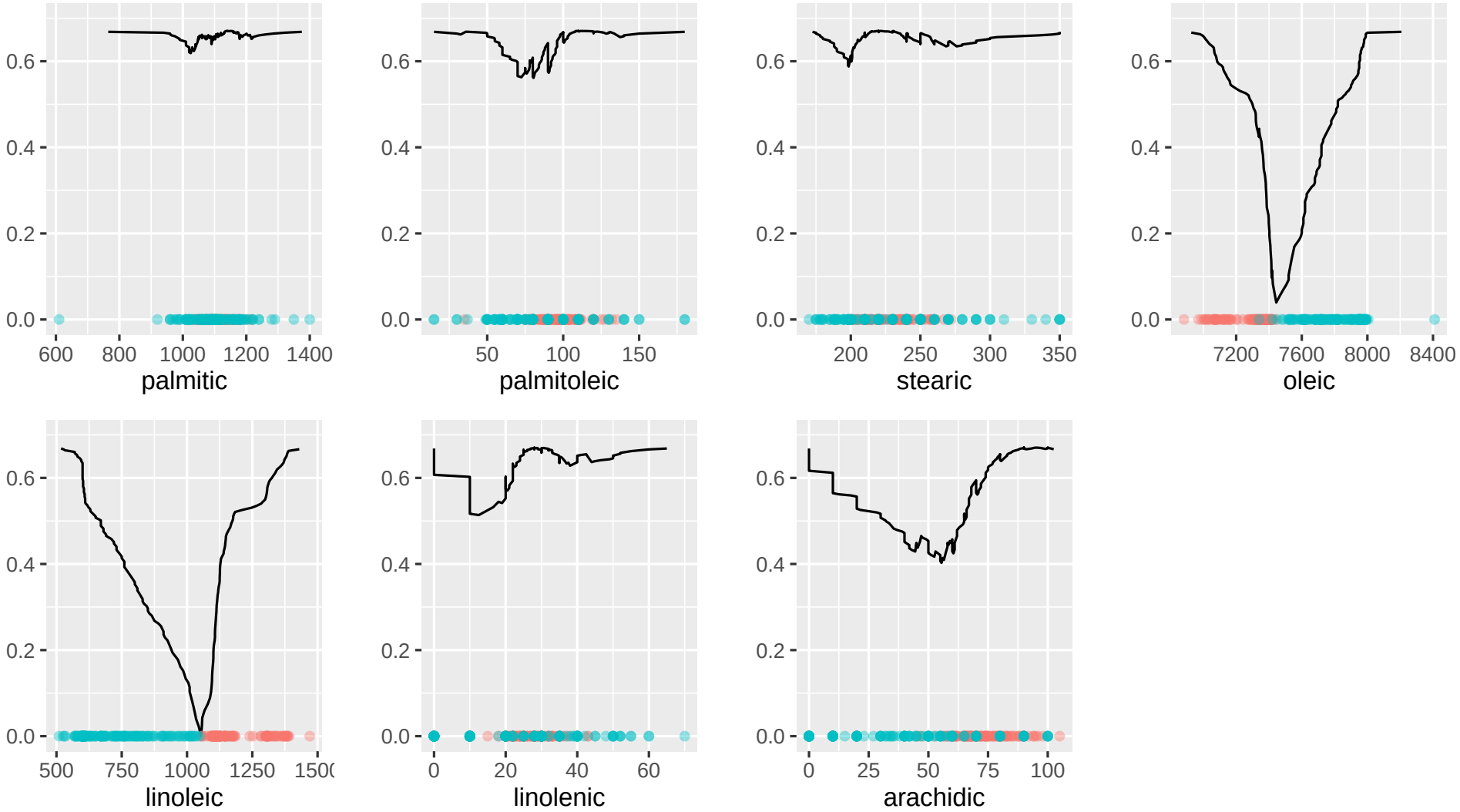
Choosing the best split



Example: olive oils



Example: olive oils



Stopping rules

- *minsplit:* minimum number of observations allowed in order to consider splitting
- *minbucket:* minimum number of observations in a terminal node
- *cp (0.01):* complexity parameter. The decrease in impurity cannot be less than this.

Overfitting

- It is possible to force a tree to fit the training sample very closely, by tweaking these stopping rules.
- This could lead to overfitting, really small error with the training data and much higher error with validation data, and hence test data.
- Tuning the algorithm control parameters with the validation set is really important.

Overfitting example

```
> df.rp <- rpart(y~., data=df,
+                control=rpart.control(minsplit=30, cp=0.01))
```

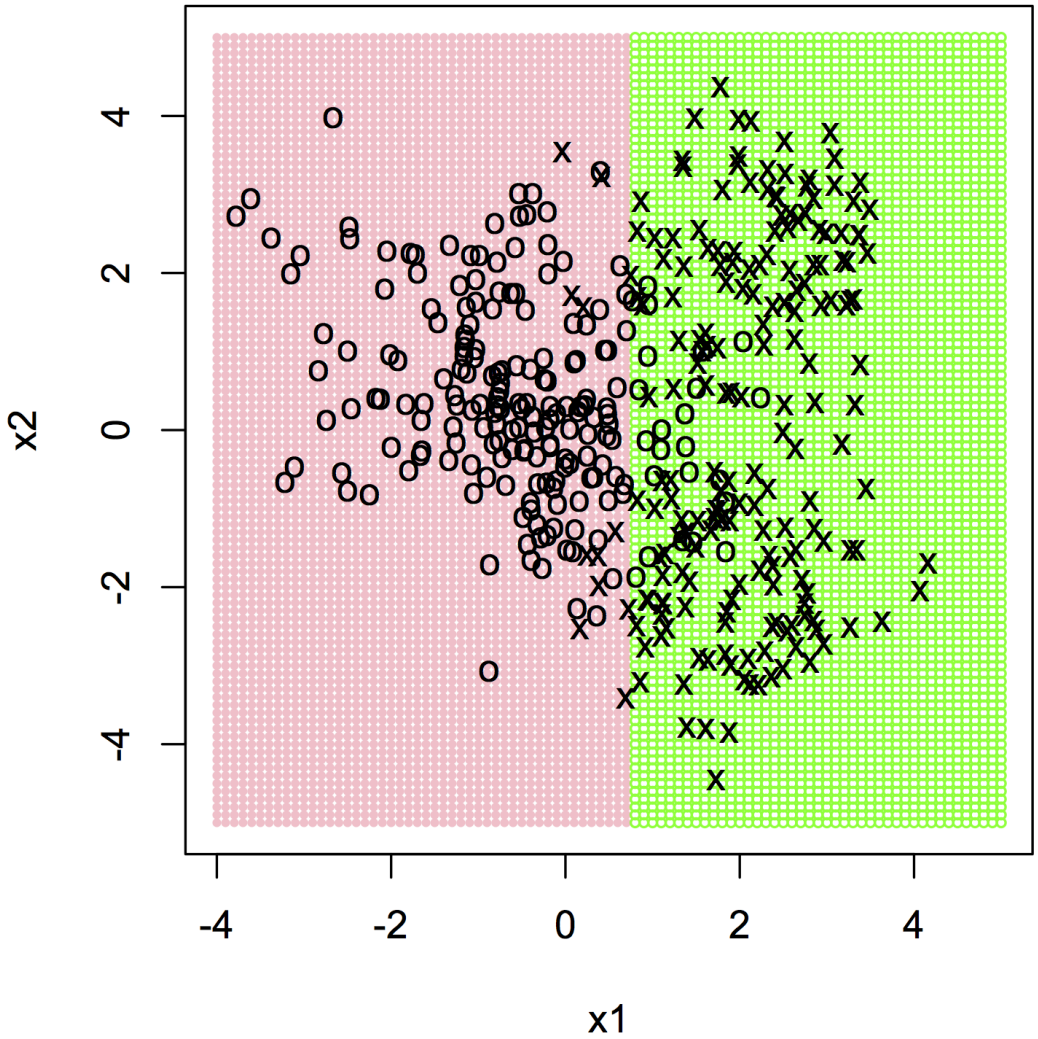


Figure 1: figures/overfit1.pdf

Overfitting example

```
> df.rp <- rpart(y~., data=df,
+                control=rpart.control(minsplit=30, cp=0.01))
```

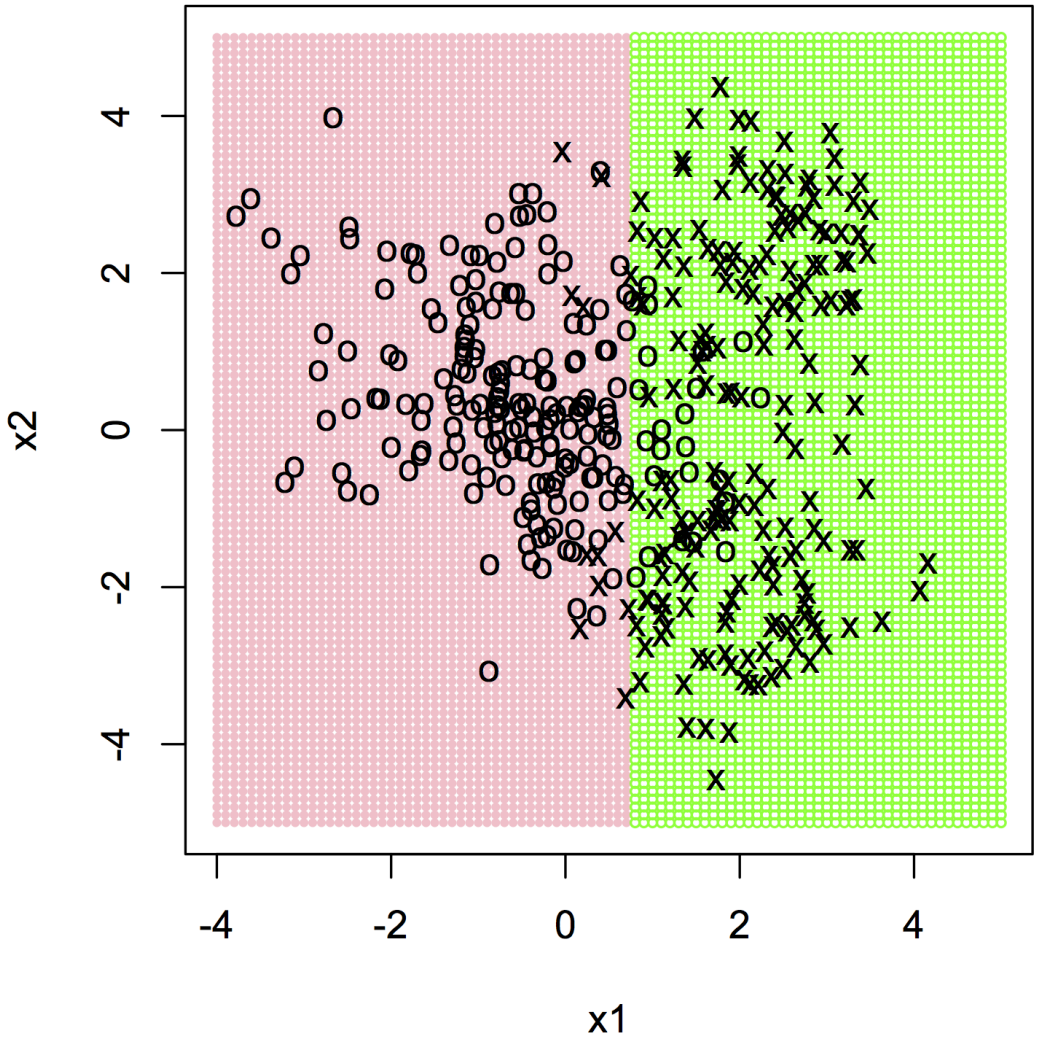


Figure 2: figures/overfit2.pdf

Overfitting example

```
> df.rp <- rpart(y~., data=df,
+                control=rpart.control(minsplit=10, cp=0.01))
```

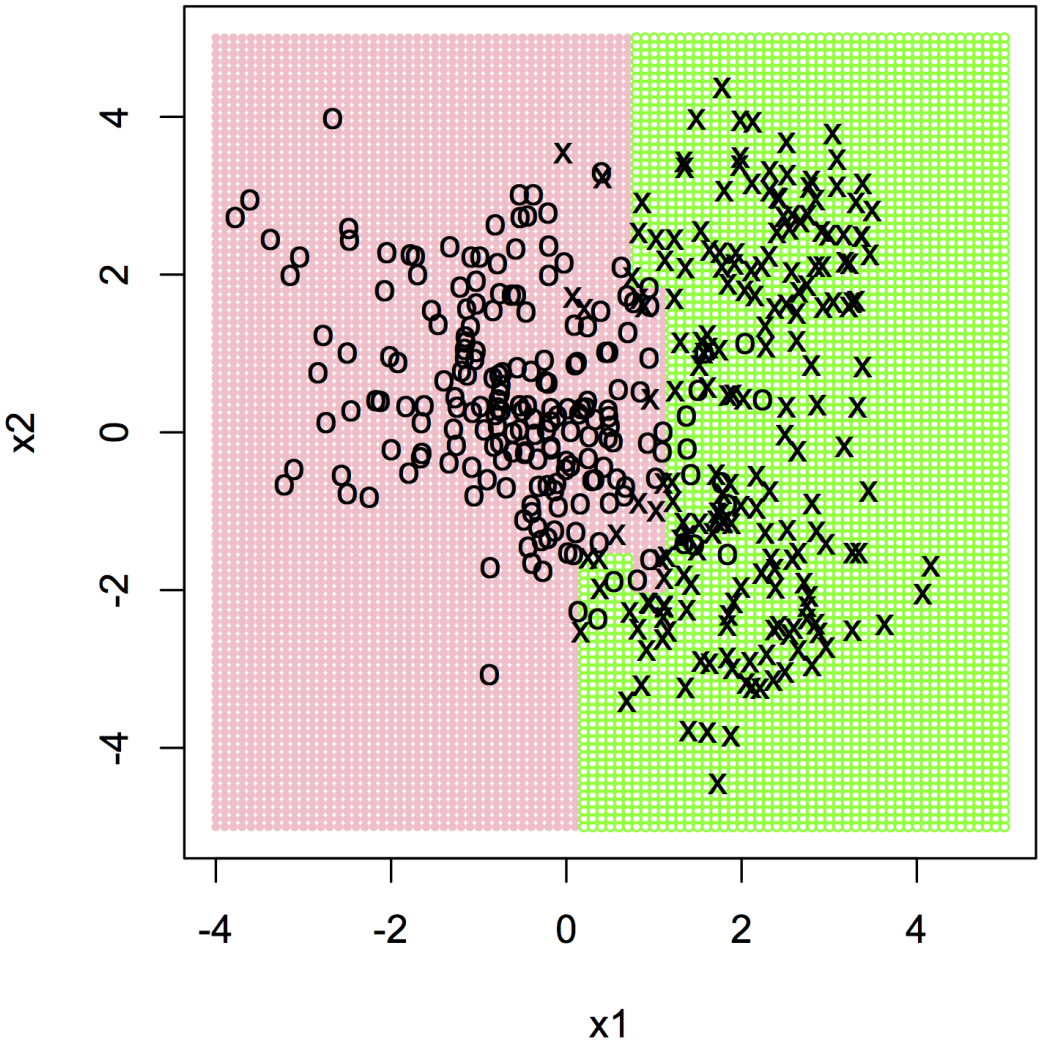


Figure 3: figures/overfit3.pdf

Overfitting example

```
> df.rp <- rpart(y~., data=df,
+                control=rpart.control(minsplit=2, cp=0.01))
```

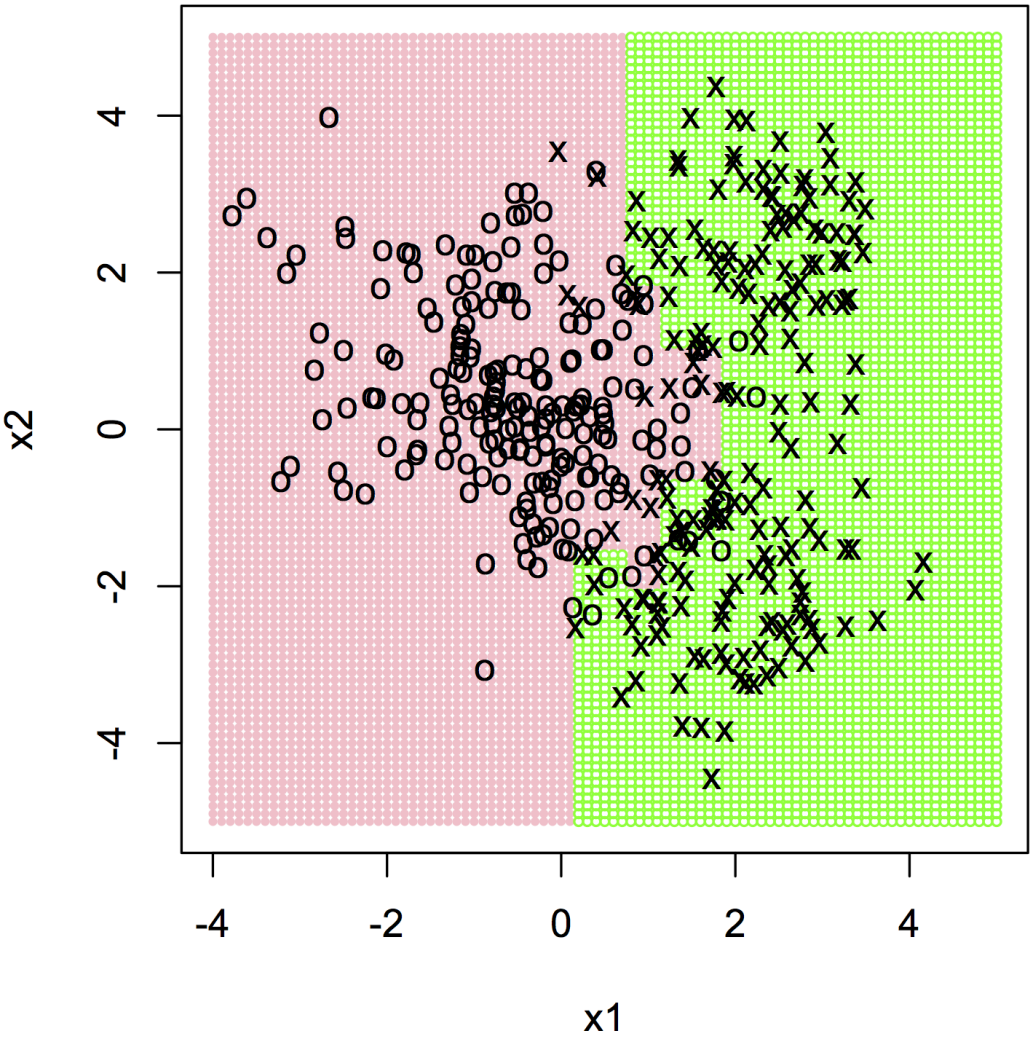


Figure 4: figures/overfit4.pdf

Overfitting example

```
> df.rp <- rpart(y~., data=df,
+                control=rpart.control(minsplit=20, cp=0.000001))
```

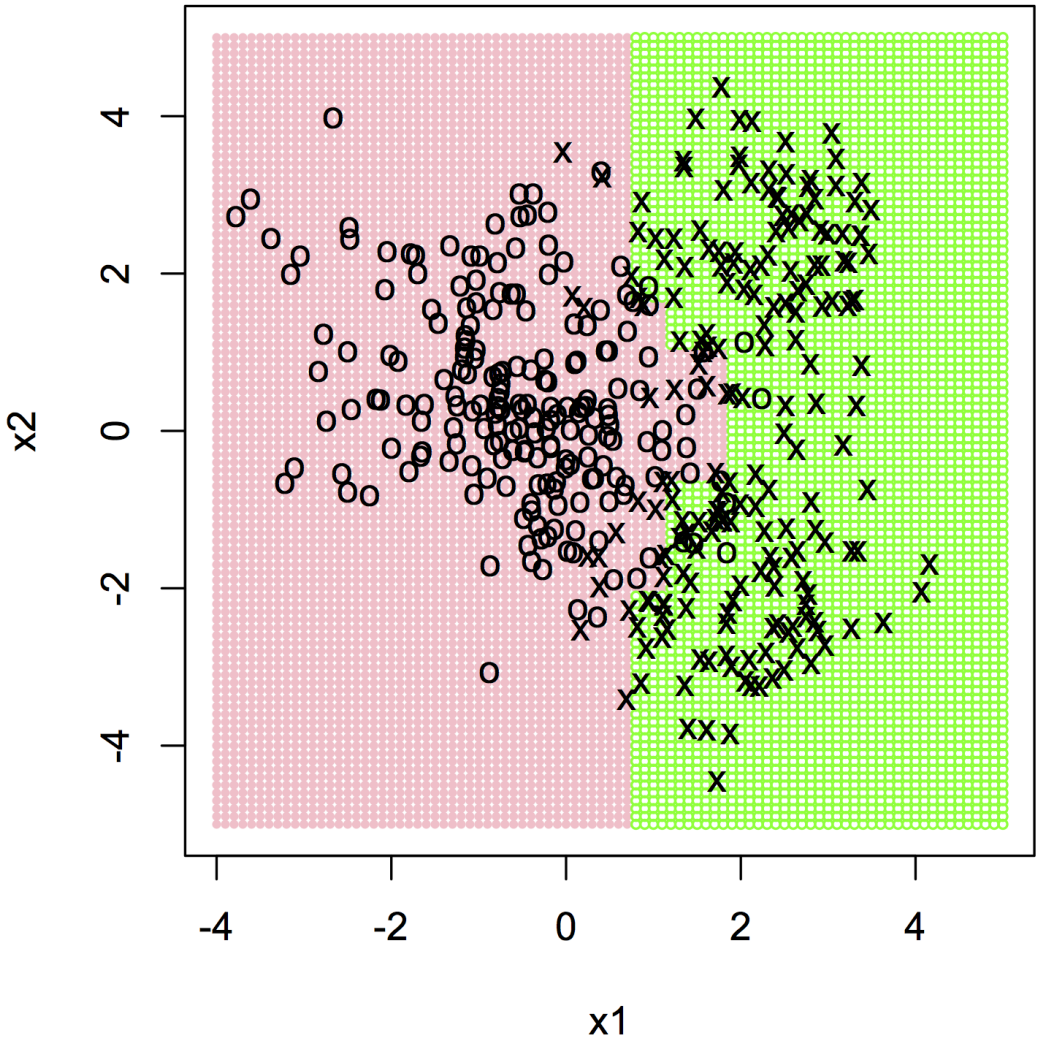


Figure 5: figures/overfit5.pdf

Overfitting example

```
> df.rp <- rpart(y~., data=df,
+                control=rpart.control(minsplit=10, cp=0.01))
```

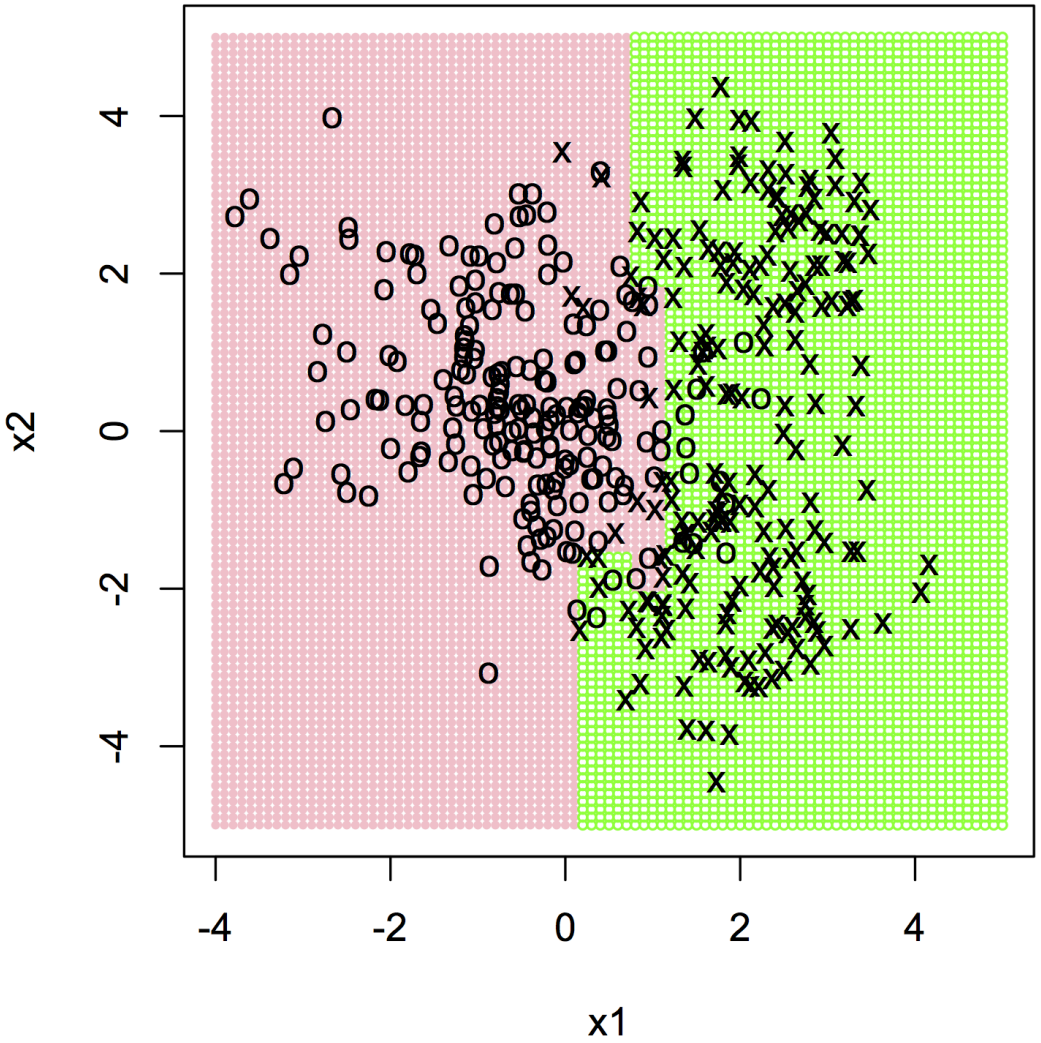


Figure 6: figures/overfit6.pdf

Overfitting example

```
> df.rp <- rpart(y~., data=df,
+                control=rpart.control(minsplit=10, cp=0.001))
```

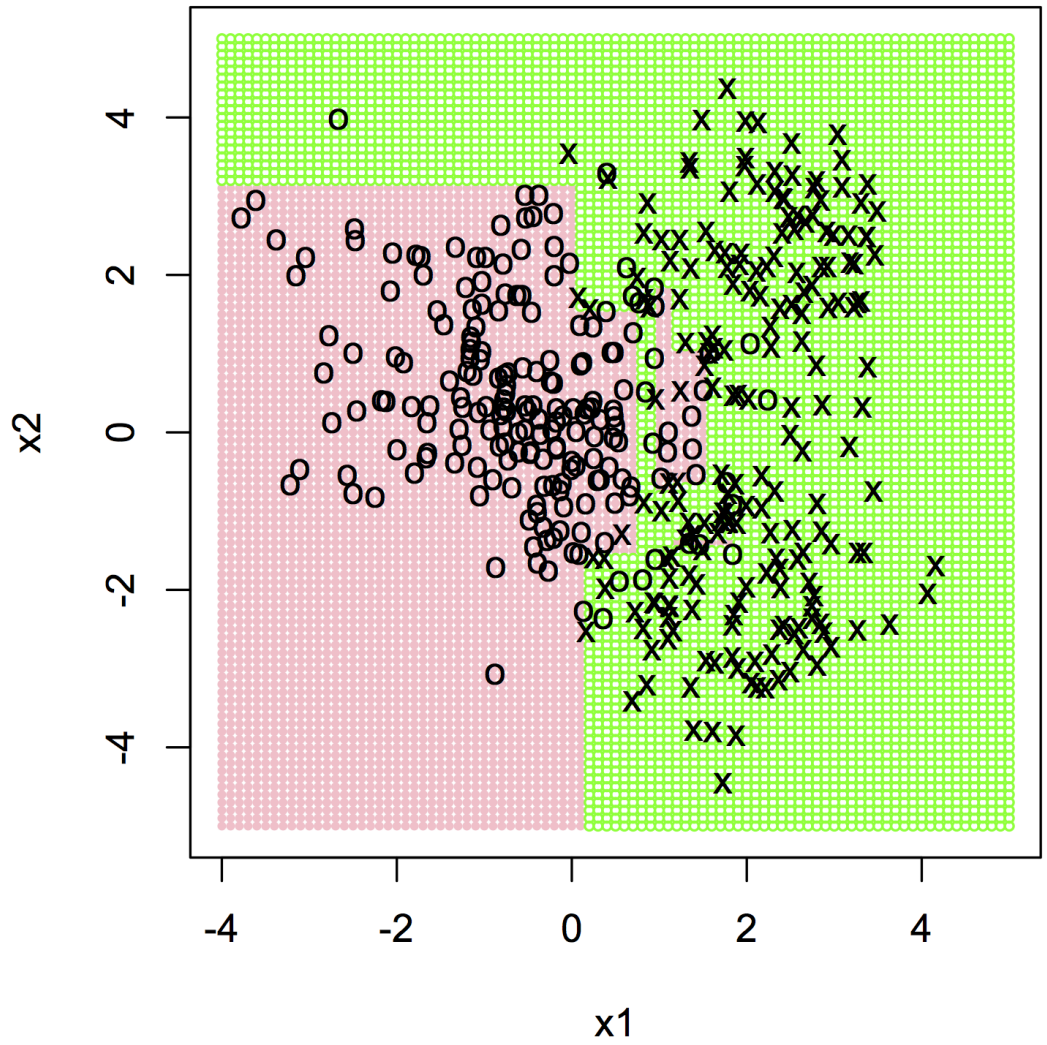


Figure 7: figures/overfit7.pdf

Overfitting example

```
> df.rp <- rpart(y~., data=df,
+                control=rpart.control(minsplit=2, cp=0.001))
```

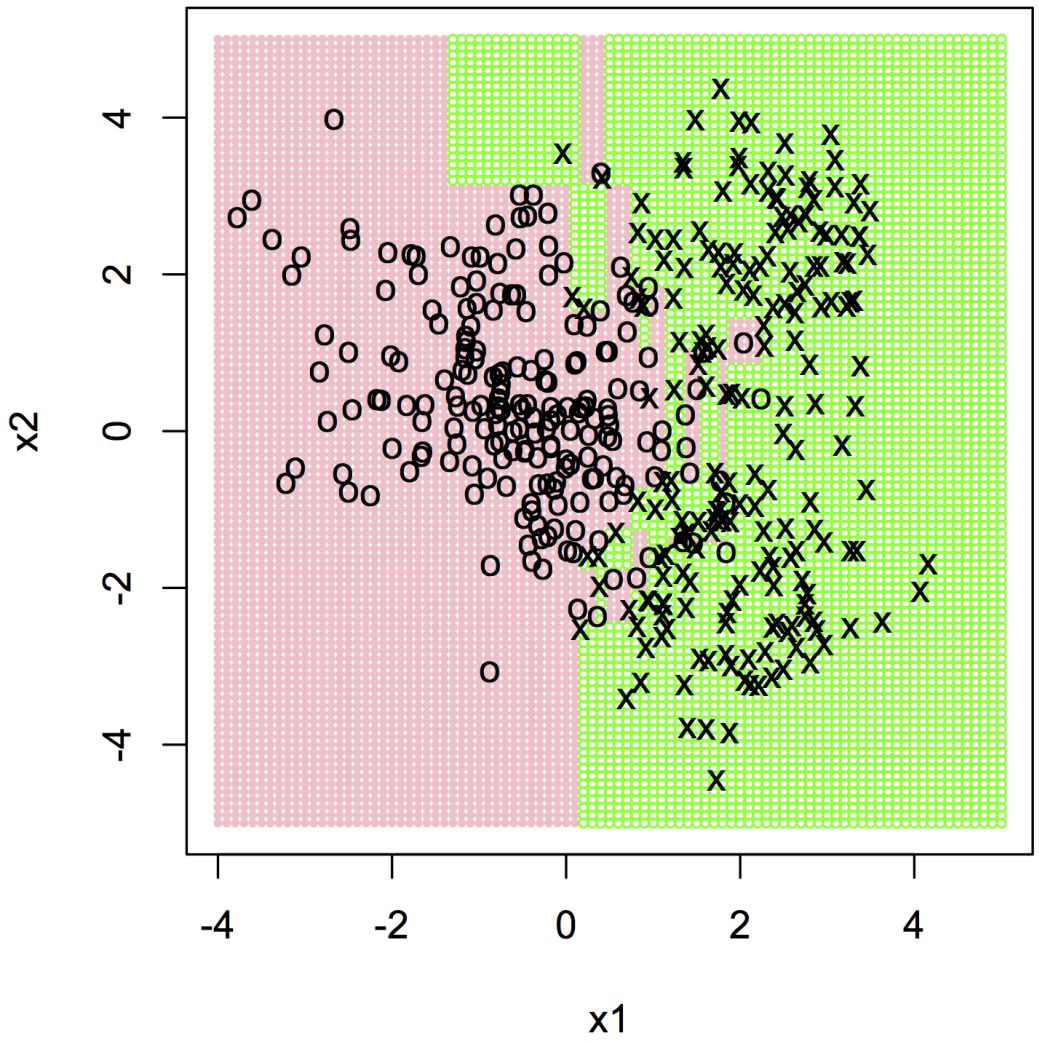


Figure 8: figures/overfit8.pdf

Overfitting example

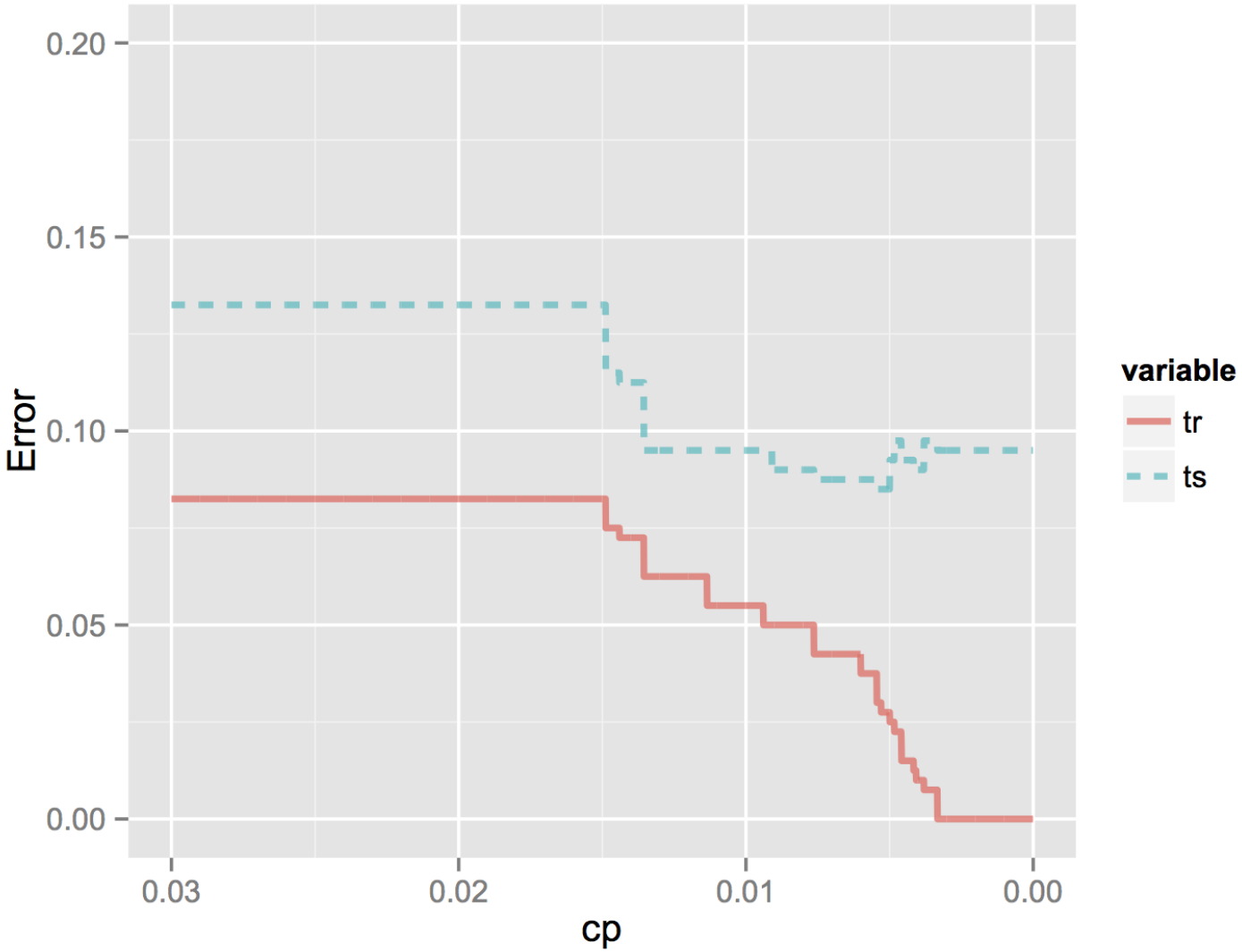


Figure 9: figures/overfit9.pdf

Overfitting example

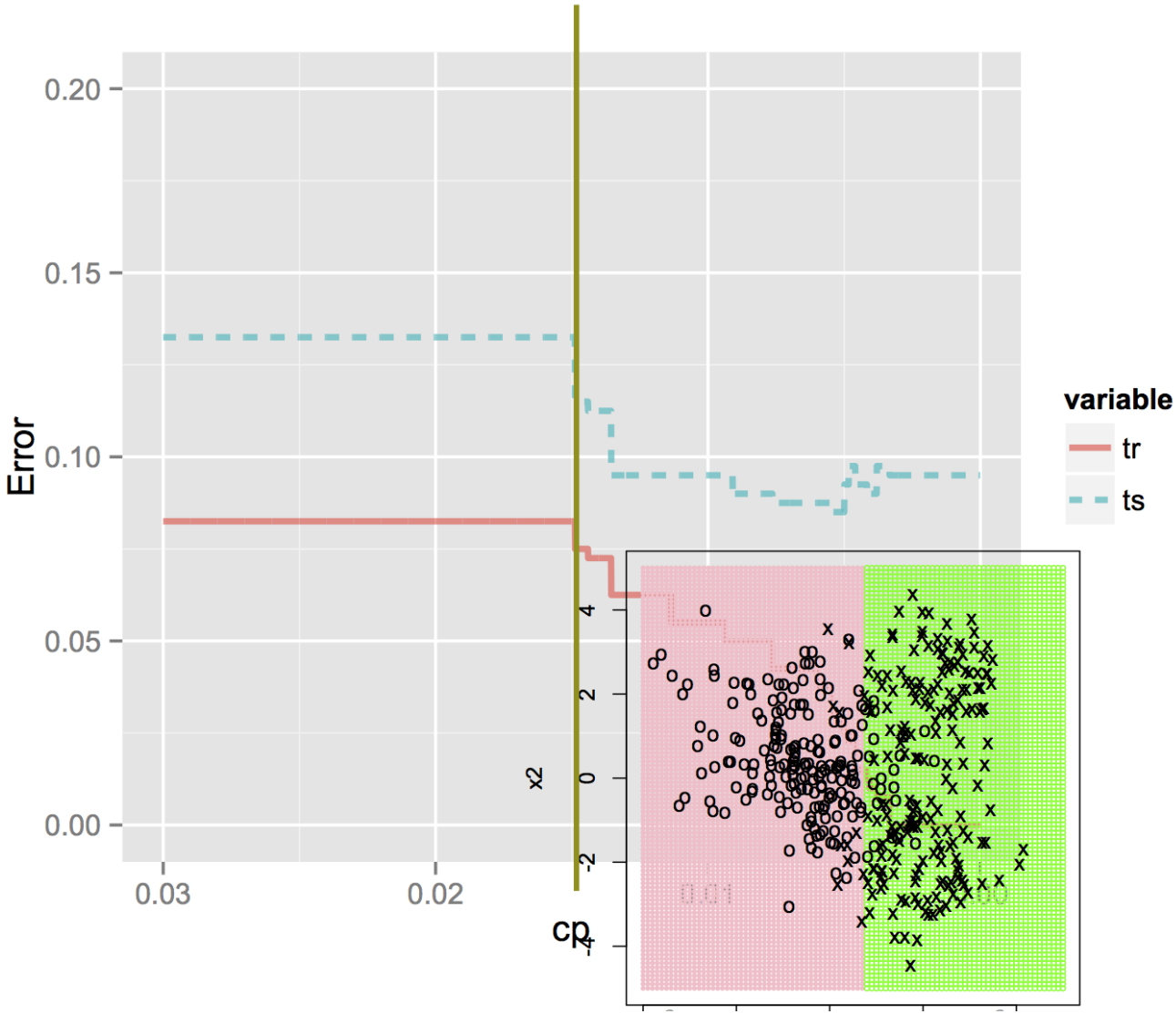


Figure 10: figures/overfit10.pdf

Overfitting example

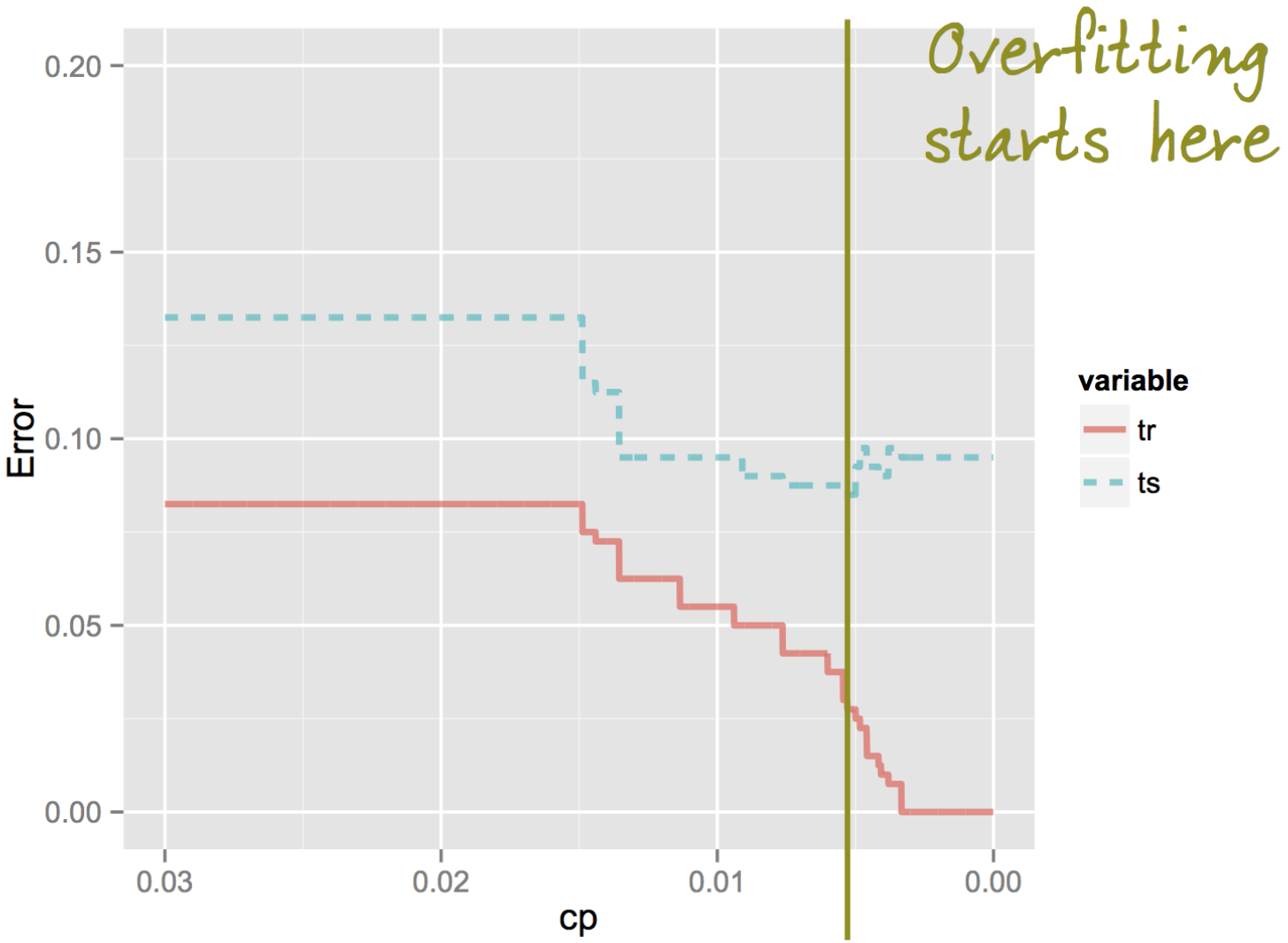


Figure 11: figures/overfit11.pdf

Overfitting example

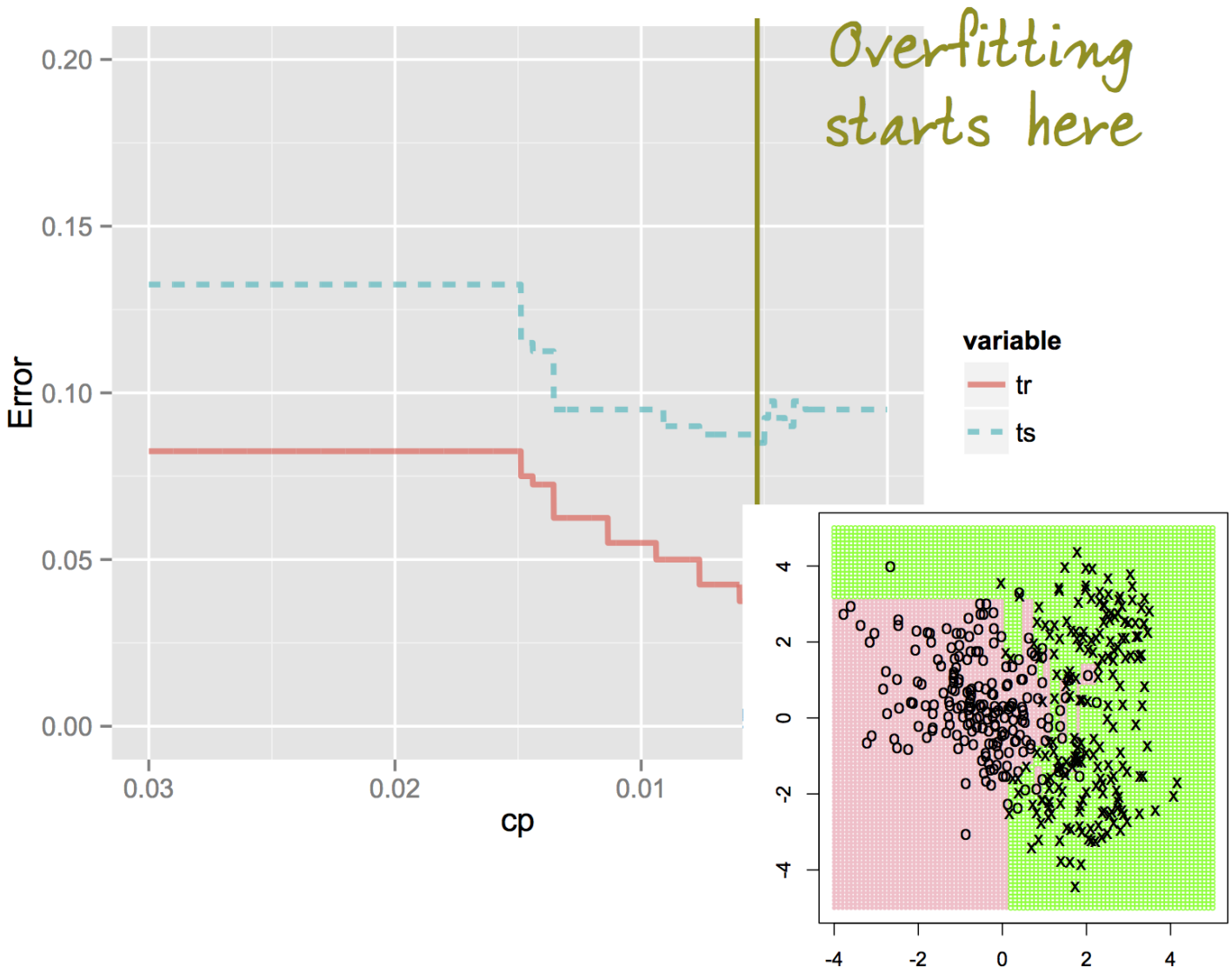


Figure 12: figures/overfit12.pdf

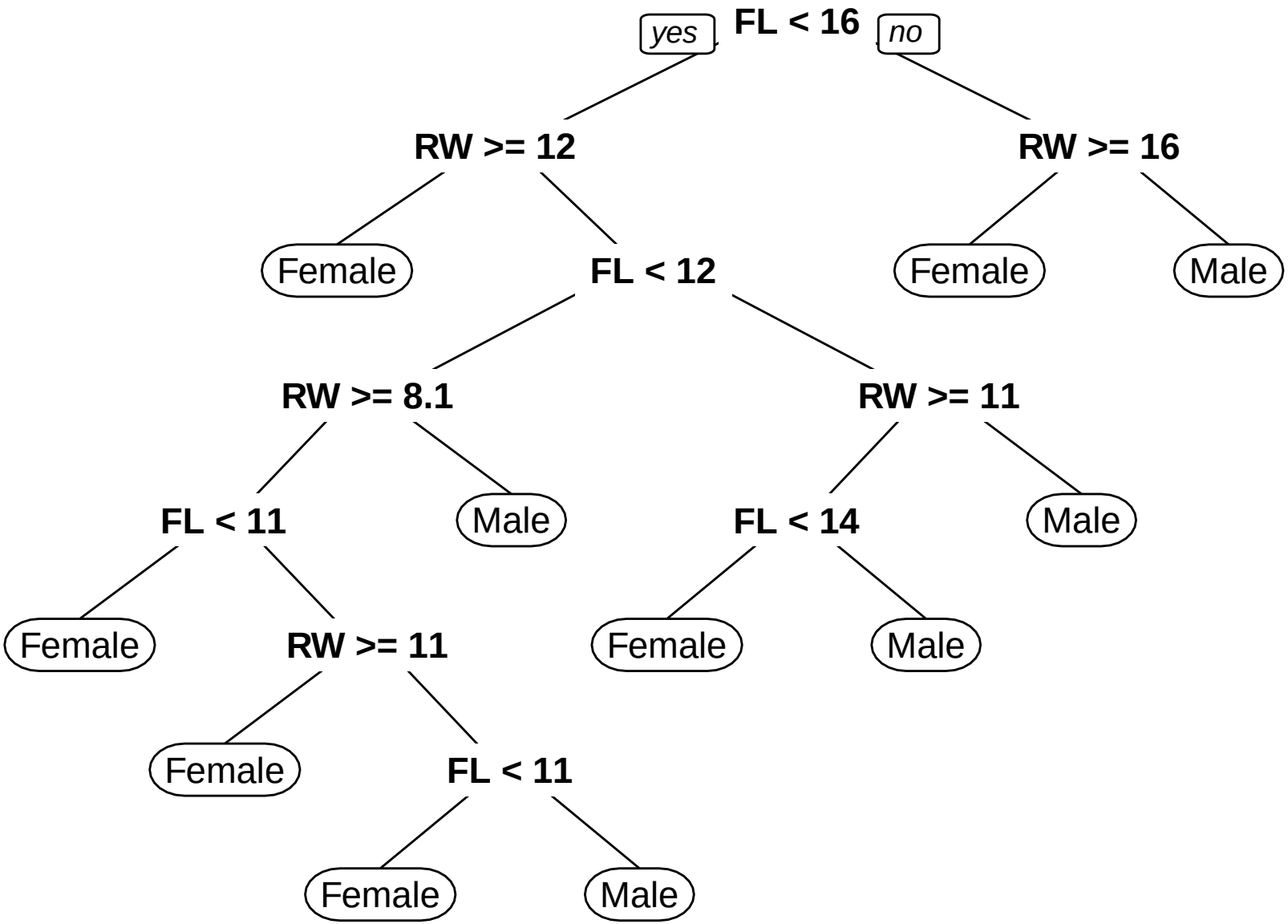
Pruning

- Grow an overly complex tree
- Prune back the weakest branches, using the cost complexity, or cross-validation to get the lowest validation error

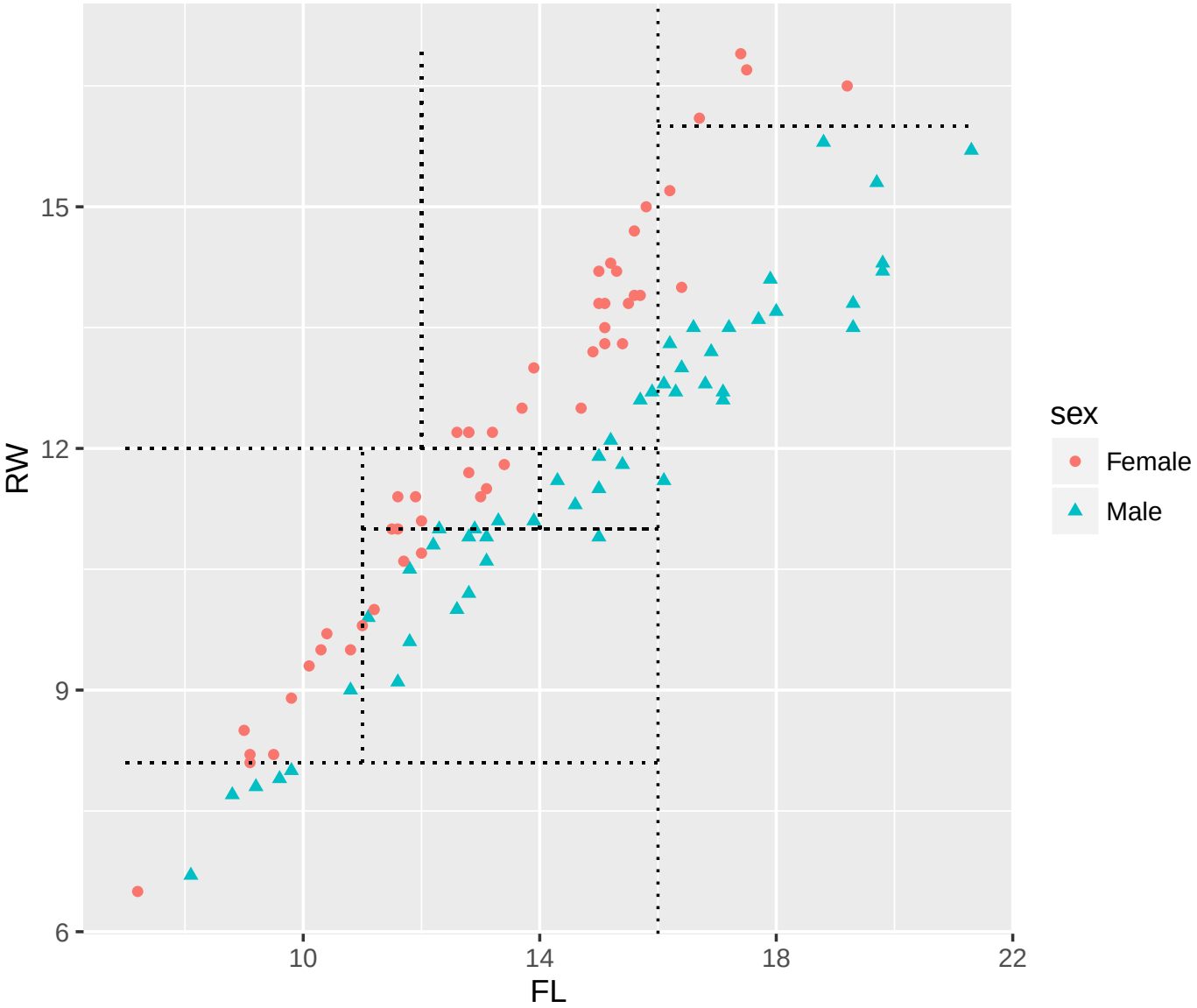
Example: crabs

```
## n= 100
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 100 50 Female (0.500 0.500)
##     2) FL< 16 72 28 Female (0.611 0.389)
##       4) RW>=12 22  1 Female (0.955 0.045) *
##       5) RW< 12 50 23 Male (0.460 0.540)
##        10) FL< 12 29 10 Female (0.655 0.345)
##          20) RW>=8.1 23  5 Female (0.783 0.217)
##            40) FL< 11 8  0 Female (1.000 0.000) *
##            41) FL>=11 15  5 Female (0.667 0.333)
##              82) RW>=11 7  0 Female (1.000 0.000) *
##              83) RW< 11 8  3 Male (0.375 0.625)
##               166) FL< 11 5  2 Female (0.600 0.400) *
##               167) FL>=11 3  0 Male (0.000 1.000) *
```

Example: crabs

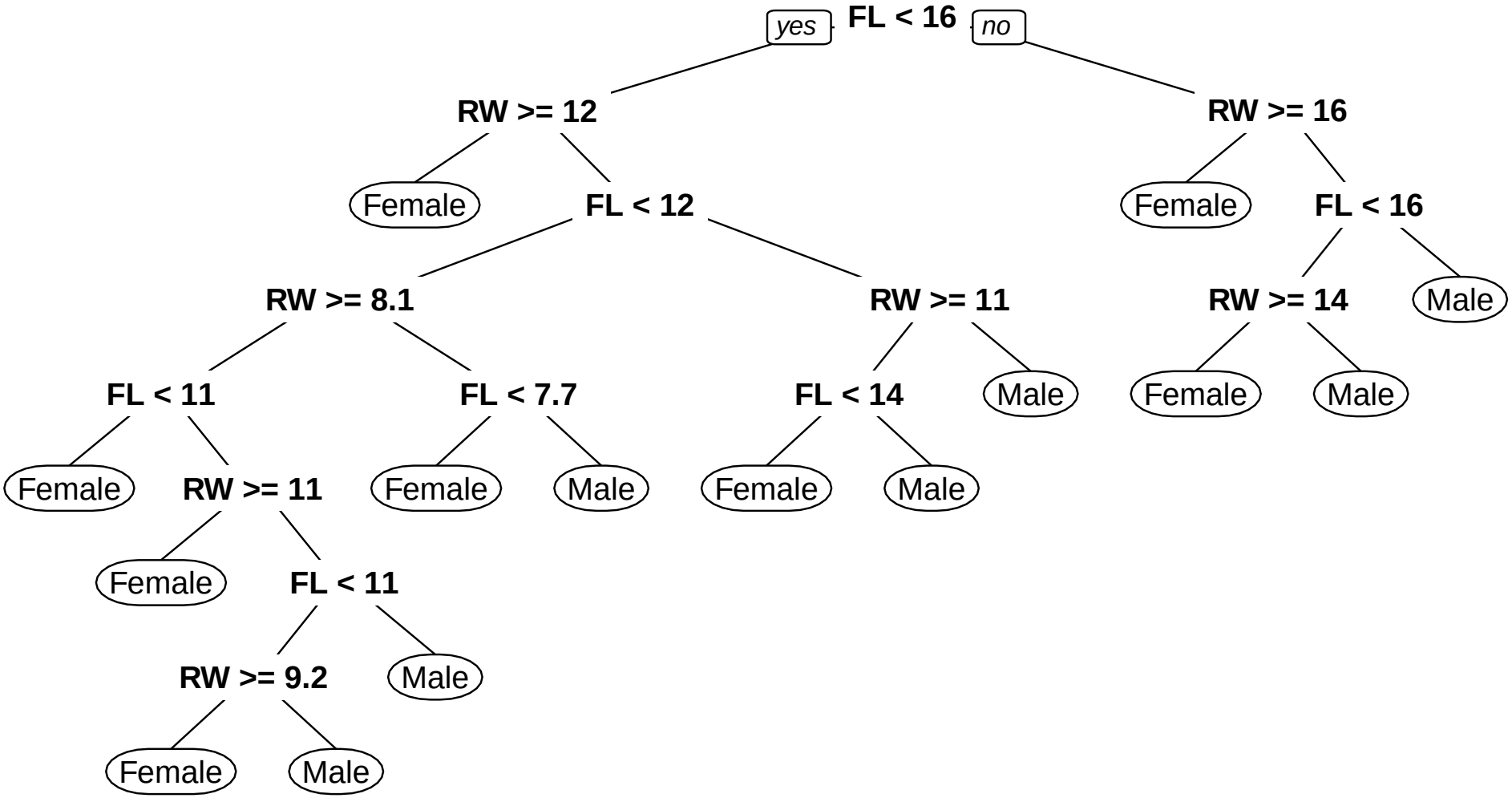


Example: crabs



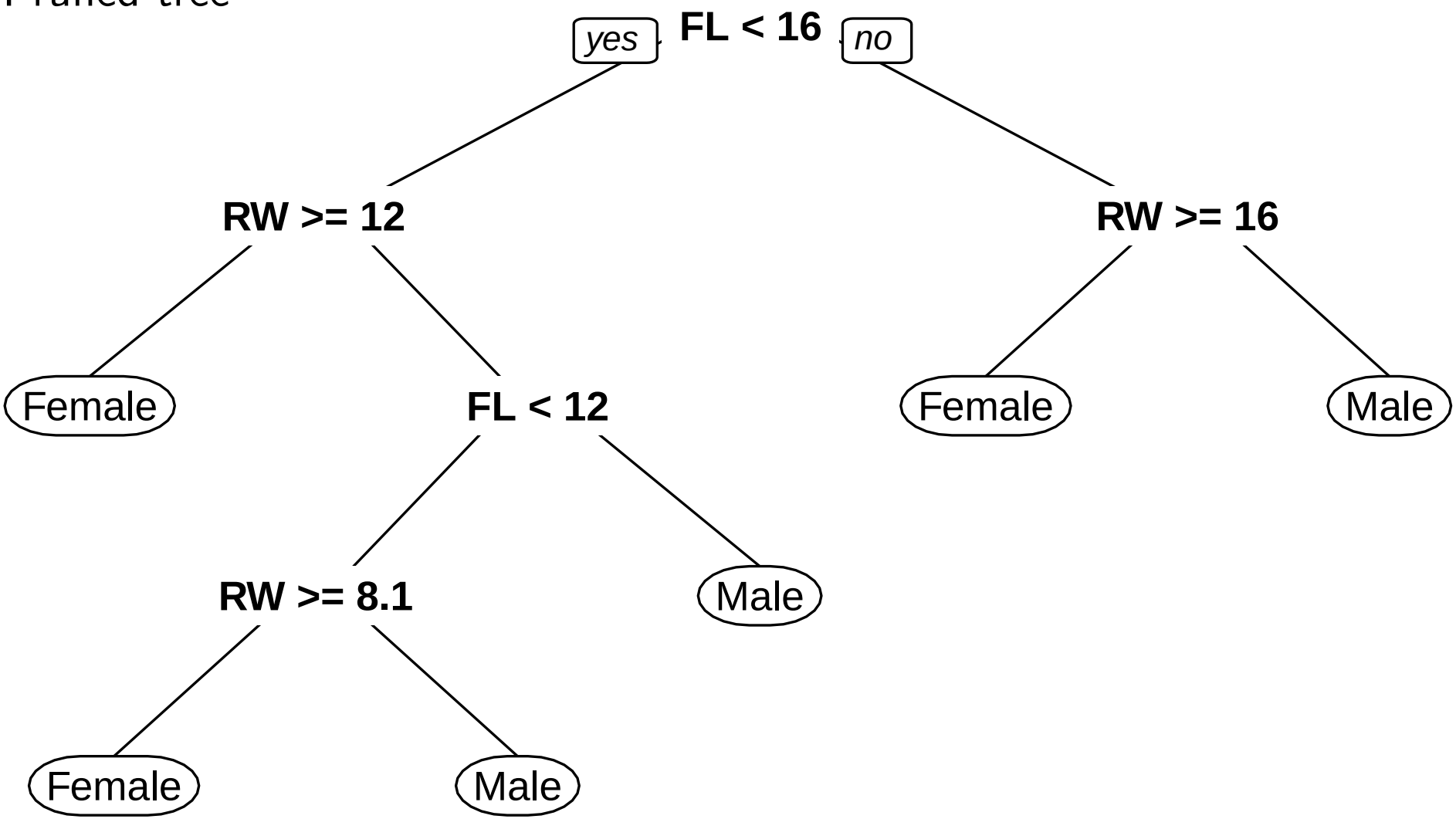
Example: crabs

Complex tree



Example: crabs

Pruned tree



Advantages and disadvantages

- The decision rules provided by trees are very easy to explain, and follow. A simple classification model.
- Trees can handle a mix of predictor types, categorical, quantitative, ...
- Trees efficiently operate when there are missing values in the predictors.
-
- Algorithm is greedy, a better final solution might be obtained by taking a second best split earlier
- When separation is in linear combinations of variables trees struggle to provide a good classification

Random forests - overview

- Multiple trees, fit to samples
- Sample cases, using bootstrapping (ones not chosen are called out-of-bag, used for testing purposes)
- Sample variables
- Lots of control parameters
- Lots of diagnostics generated!

Bagging

- Bagging stands for "bootstrap aggregation". Combine the results from multiple models built on different bootstrap samples.
- Random forests are an example of bagging
- Bagging can be used with almost any classifier
- Bagging reduces variation in estimates

Forest algorithm

1. Input: $L = (x_i, y_i), i = 1, ..., n, y_i \in \{1, ..., g\}, m < p$, number of variables chosen for each tree, B is the number of bootstrap samples.
2. For $b = 1, 2, ..., B$:

   - Draw a bootstrap sample, L^{*b} of size n^{*b} from L.
   - Grow tree classifier, T^{*b}. At each node use a random selection of m variables, and grow to maximum depth without pruning.
   - Predict the class of each case not drawn in L^{*b}.

3. Combine the predictions for each case, by majority vote, to give predicted class.

Input defaults

- B is at least 1000
- $m = \sqrt{(p)}$
- n^{*b} is usually about $\frac{2}{3}n$

Error

Compute the proportion of times the case is misclassified when it is out-of-bag (oob). Average these to give the predictive error.

Diagnostics

- Variable importance: more complicated than one might think
- Vote matrix, $n \times g$: Proportion of times a case is predicted to the class k.
- Proximities, $n \times n$: Closeness of cases measured by how often they are in the same terminal node.

Variable importance

1. For every tree predict the oob cases and count the number of votes cast for the correct class.
2. Randomly permute the values on a variable in the oob cases and predict the class for these cases.
3. Subtract the number of votes for the correct class in the variable-permuted oob cases from the number of votes for the correct class in the real oob cases. The average of this number over all trees in the forest is the raw importance score for that variable. If the value is small, then the variable is not very important.

Gini importance

- Gini importance adds up the difference in impurity value of the descendant nodes with the parent node.
- Quick to calculate, and usually consistent with the results of the permutation method.

Vote matrix

- Proportion of trees the case is predicted to be each class, ranges between 0-1
- Can be used to identify troublesome cases.
- Used with plots of the actual data can help determine if it is the record itself that is the problem, or if it is a limitation of the method.
- Understand the difference in accuracy of prediction for different classes.

Proximities

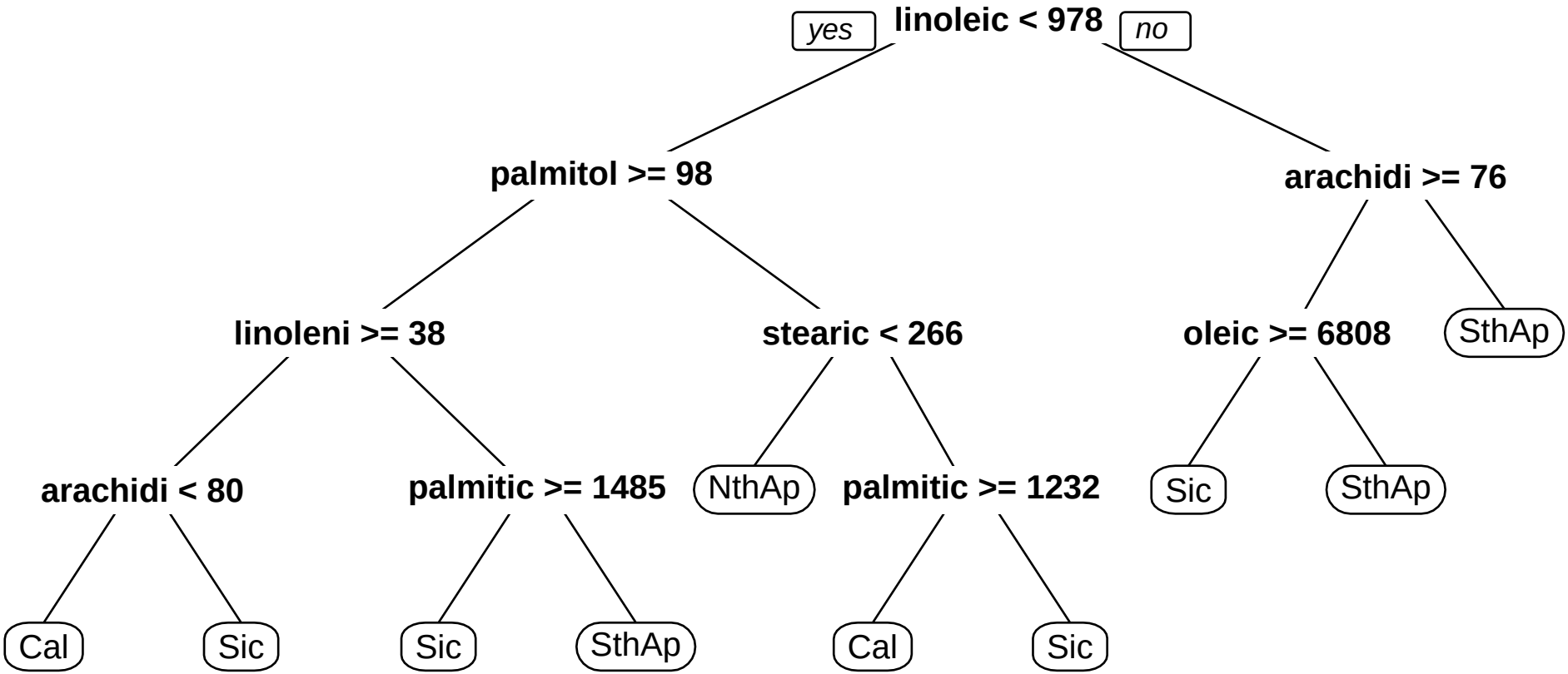
- Run both in- and out-of-bag cases down the tree, and increase proximity value of cases i, j by 1 each time they are in the same terminal node.
- Normalize by dividing by B.

Example: Fit tree to olive samples from the south

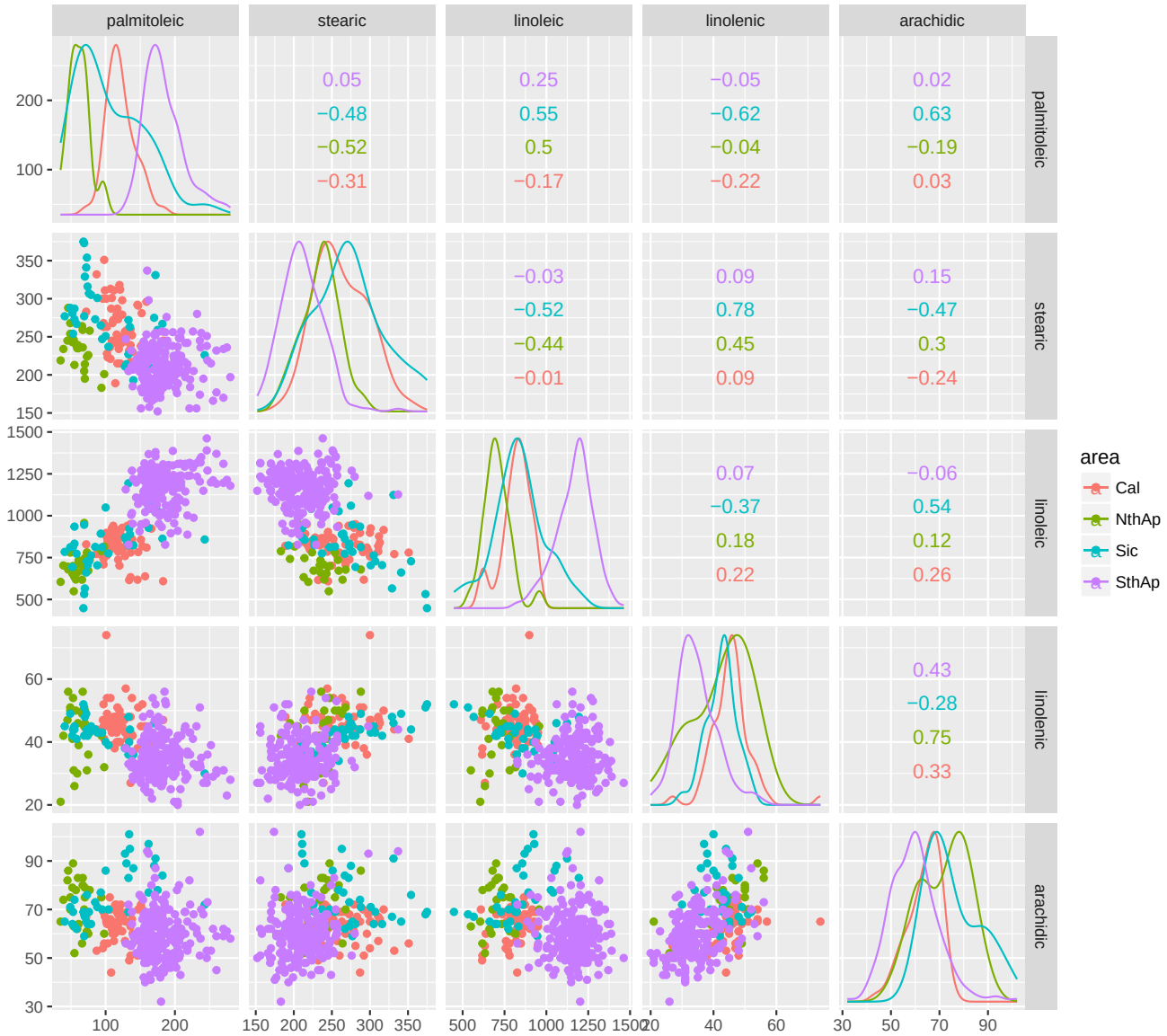
```
##        Cal NthAp Sic SthAp Sum   error
## Cal     25     1   0     2  28 0.10714
## NthAp    0    11   1     0  12 0.08333
## Sic      1     3  10     4  18 0.44444
## SthAp    2     0   5    97 104 0.06731
```

Test error = 0.117

Example: Tree model



Example: A look at the data



Example: Fit a random forest model

```
##
## Call:
##  randomForest(formula = area ~ ., data = olive.sth, importa
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8.05%
## Confusion matrix:
##       Cal NthAp Sic SthAp class.error
## Cal    53     0   1     2     0.05357
## NthAp   1    22   1     1     0.12000
## Sic     4     3  21     8     0.41667
## SthAp   2     0   3   201     0.02427
```

Example: Think about it

- Error rates: notice anything?
- What were the input parameters?

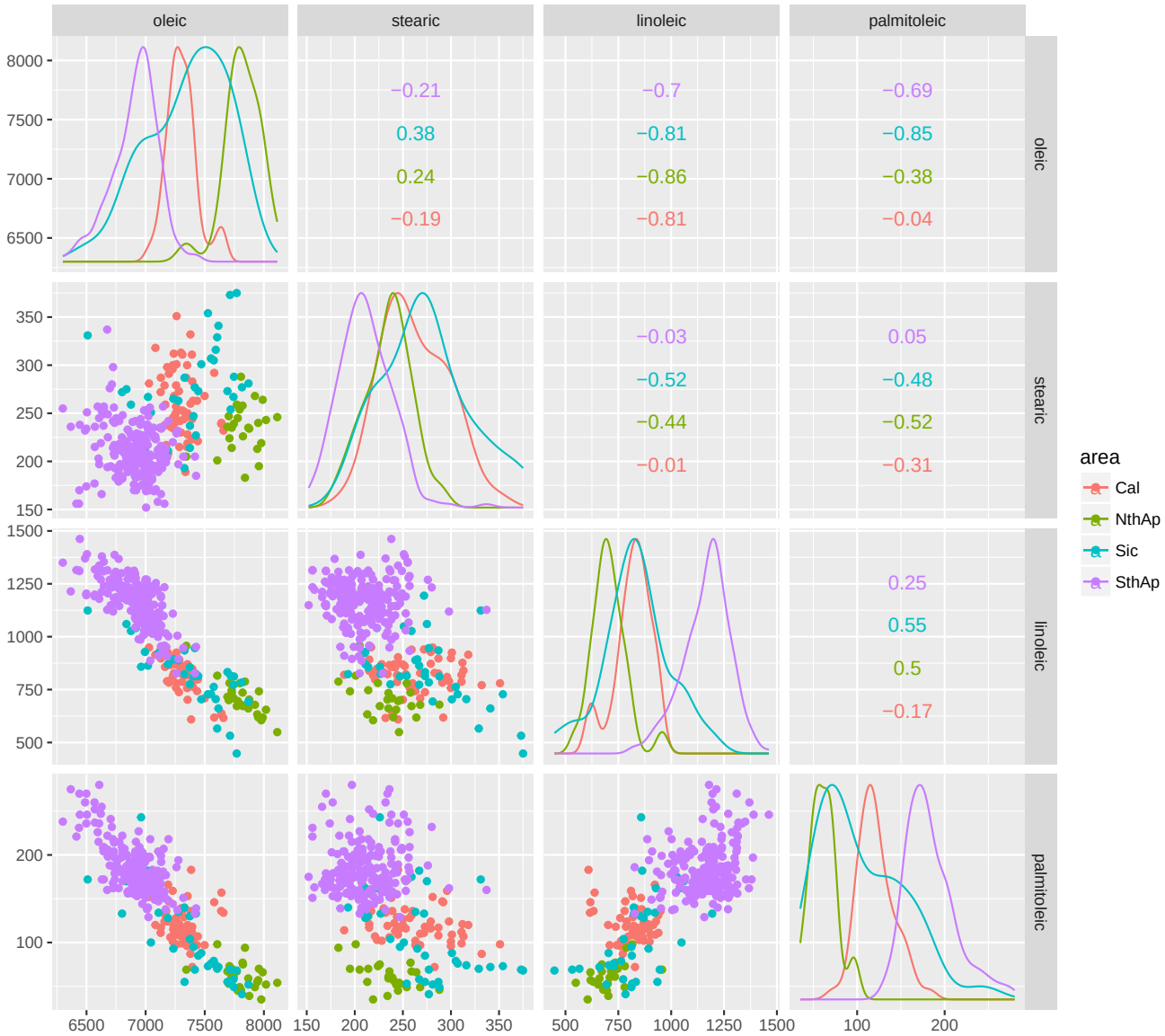
Example: Variable importance, overall

```
##          vars MeanDecreaseAccuracy MeanDecreaseGini
## 1    linoleic                 0.148              38
## 2 palmitoleic                 0.118              37
## 3       oleic                 0.090              30
## 4     stearic                 0.048              18
## 5    palmitic                 0.043              16
## 6   linolenic                 0.036              14
## 7   arachidic                 0.023              12
## 8  eicosenoic                 0.020              11
```

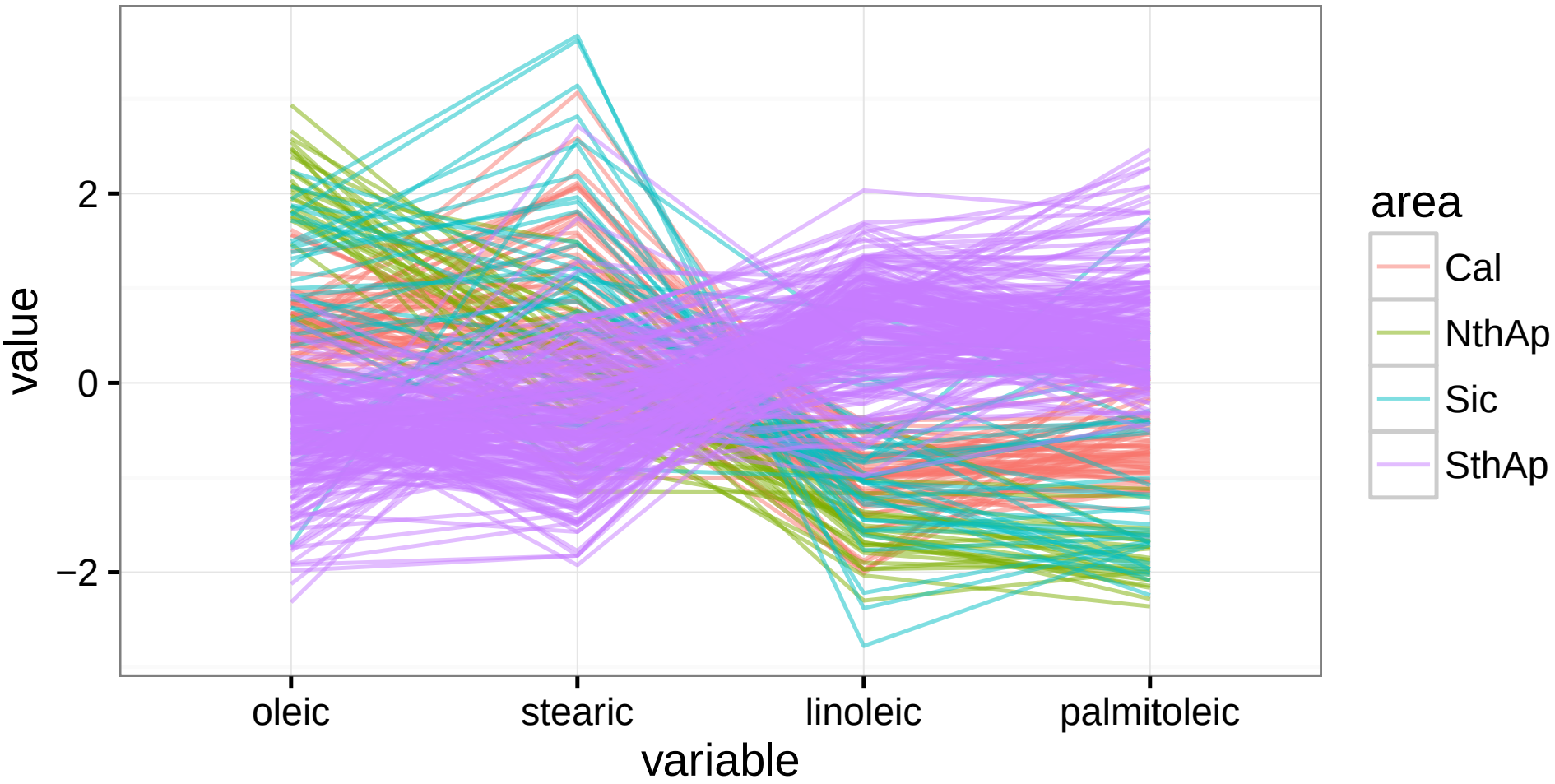
Example: by class

```
##          vars    Cal   NthAp   Sic  SthAp
## 1    linoleic 0.2667  0.1972 0.033 0.1321
## 2       oleic 0.1417  0.3121 0.031 0.0611
## 3   linolenic 0.1346  0.0074 0.042 0.0123
## 4 palmitoleic 0.0982  0.2878 0.137 0.1006
## 5     stearic 0.0610 -0.0016 0.119 0.0388
## 6    palmitic 0.0312  0.3024 0.024 0.0177
## 7   arachidic 0.0312  0.0549 0.098 0.0046
## 8  eicosenoic 0.0057  0.0033 0.108 0.0116
```

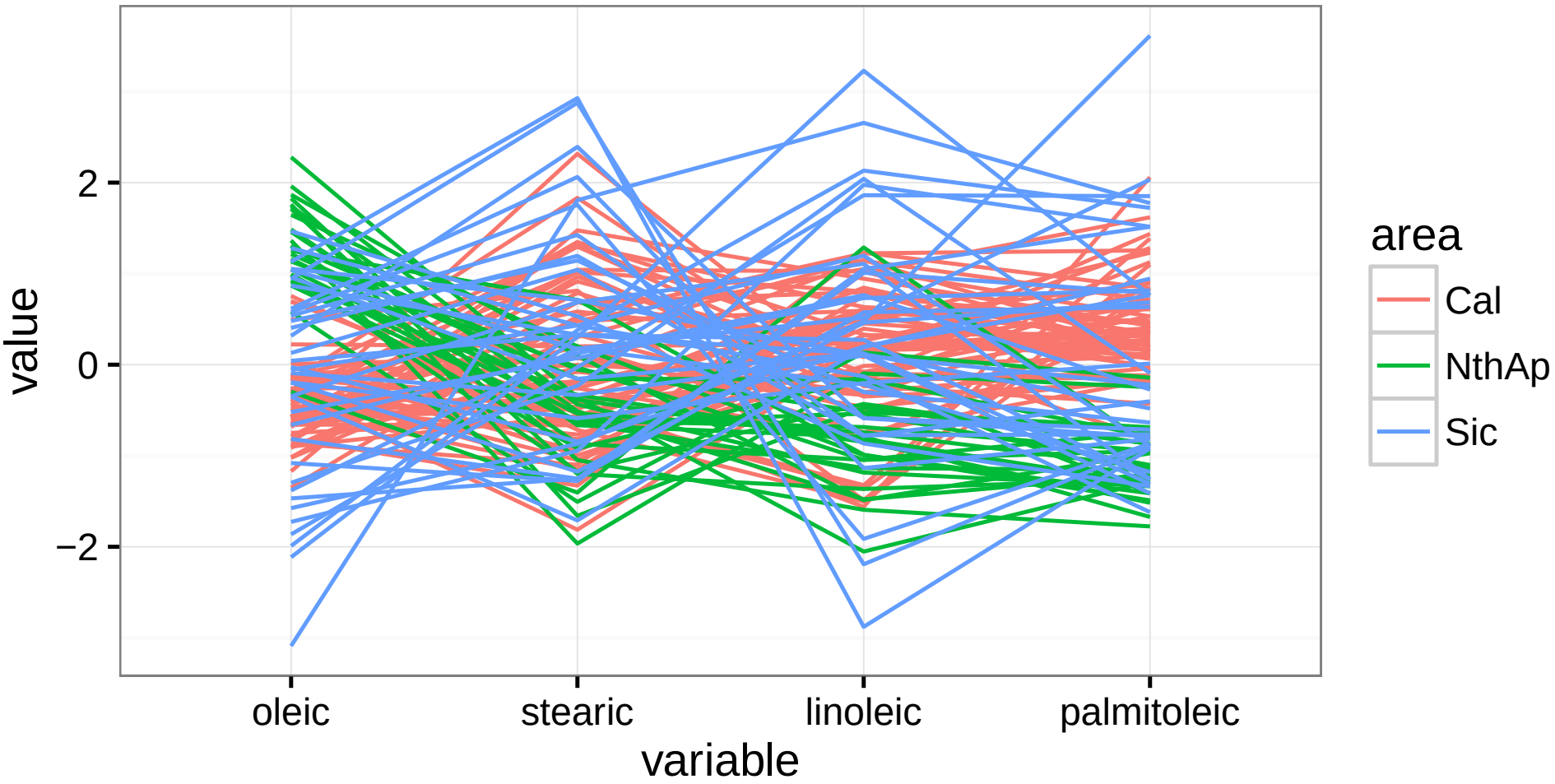
Example: Use to choose vars for scatmat



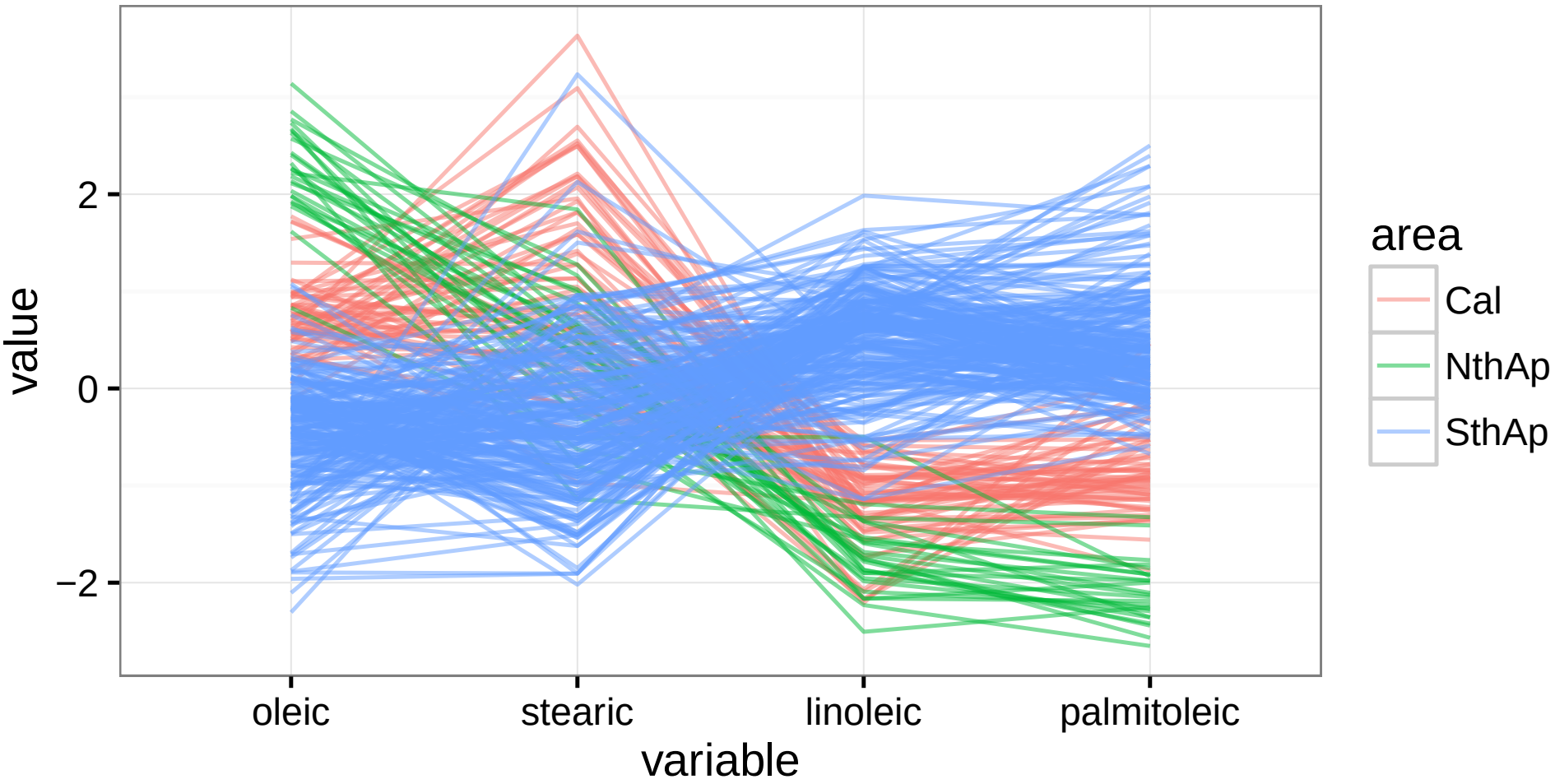
Example: Use to arrange par coords



Example: Without the large group



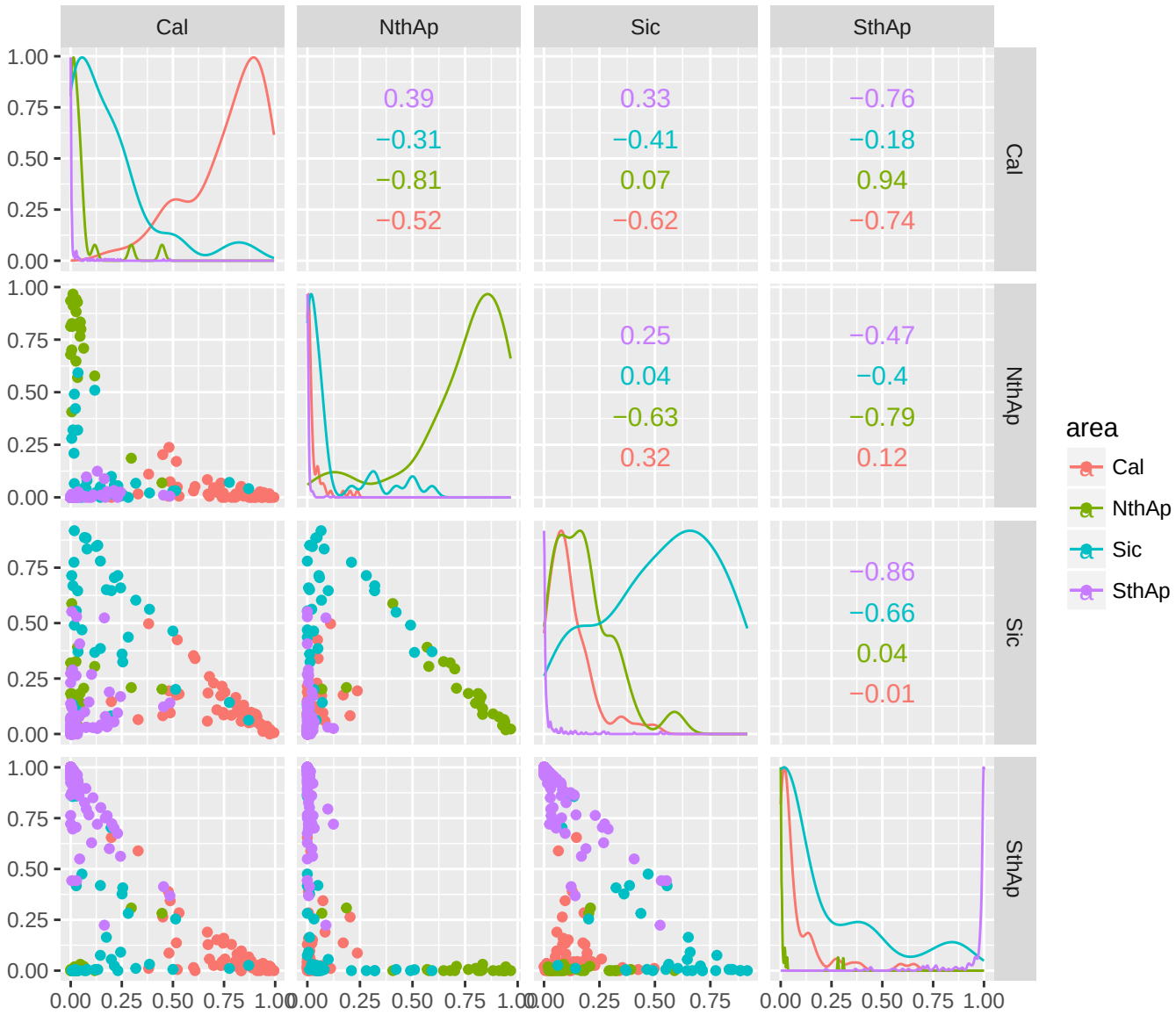
Example: Without the trouble maker class



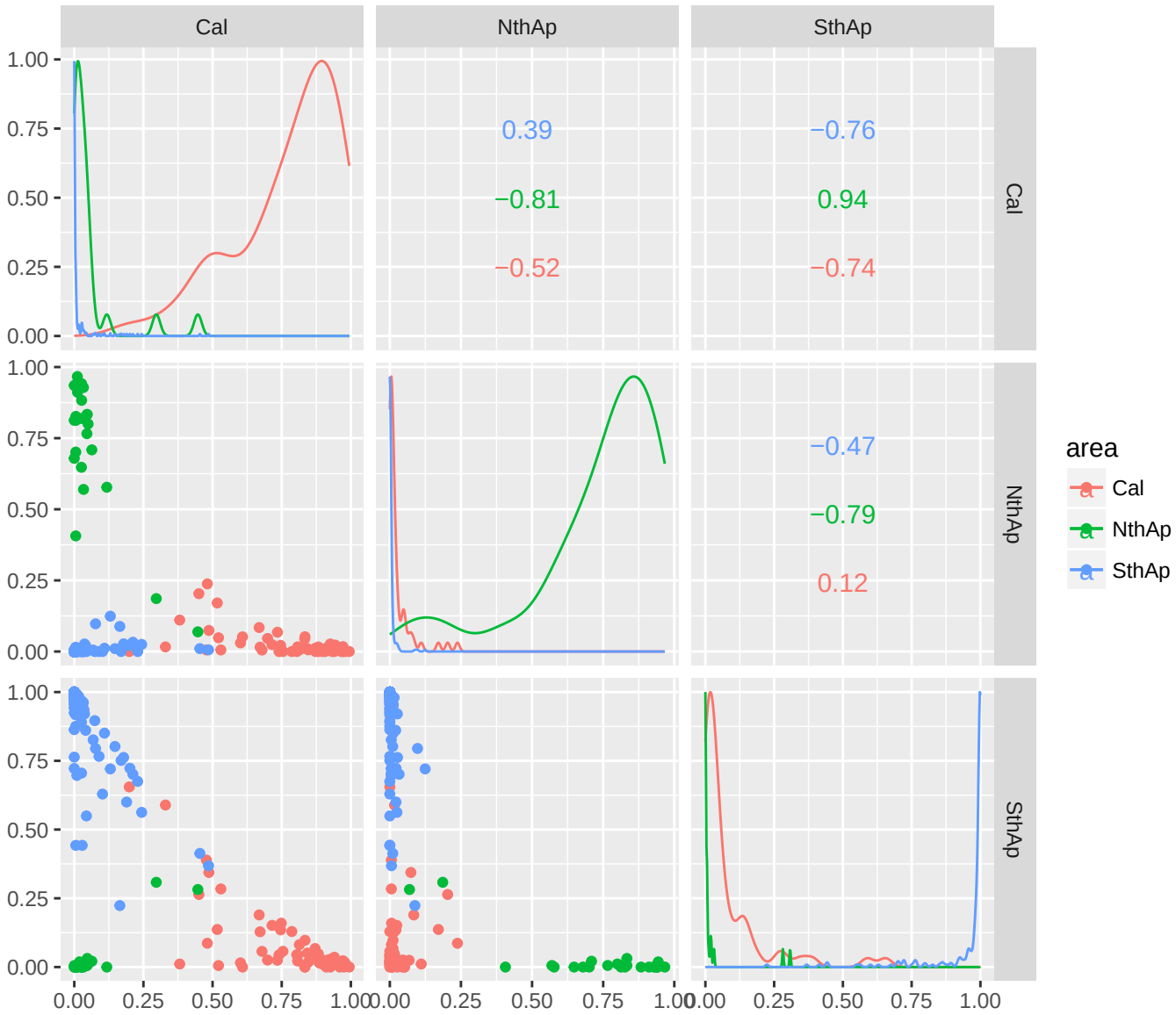
Example: Vote matrix

```
##     Cal  NthAp    Sic SthAp
## 1 0.79 0.0000 0.0843 0.129
## 2 0.49 0.0741 0.0952 0.344
## 3 0.98 0.0000 0.0053 0.016
## 4 0.83 0.0114 0.0571 0.097
## 5 0.88 0.0000 0.0979 0.021
## 6 0.68 0.0052 0.2591 0.057
```

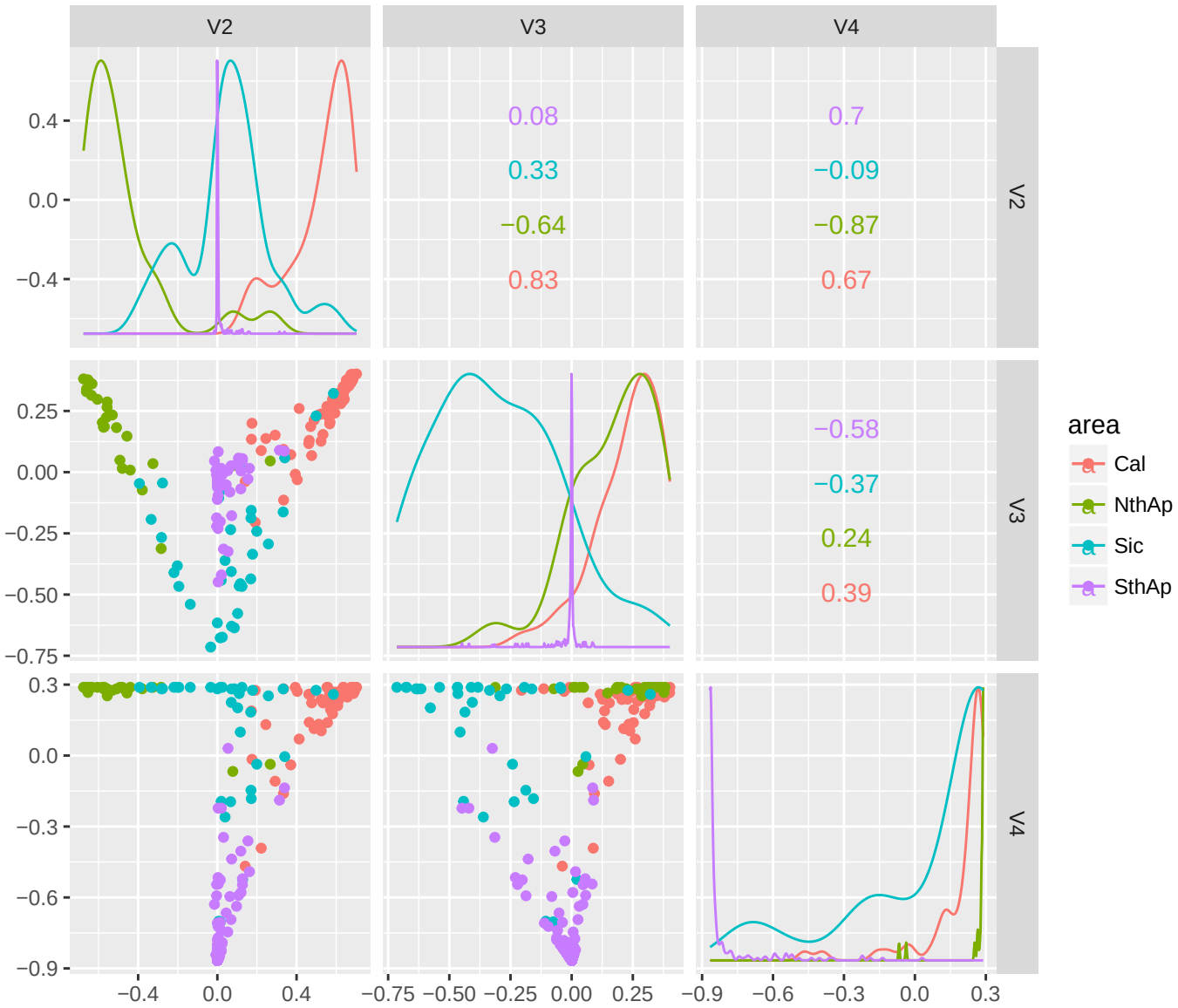
Example: Vote matrix



Example: Vote matrix



Example: Vote matrix



Videos explaining exploring trees and forests

Cook & Swayne (2007) "Interactive and Dynamic Graphics for Data Analysis: With Examples Using R and GGobi" have several videos illustrating techniques for exploring high-dimensional data in association with trees and forest classifiers:

- Trees
- Forests

And this paper (Wickham, Cook and Hofmann, 2015) contains links to videos describing why and how of visualisaing models in high-dimensional spaces.

Proximity matrix

- 323×323 matrix, effectively a distance matrix for all cases from each other
- These distances can be passed to an unsupervised classification, clustering, to examine similarity between cases.
- You would expect cases in different classes to be further from each other, cases within the same class to be close to each other by this metric.
- We will talk about clustering in the next section of the class.

Share and share alike