Tree-based methods

Rosa Karlić

NGS School 2019: Machine Learning in Biomedicine

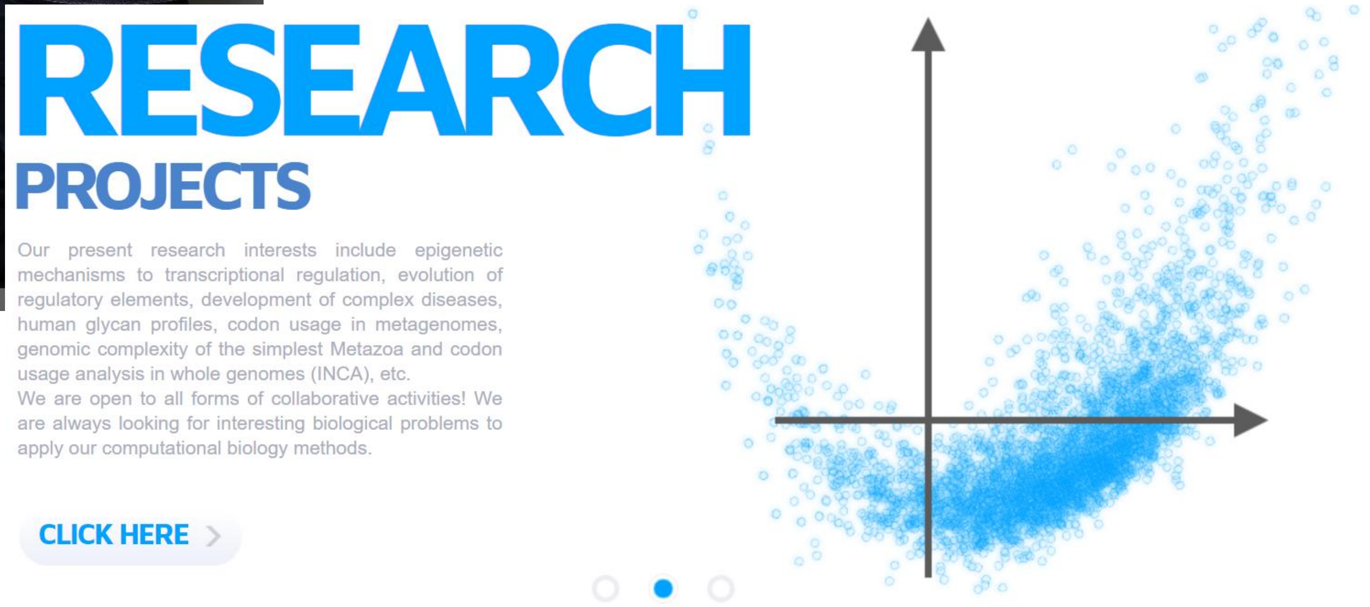
Białobrzegi, Poland

28.10.2019.

Bioinformatics Group, Faculty of Science, University of Zagreb, Croatia

Prof. Kristian Vlahoviček
Asst.Prof. Rosa Karlić

PhD students:
Maja Kuzman
Filip Horvat
Antonio Ruiz
Dunja Glavaš



RESEARCH
PROJECTS

Our present research interests include epigenetic mechanisms to transcriptional regulation, evolution of regulatory elements, development of complex diseases, human glycan profiles, codon usage in metagenomes, genomic complexity of the simplest Metazoa and codon usage analysis in whole genomes (INCA), etc.
We are open to all forms of collaborative activities! We are always looking for interesting biological problems to apply our computational biology methods.

www.bioinfo.hr

Tree-based methods

Supervised learning methods, can be used for both classification and regression

Simple but quite powerful

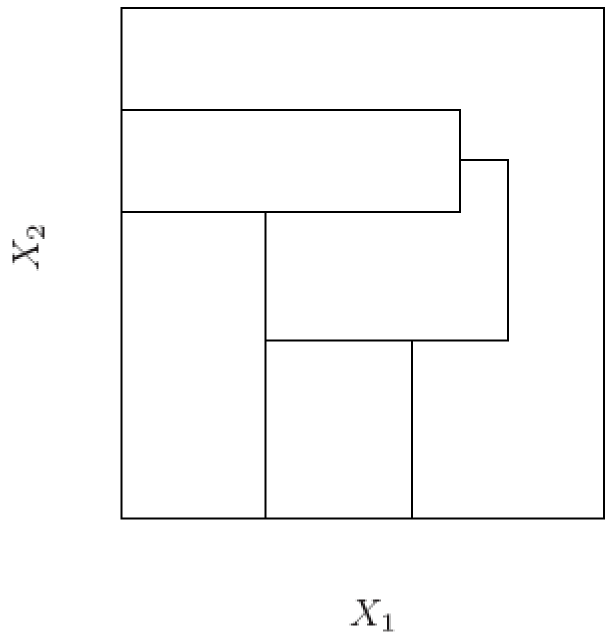
Easy to interpret

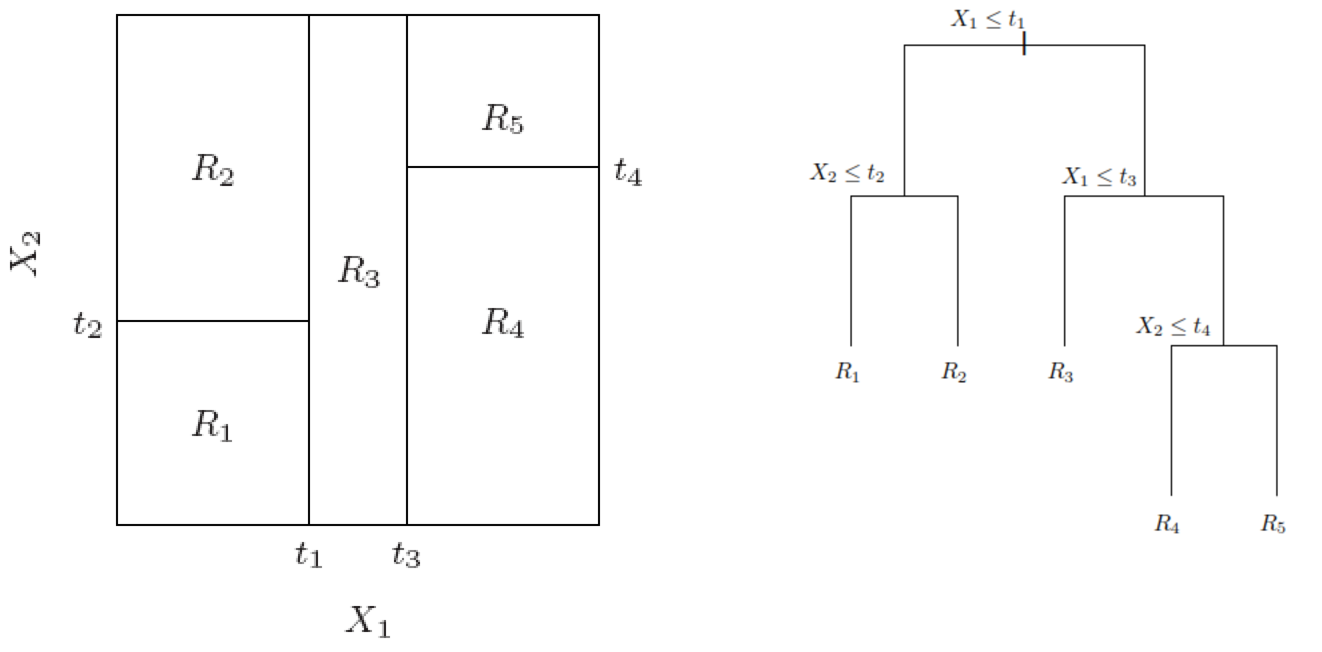
CART - popular method for tree-based regression and classification

Various applications in computational and systems biology:

- modeling regulatory networks, secondary structure prediction, prediction of drug resistance, prediction of protein-protein interactions, …

Decision trees





Partition the feature space into a set of rectangles, and then fit a simple model in each one.

Although each partitioning line has a simple description like *X1 = c,* some of the resulting regions are complicated to describe - to simplify, we restrict attention to recursive binary partitions

Nodes and leaves

Classification trees

In a node *m*, representing a region R_m with N_m observations, we calculate the proportion of each class, $\hat{p}_{mk}$

We classify the observations in node *m* to class

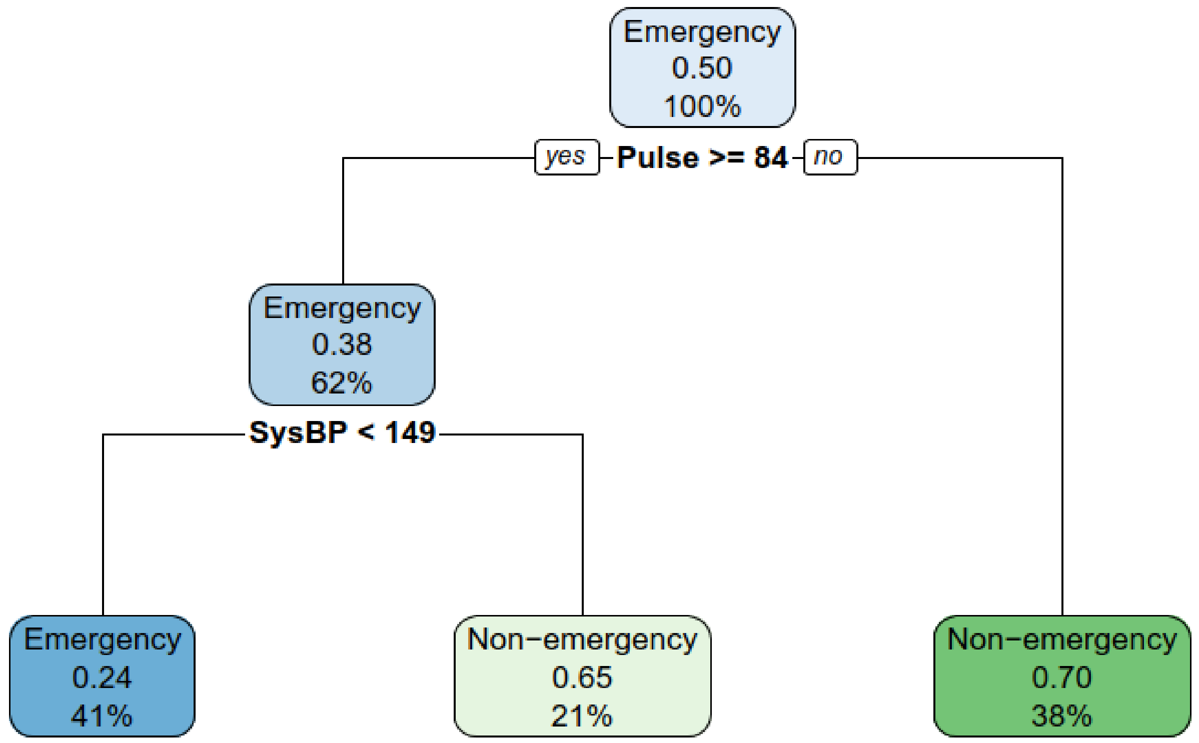
$$k(m) = \arg\max_k \hat{p}_{mk}$$

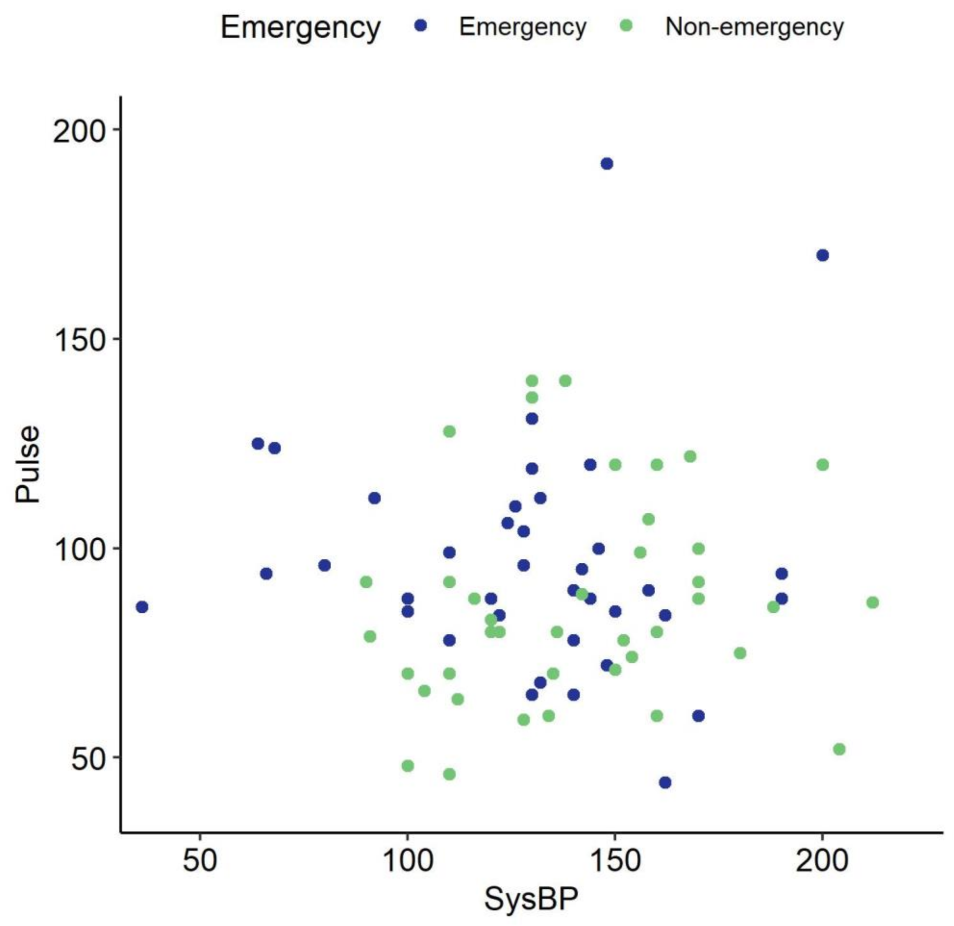
Criteria for splitting the nodes – different measures of node impurity:

Missclassification error	Gini index	Cross-entropy
$1 - \hat{p}_{mk(m)}$	$\sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})$	$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

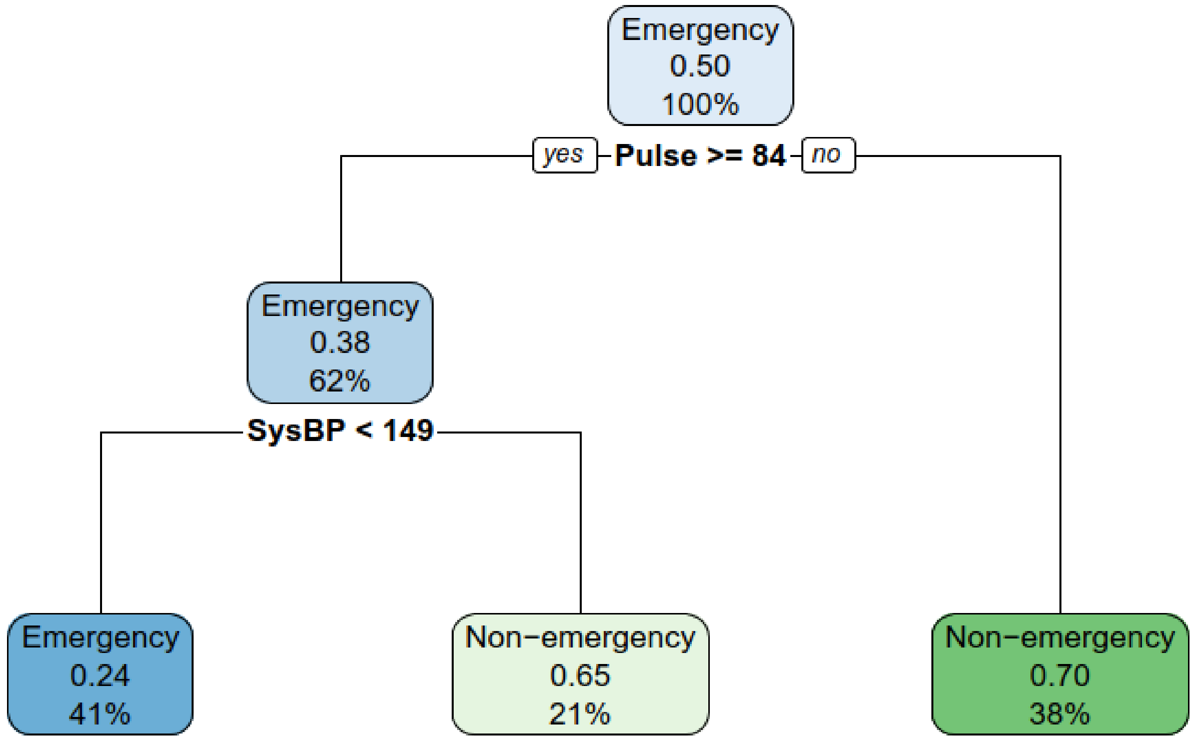
For max k

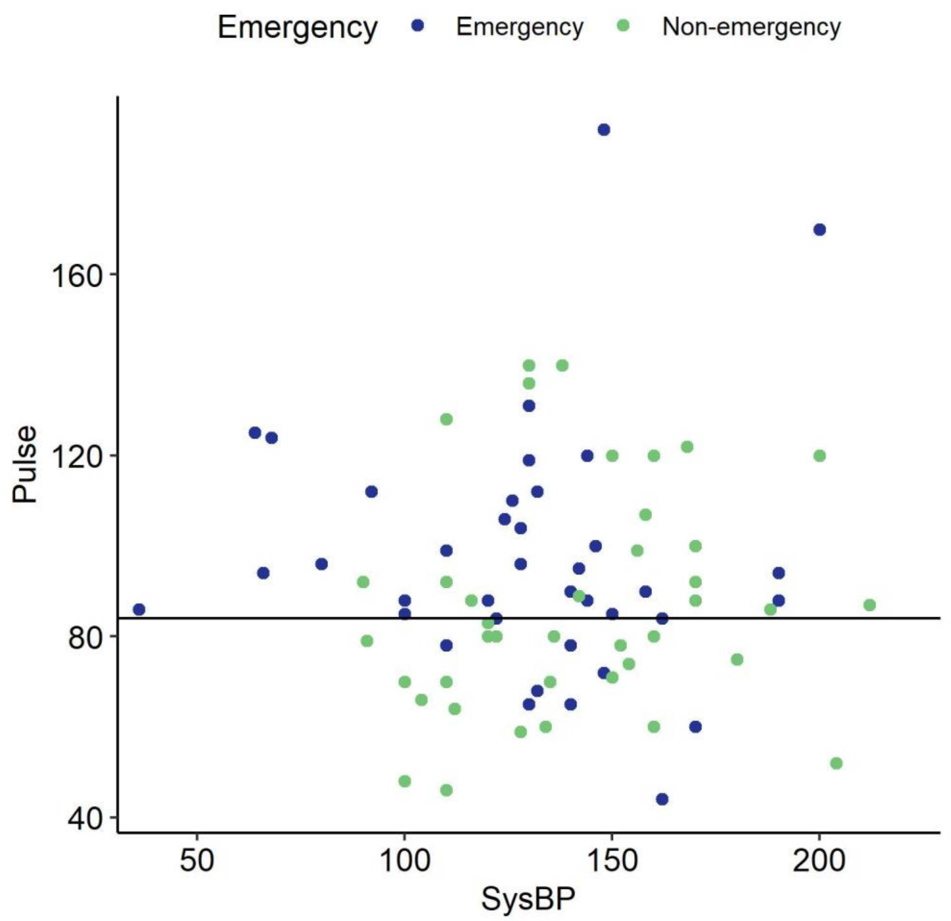
Classification tree - example



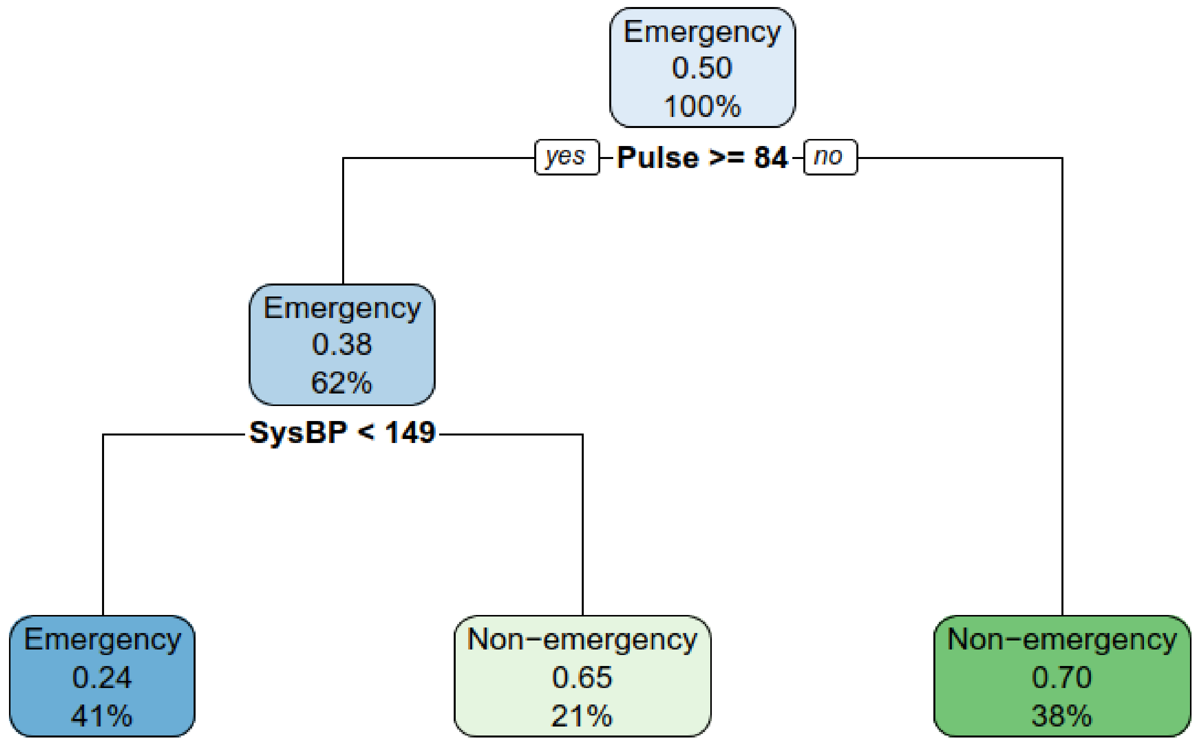


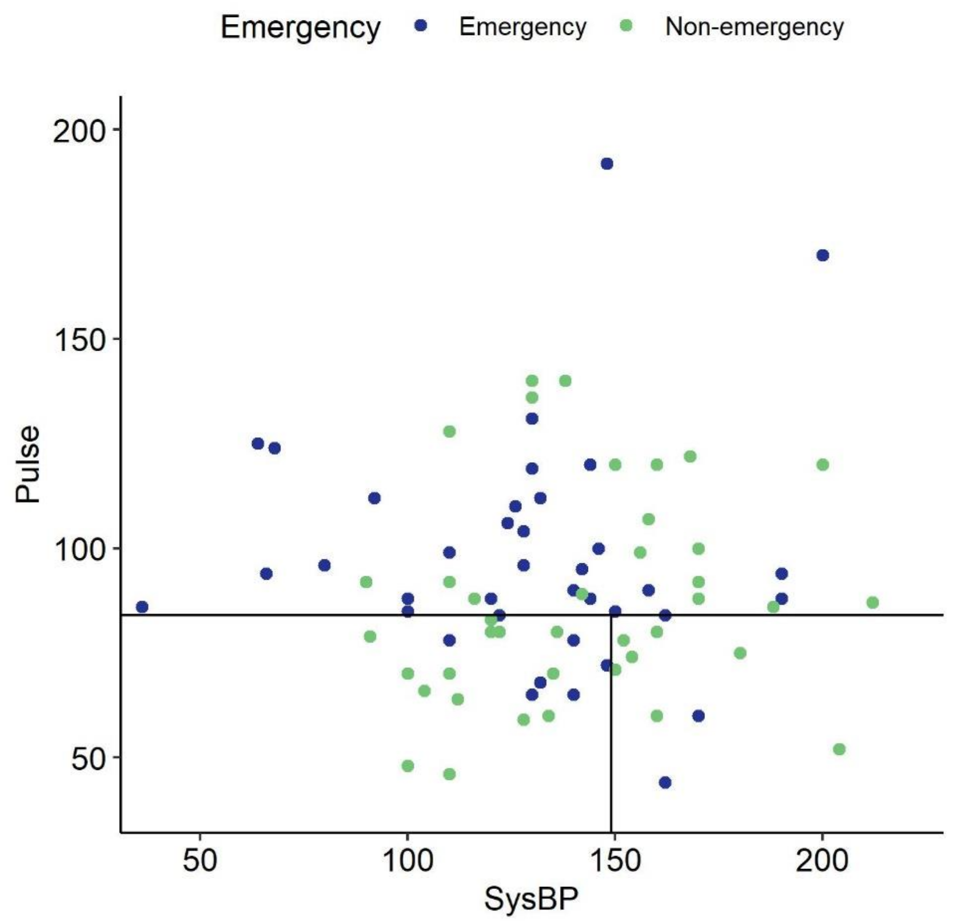
Classification tree - example



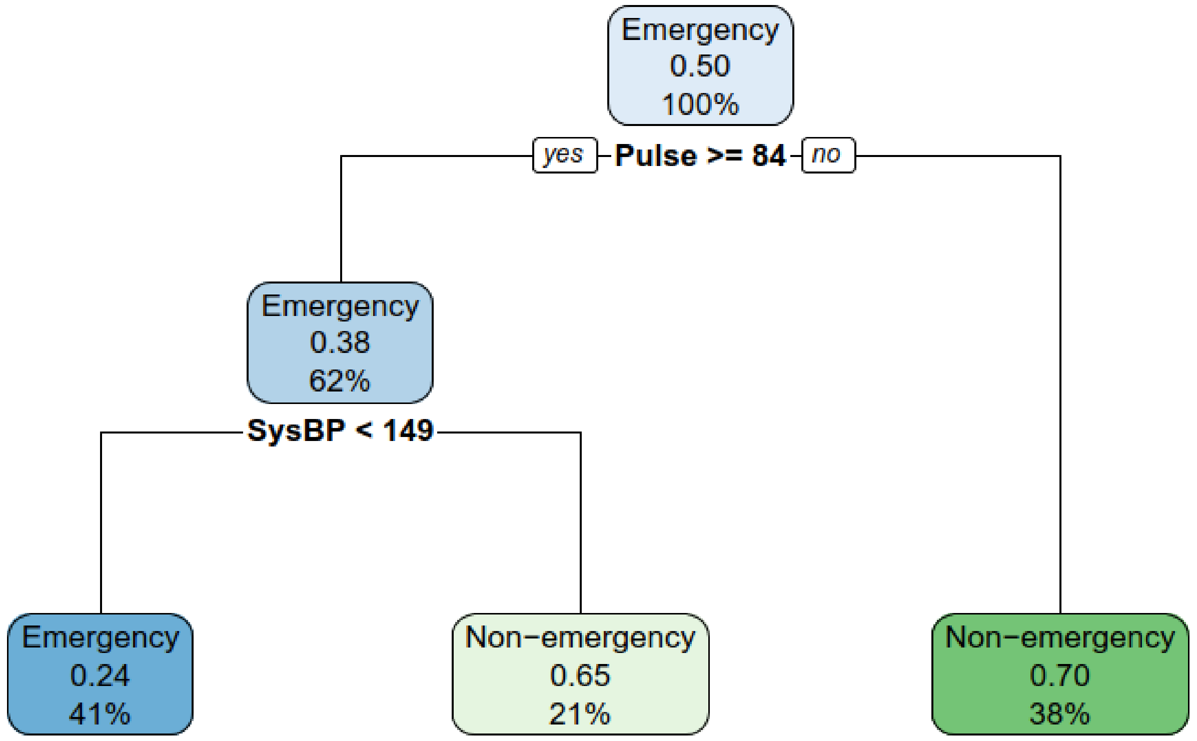


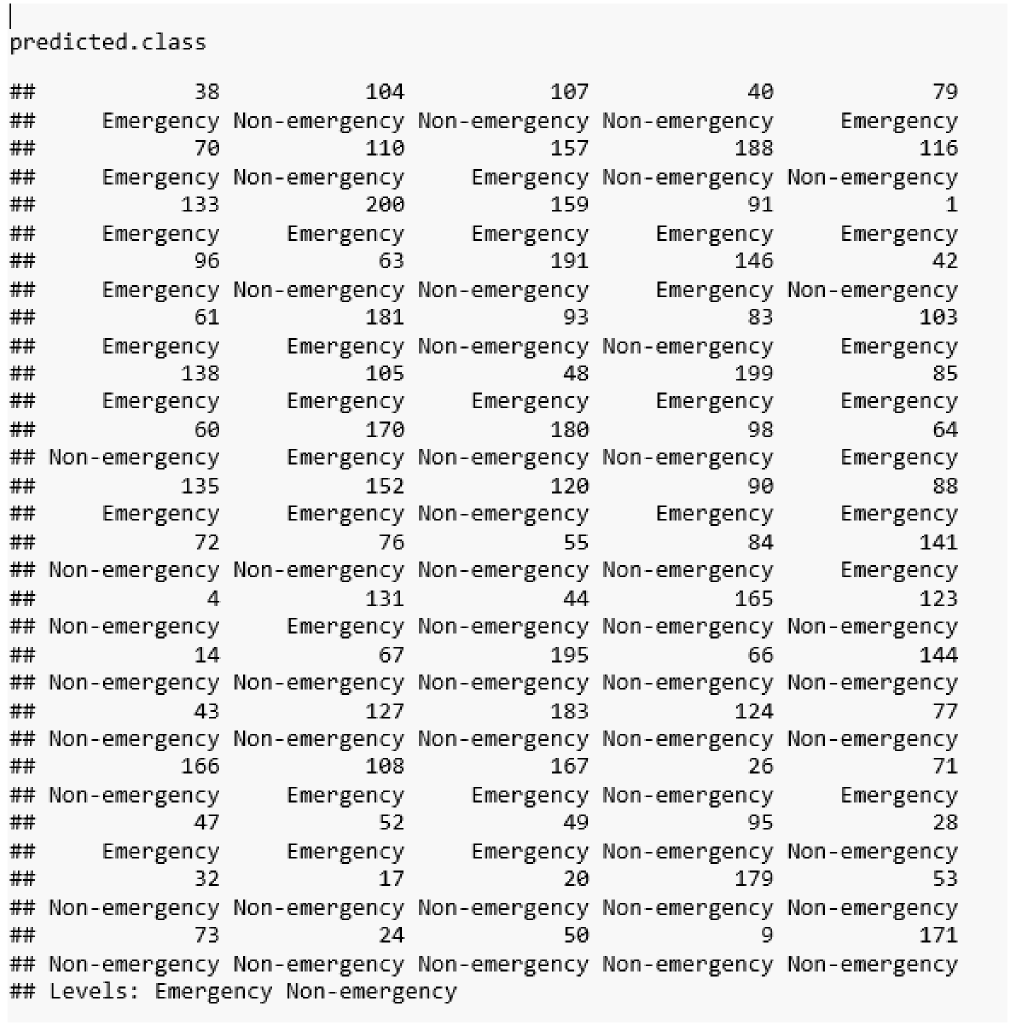
Classification tree - example





Classification tree - example

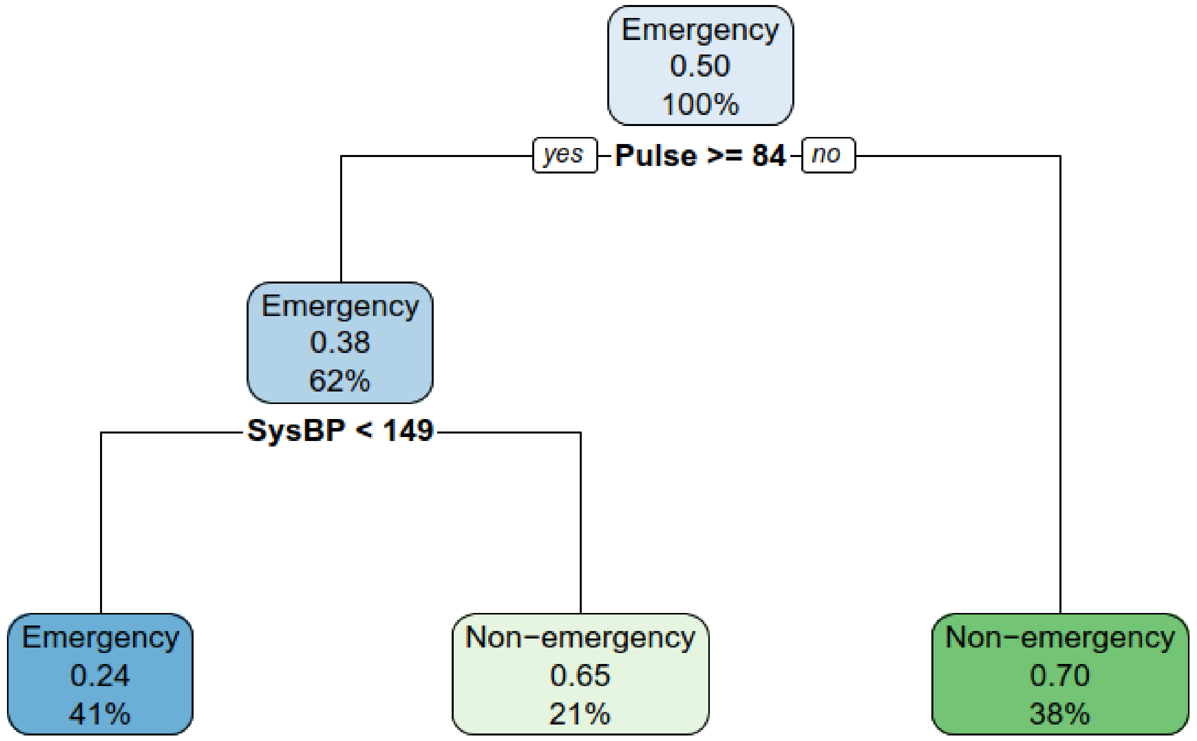




```
predicted.class

##            38           104           107            40            79
##     Emergency Non-emergency Non-emergency Non-emergency     Emergency
##            70           110           157           188           116
##     Emergency Non-emergency     Emergency Non-emergency Non-emergency
##           133           200           159            91             1
##     Emergency     Emergency     Emergency     Emergency     Emergency
##            96            63           191           146            42
##     Emergency Non-emergency Non-emergency     Emergency Non-emergency
##            61           181            93            83           103
##     Emergency     Emergency Non-emergency Non-emergency     Emergency
##           138           105            48           199            85
##     Emergency     Emergency     Emergency     Emergency     Emergency
##            60           170           180            98            64
## Non-emergency     Emergency Non-emergency Non-emergency     Emergency
##           135           152           120            90            88
##     Emergency     Emergency Non-emergency     Emergency     Emergency
##            72            76            55            84           141
## Non-emergency Non-emergency Non-emergency Non-emergency     Emergency
##             4           131            44           165           123
## Non-emergency     Emergency Non-emergency Non-emergency Non-emergency
##            14            67           195            66           144
## Non-emergency Non-emergency Non-emergency Non-emergency Non-emergency
##            43           127           183           124            77
## Non-emergency Non-emergency Non-emergency Non-emergency Non-emergency
##           166           108           167            26            71
## Non-emergency     Emergency     Emergency Non-emergency     Emergency
##            47            52            49            95            28
##     Emergency     Emergency     Emergency Non-emergency Non-emergency
##            32            17            20           179            53
## Non-emergency Non-emergency Non-emergency Non-emergency Non-emergency
##            73            24            50             9           171
## Non-emergency Non-emergency Non-emergency Non-emergency Non-emergency
## Levels: Emergency Non-emergency
```

How do we calculate the accuracy of the model?



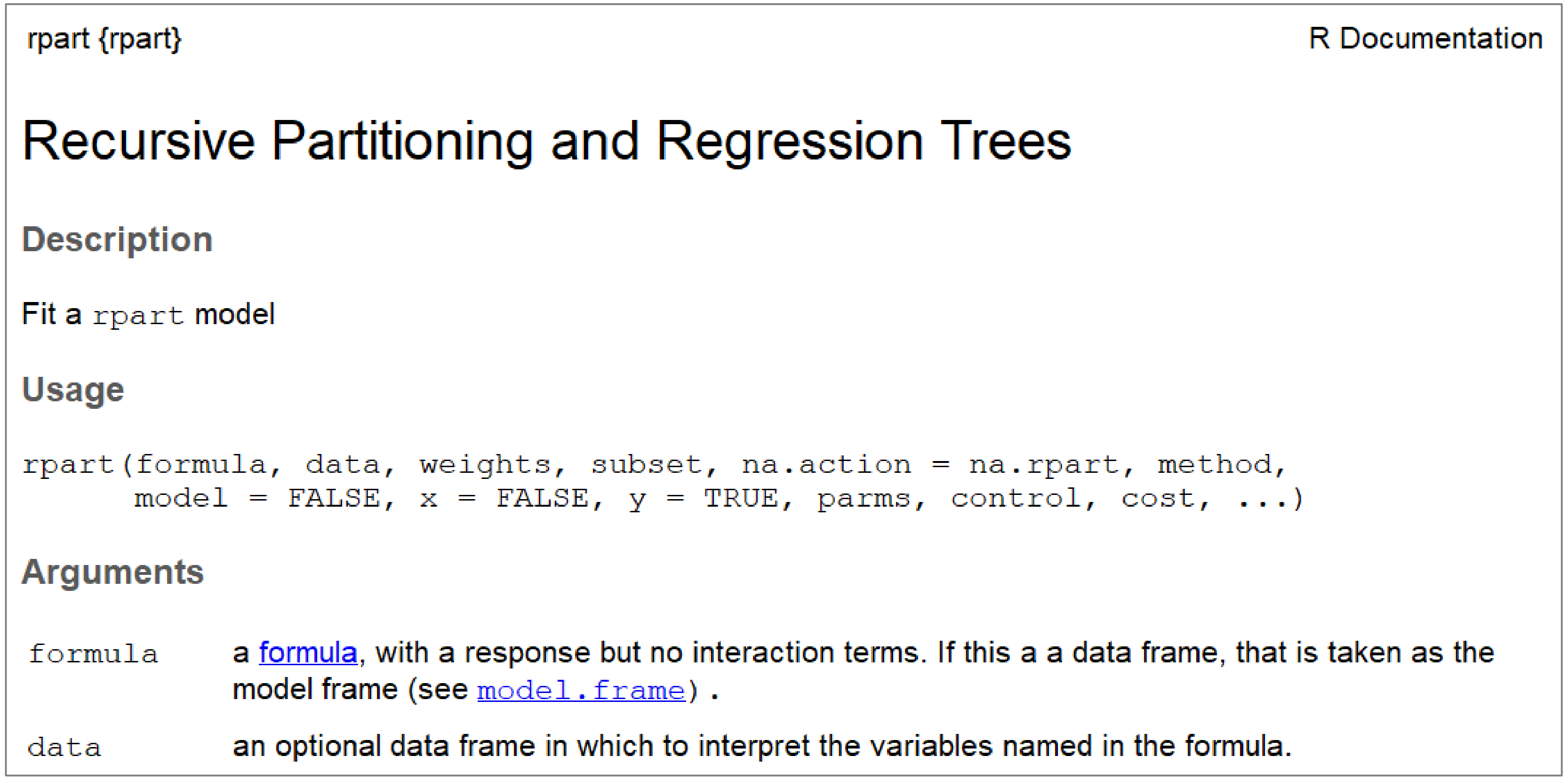
Confusion matrix

	Emergency	Non-emergency
Emergency	25	8
Non-emergency	15	32

Classification error rate

```
> (8+15)/80
[1] 0.2875
```

rpart()



rpart {rpart} R Documentation

Recursive Partitioning and Regression Trees

Description

Fit a `rpart` model

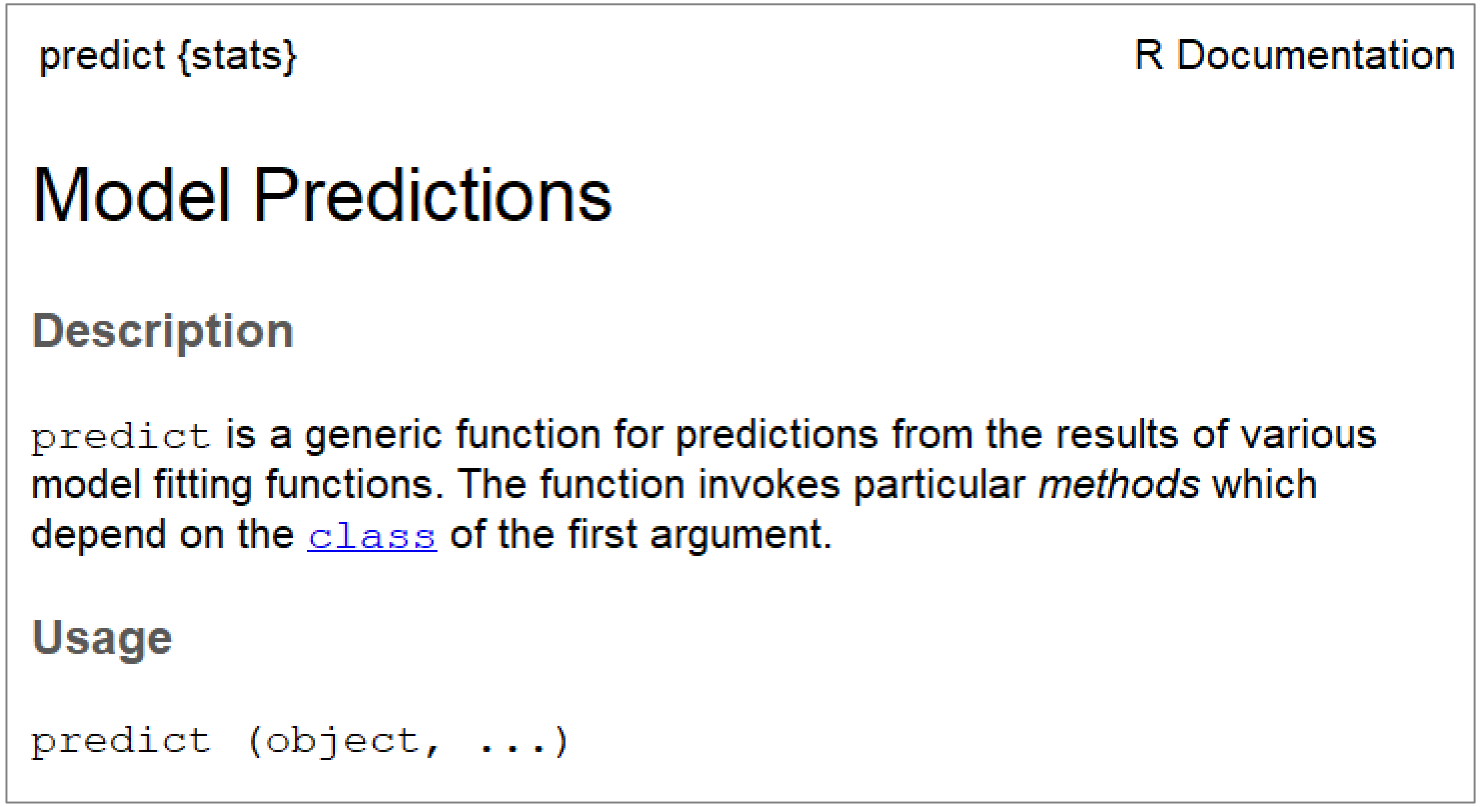
Usage

```
rpart(formula, data, weights, subset, na.action = na.rpart, method,
      model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)
```

Arguments

`formula`	a formula, with a response but no interaction terms. If this a a data frame, that is taken as the model frame (see `model.frame`) .
`data`	an optional data frame in which to interpret the variables named in the formula.

predict()



predict {stats} R Documentation

Model Predictions

Description

`predict` is a generic function for predictions from the results of various model fitting functions. The function invokes particular *methods* which depend on the `class` of the first argument.

Usage

```
predict (object, ...)
```

Exercise 1 – Classification trees with rpart

Using the rpart() function, create a classification tree using the adapted ICU data (made from the ICU dataset available in the Stat2Data package).

Plot the tree using the rpart.plot() function

Is the tree the same as before? What is your explanation for this?

Use the predict() function to classify the data and calculate the classification error rate of our tree.

What are the splitting rules for this tree?

How to grow a decision tree?

Criterion: minimize the error

Find regions R1 and R2 such that

$$R_{1(j,s)} = \left\{ X \mid X_j \leq s \right\} \text{ and } R_{2(j,s)} = \left\{ X \mid X_j > s \right\}$$

Use a greedy algorithm to find splitting variables *j* and splitting points *s*

How large should our tree be?

Split tree nodes only if the decrease in error due to the split exceeds some threshold

Or prune a large tree is using *cost-complexity pruning*

- $|T|$ - number of terminal nodes in the tree
- $Q_m(T)$ – impurity measure

Cost-complexitiy criterion

$$\boxed{C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|}$$

Optimal value of α can be determined by cross-validation

Training and test error

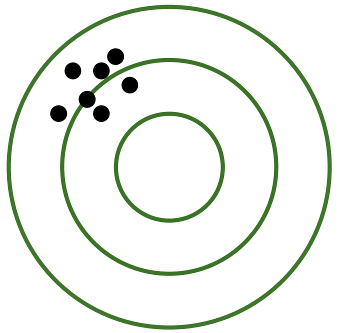
Test error – average error that results from using a statistical learning method to predict the response on a new observation (not used in training the model)

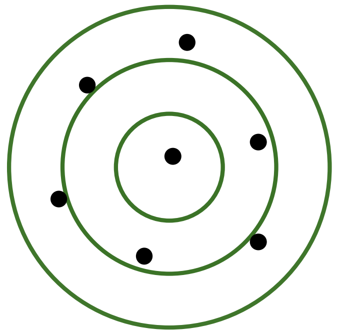
Training error – calculated by applying the statistical learning method to the observations used in its training

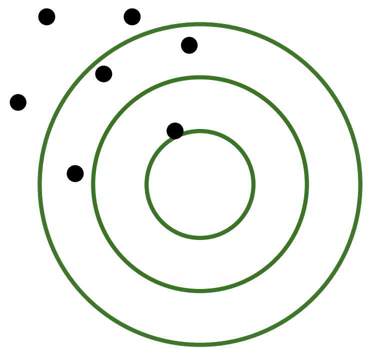
Training error rate often quite different from the test error rate (test error rate can be underestimated)

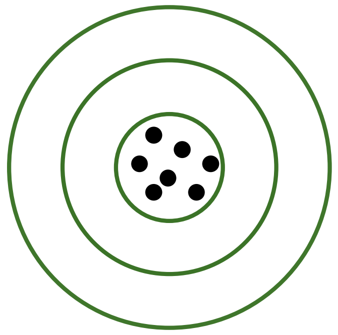
Overfitting!

Properties of estimators

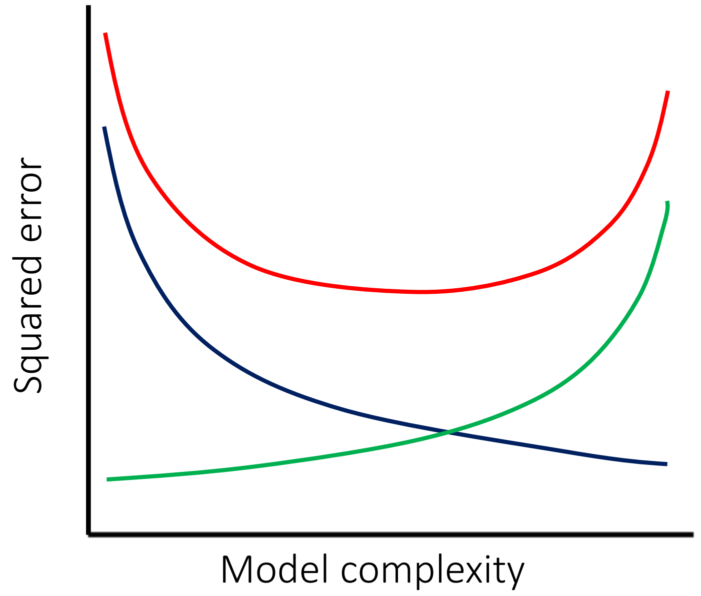

high bias
low variance


low bias
high variance

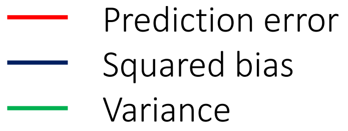

high bias
high variance


low bias
low variance

Bias – variance decomposition



$$\text{Prediction error} = Var(\hat{\Theta}) + Bias(\hat{\Theta})^2 + \sigma^2$$

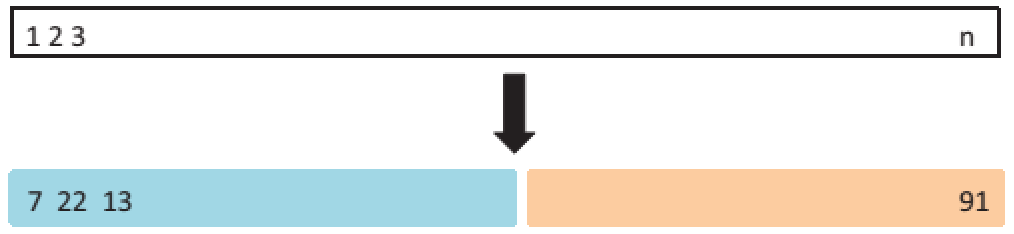


Cross-validation

Used to estimate the test error associated with a given statistical learning method in order to evaluate its performance (model assessment) or to select the appropriate level of flexibility (model selection)

- Validation set approach
- Leave-one-out cross-validation
- K-fold cross-validation

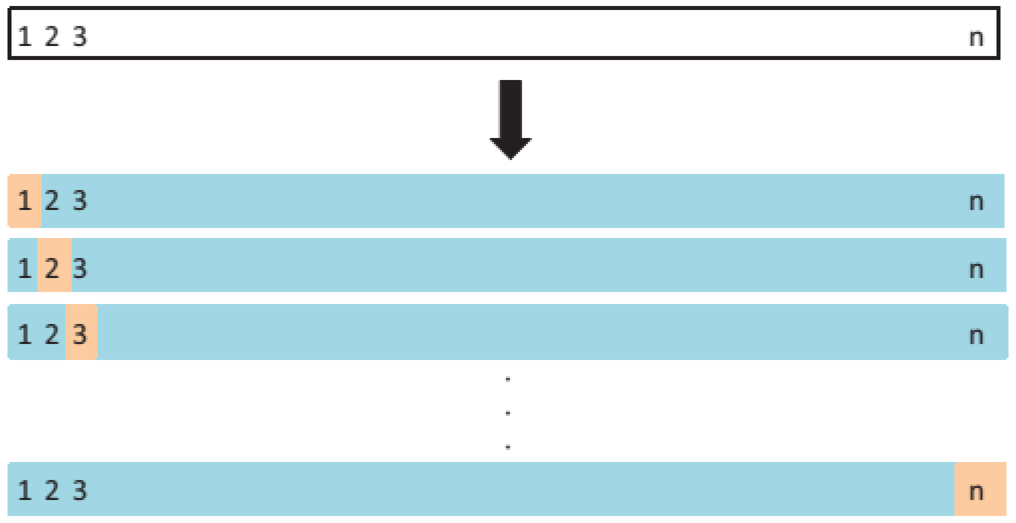
Validation set approach



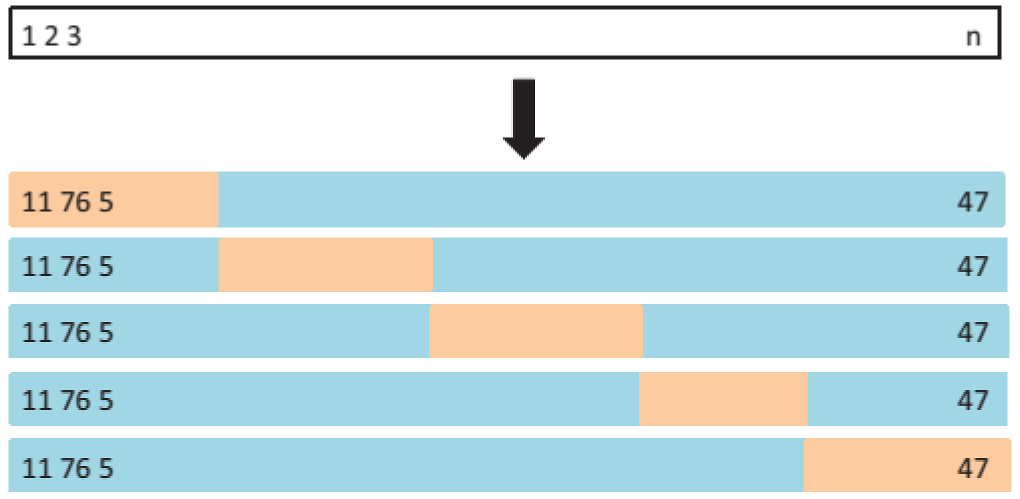
Variable estimates of test error rate

Might overestimate the test error rate

Leave-one-out cross-validation



K-fold cross-validation



Exercise 2 – How large should my tree be?

Set the cp parameter to 0.1 and construct another tree with the ICU data. What do you notice?

Check the cp parameter of the original tree that you created (ICU.tree object) - use the summary() and plotcp() functions.

What is the prediction accuracy of the original tree (ICU.tree object) but on new data?

Exercise 3

Read in the original ICU dataset from the Stat2Data package. You want to construct a training set with 50 Emergency and 50 Non-emergency patients and train a classification tree. Repeat this process 3 times and compare the resulting classification trees. Set the seed to 1, 2 and 3 respectively. What do you notice?

Exercise 3

Read in the original ICU dataset from the Stat2Data package. You want to construct a training set with 50 Emergency and 50 Non-emergency patients and train a classification tree. Repeat this process 3 times and compare the resulting classification trees. Set the seed to 1, 2 and 3 respectively. What do you notice?

Trees are instable!

- High variance of trees

- Small changes in the data can result in a very different series of splits because of the hierarchical nature of the process

Regression trees

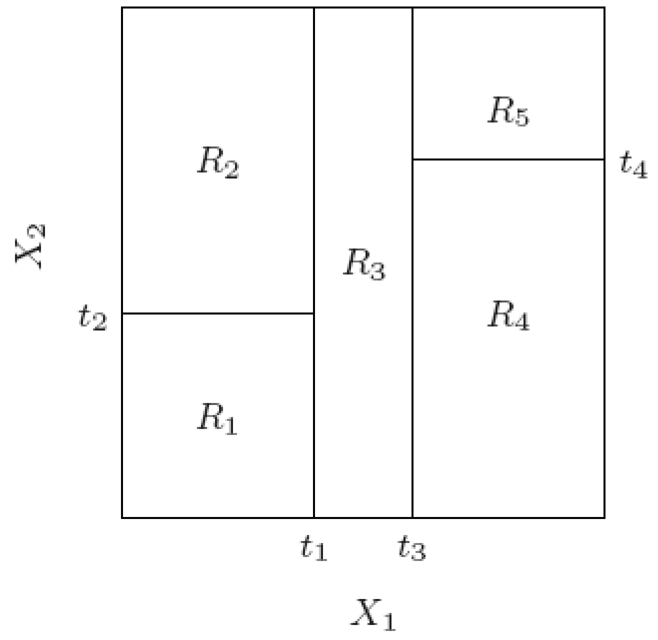
Regression problem with continuous response *Y* and inputs *X1* and *X2*

Split the space into two regions, and model the response by the mean of *Y* in each region.

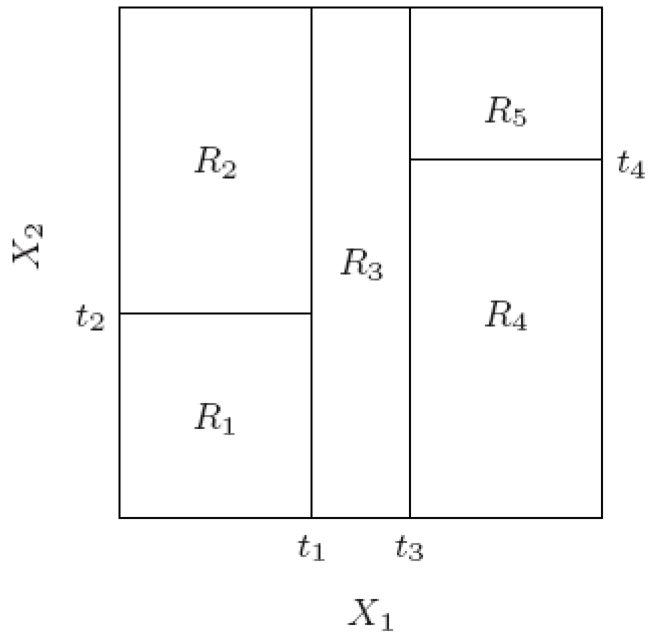
Choose the variable and split-point to achieve the best fit.

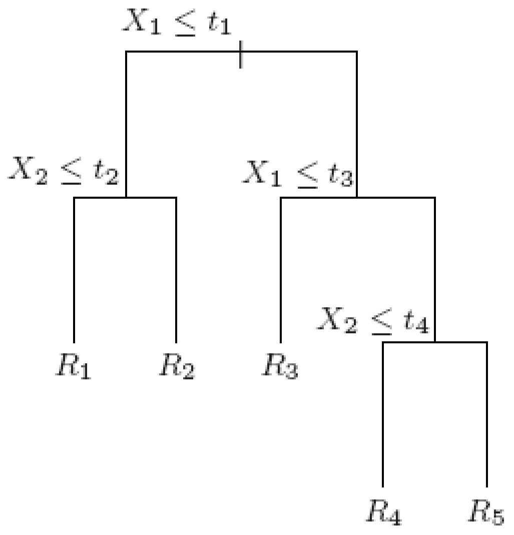
Continue until some stopping rule is applied.

$$\hat{y} = \frac{1}{c}\sum_{i=1}^{c} y_i$$



Regression trees





Model can be represented by a binary tree

Full dataset sits at the top of the tree and observations satisfying the condition at each junction are assigned to the left branch

Terminal nodes (leaves) of the tree correspond to the regions *R1, . . . , R5.*

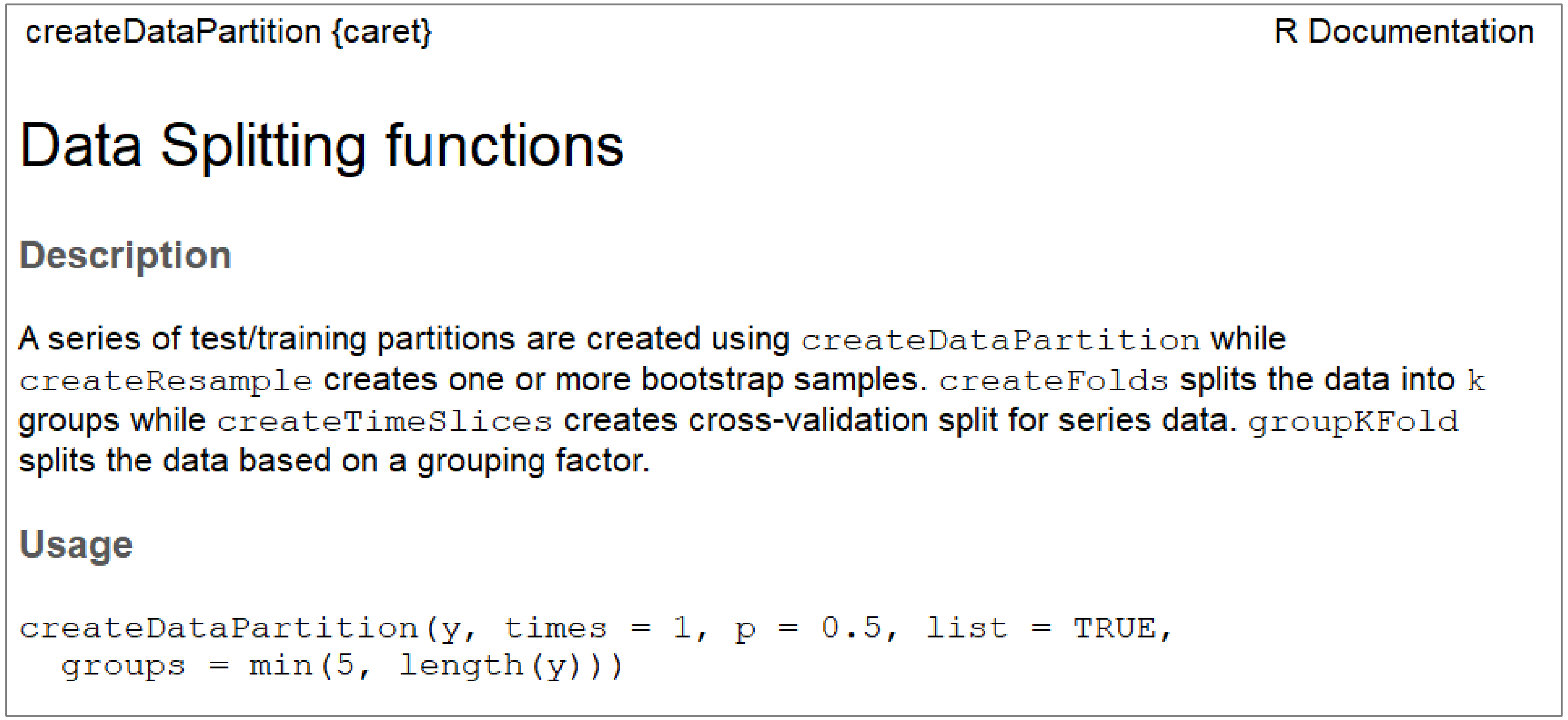
caret package

caret package(Classification And REgression Training) - many functions for building predictive models

Tools for: data splitting, pre-processing, feature selection, tuning and supervised – unsupervised learning algorithms, etc.

https://topepo.github.io/caret/index.html

createDataPartition()



createDataPartition {caret} R Documentation

Data Splitting functions

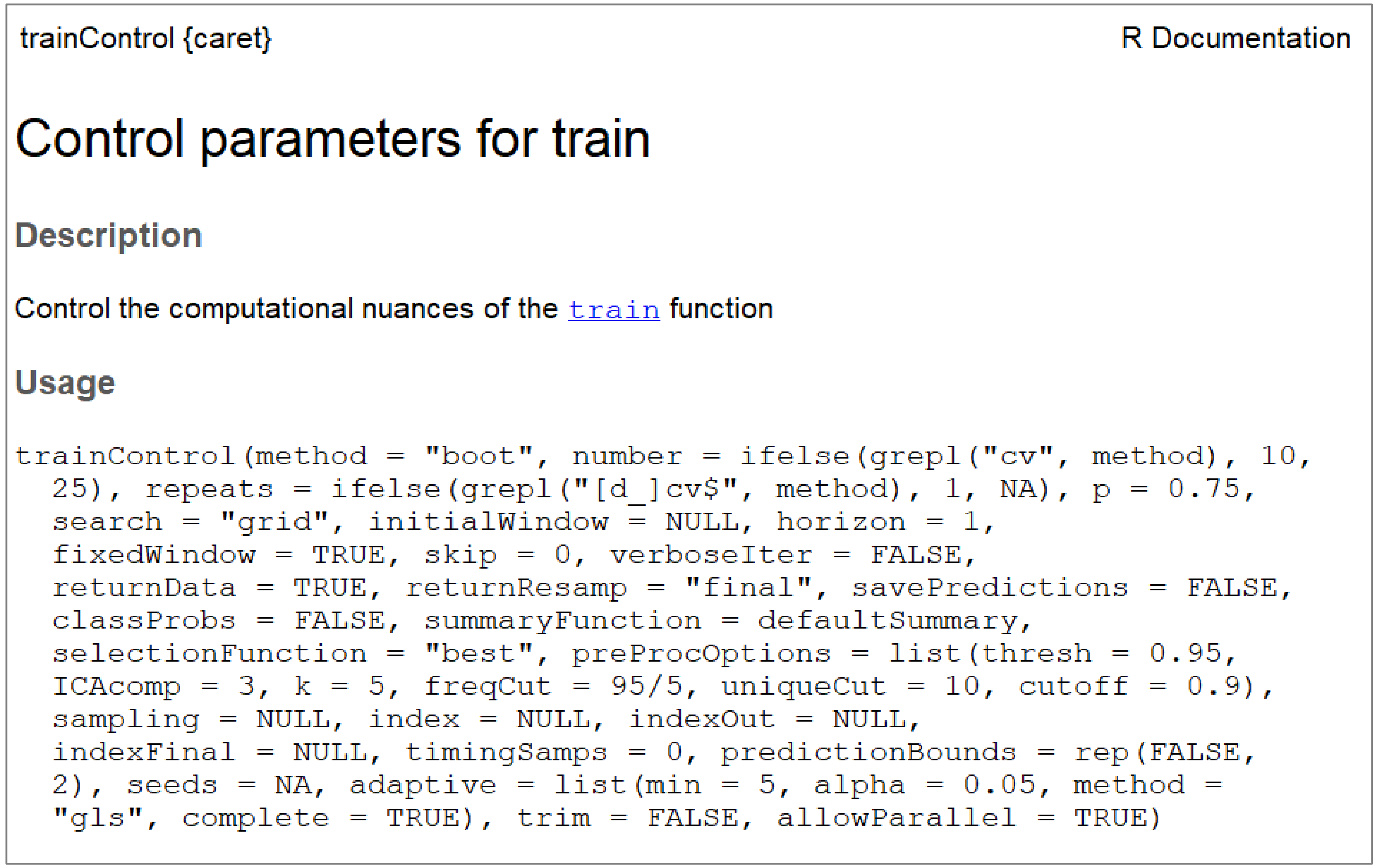
Description

A series of test/training partitions are created using `createDataPartition` while `createResample` creates one or more bootstrap samples. `createFolds` splits the data into `k` groups while `createTimeSlices` creates cross-validation split for series data. `groupKFold` splits the data based on a grouping factor.

Usage

```
createDataPartition(y, times = 1, p = 0.5, list = TRUE,
  groups = min(5, length(y)))
```

trainControl()



trainControl {caret} R Documentation

Control parameters for train

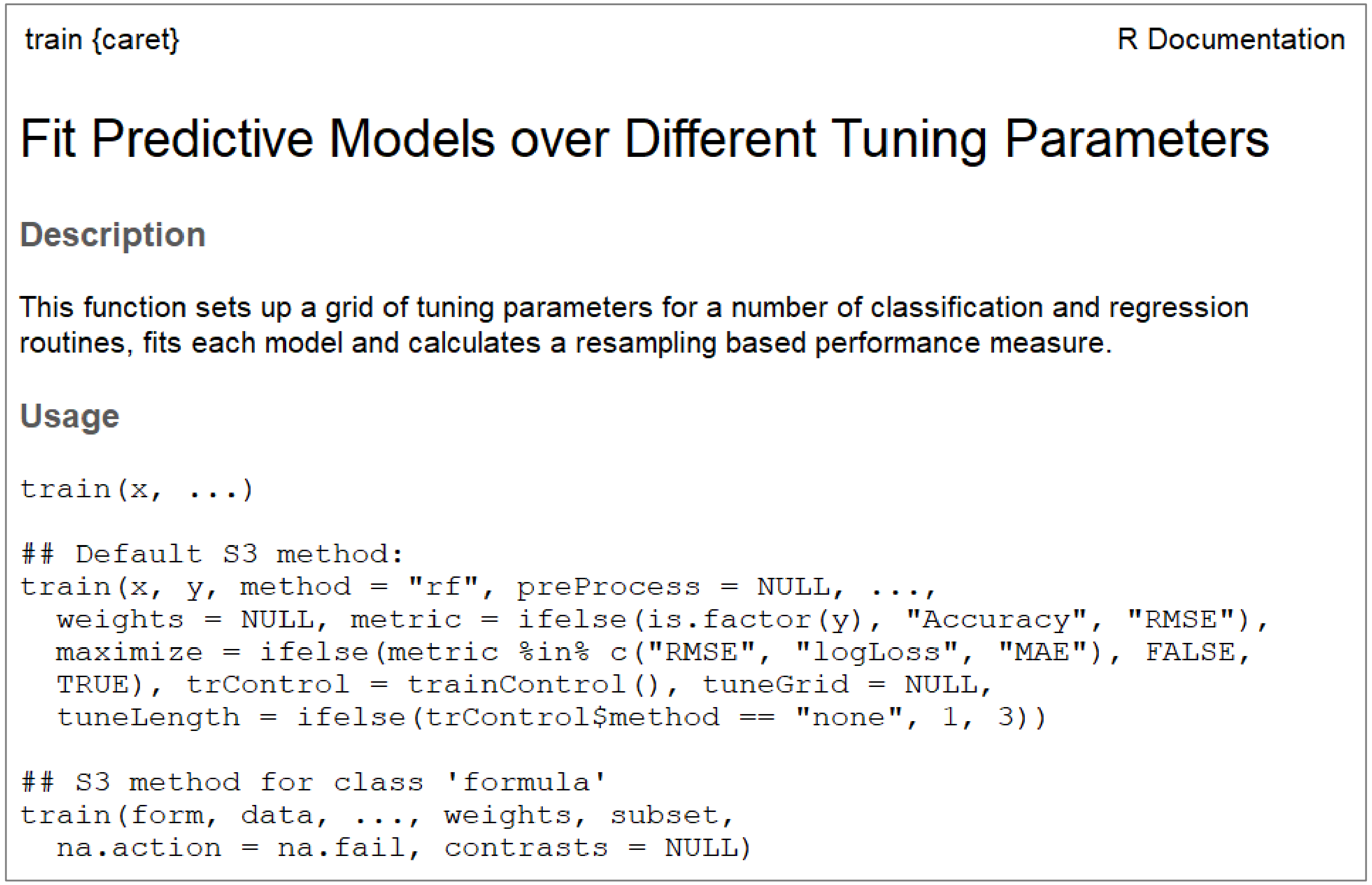
Description

Control the computational nuances of the train function

Usage

```
trainControl(method = "boot", number = ifelse(grepl("cv", method), 10,
  25), repeats = ifelse(grepl("[d_]cv$", method), 1, NA), p = 0.75,
  search = "grid", initialWindow = NULL, horizon = 1,
  fixedWindow = TRUE, skip = 0, verboseIter = FALSE,
  returnData = TRUE, returnResamp = "final", savePredictions = FALSE,
  classProbs = FALSE, summaryFunction = defaultSummary,
  selectionFunction = "best", preProcOptions = list(thresh = 0.95,
  ICAcomp = 3, k = 5, freqCut = 95/5, uniqueCut = 10, cutoff = 0.9),
  sampling = NULL, index = NULL, indexOut = NULL,
  indexFinal = NULL, timingSamps = 0, predictionBounds = rep(FALSE,
  2), seeds = NA, adaptive = list(min = 5, alpha = 0.05, method =
  "gls", complete = TRUE), trim = FALSE, allowParallel = TRUE)
```

train()



train {caret}   R Documentation

Fit Predictive Models over Different Tuning Parameters

Description

This function sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure.

Usage

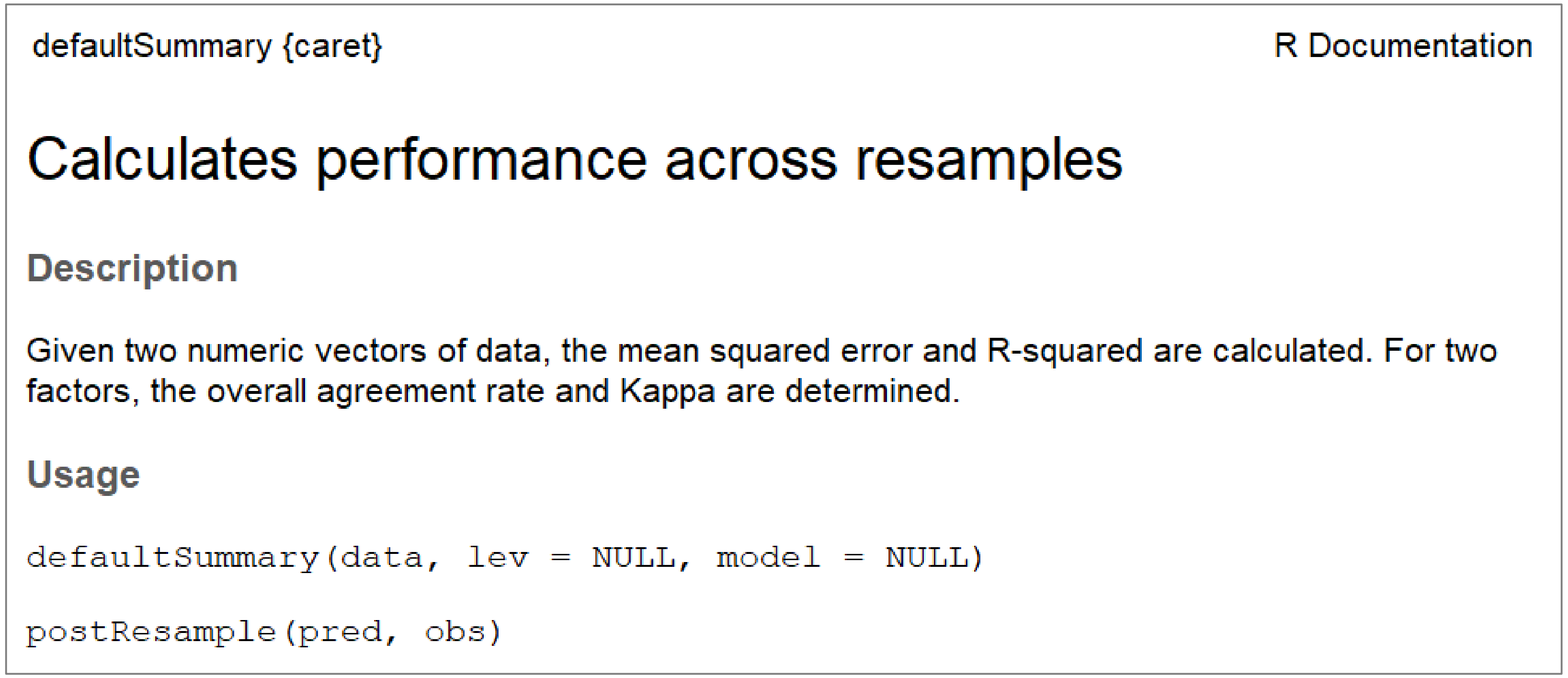
```
train(x, ...)

## Default S3 method:
train(x, y, method = "rf", preProcess = NULL, ...,
  weights = NULL, metric = ifelse(is.factor(y), "Accuracy", "RMSE"),
  maximize = ifelse(metric %in% c("RMSE", "logLoss", "MAE"), FALSE,
  TRUE), trControl = trainControl(), tuneGrid = NULL,
  tuneLength = ifelse(trControl$method == "none", 1, 3))

## S3 method for class 'formula'
train(form, data, ..., weights, subset,
  na.action = na.fail, contrasts = NULL)
```

https://topepo.github.io/caret/train-models-by-tag.html

postResample()


defaultSummary {caret} R Documentation

Calculates performance across resamples

Description

Given two numeric vectors of data, the mean squared error and R-squared are calculated. For two factors, the overall agreement rate and Kappa are determined.

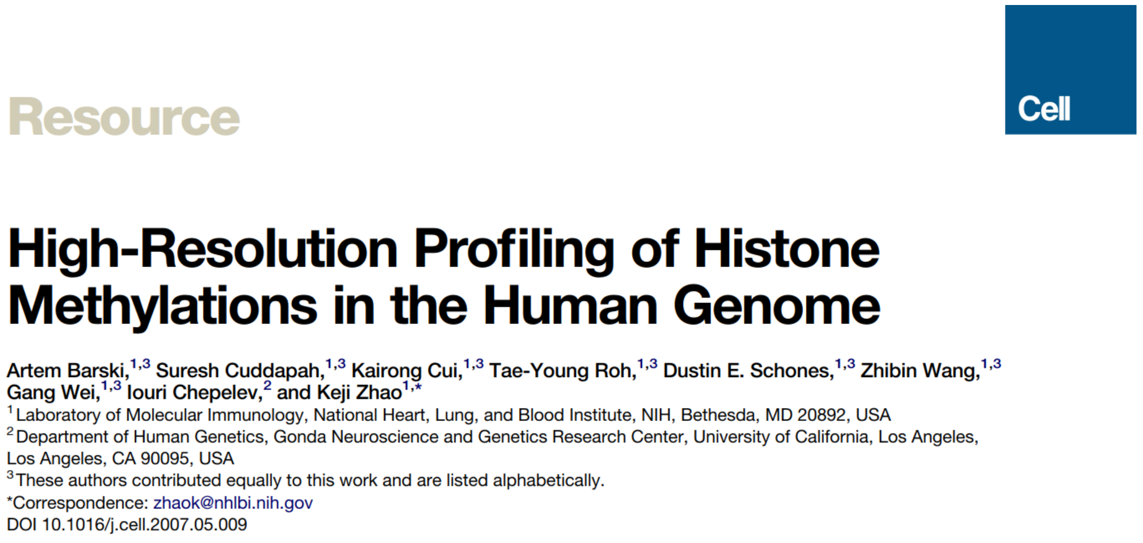
Usage

```
defaultSummary(data, lev = NULL, model = NULL)

postResample(pred, obs)
```

Exercise 4 - Training a regression tree using the caret package

Using the rpart method implemented in caret package, train a regression tree to predict gene expression from histone modifications. Report the accuracy of your model (hcp.model)



Resource

Cell

High-Resolution Profiling of Histone Methylations in the Human Genome

Artem Barski,[1,3] Suresh Cuddapah,[1,3] Kairong Cui,[1,3] Tae-Young Roh,[1,3] Dustin E. Schones,[1,3] Zhibin Wang,[1,3] Gang Wei,[1,3] Iouri Chepelev,[2] and Keji Zhao[1,*]

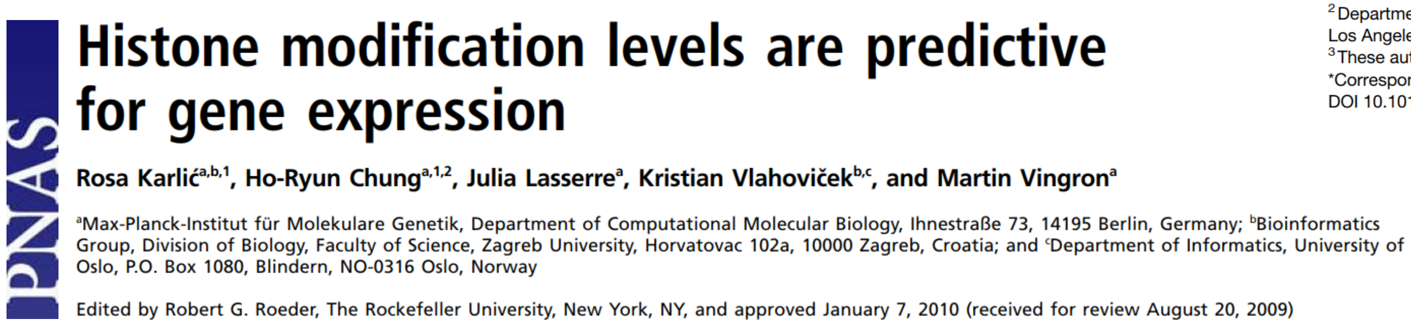
[1] Laboratory of Molecular Immunology, National Heart, Lung, and Blood Institute, NIH, Bethesda, MD 20892, USA

[2] Department of Human Genetics, Gonda Neuroscience and Genetics Research Center, University of California, Los Angeles, Los Angeles, CA 90095, USA

[3] These authors contributed equally to this work and are listed alphabetically.

*Correspondence: zhaok@nhlbi.nih.gov

DOI 10.1016/j.cell.2007.05.009



PNAS

Histone modification levels are predictive for gene expression

Rosa Karlić[a,b,1], Ho-Ryun Chung[a,1,2], Julia Lasserre[a], Kristian Vlahoviček[b,c], and Martin Vingron[a]

[a]Max-Planck-Institut für Molekulare Genetik, Department of Computational Molecular Biology, Ihnestraße 73, 14195 Berlin, Germany; [b]Bioinformatics Group, Division of Biology, Faculty of Science, Zagreb University, Horvatovac 102a, 10000 Zagreb, Croatia; and [c]Department of Informatics, University of Oslo, P.O. Box 1080, Blindern, NO-0316 Oslo, Norway

Edited by Robert G. Roeder, The Rockefeller University, New York, NY, and approved January 7, 2010 (received for review August 20, 2009)

Missing predictor values

- Discard any observation with some missing values
- Impute the missing values (ex. mean over the non-missing observations)
- Two additional approaches for tree-based data:
  - Make a new category for “missing”
  - Construction of surrogate variables.
    - *When considering a predictor for a split, we use only the observations for which that predictor is not missing.*
    - *Having chosen the best (primary) predictor and split point, we form a list of surrogate predictors and split points which best mimics the split of the training data achieved by the primary split*
    - *When sending observations down the tree we use the surrogate splits in order, if the primary splitting predictor is missing.*

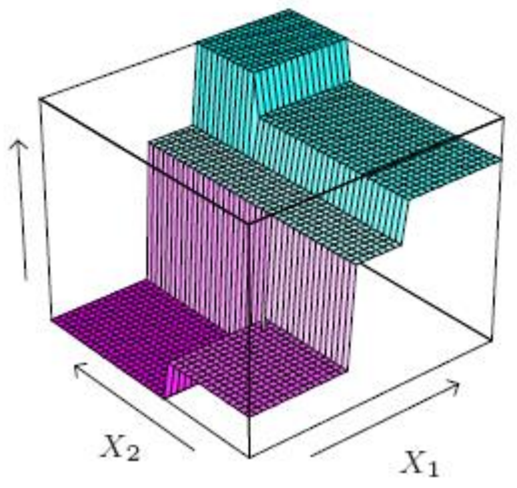
Exercise 5 – Visualize the regression tree and the predictions

Plot the binary tree representing the model and visualize the observed and predicted expression data. What do you notice?

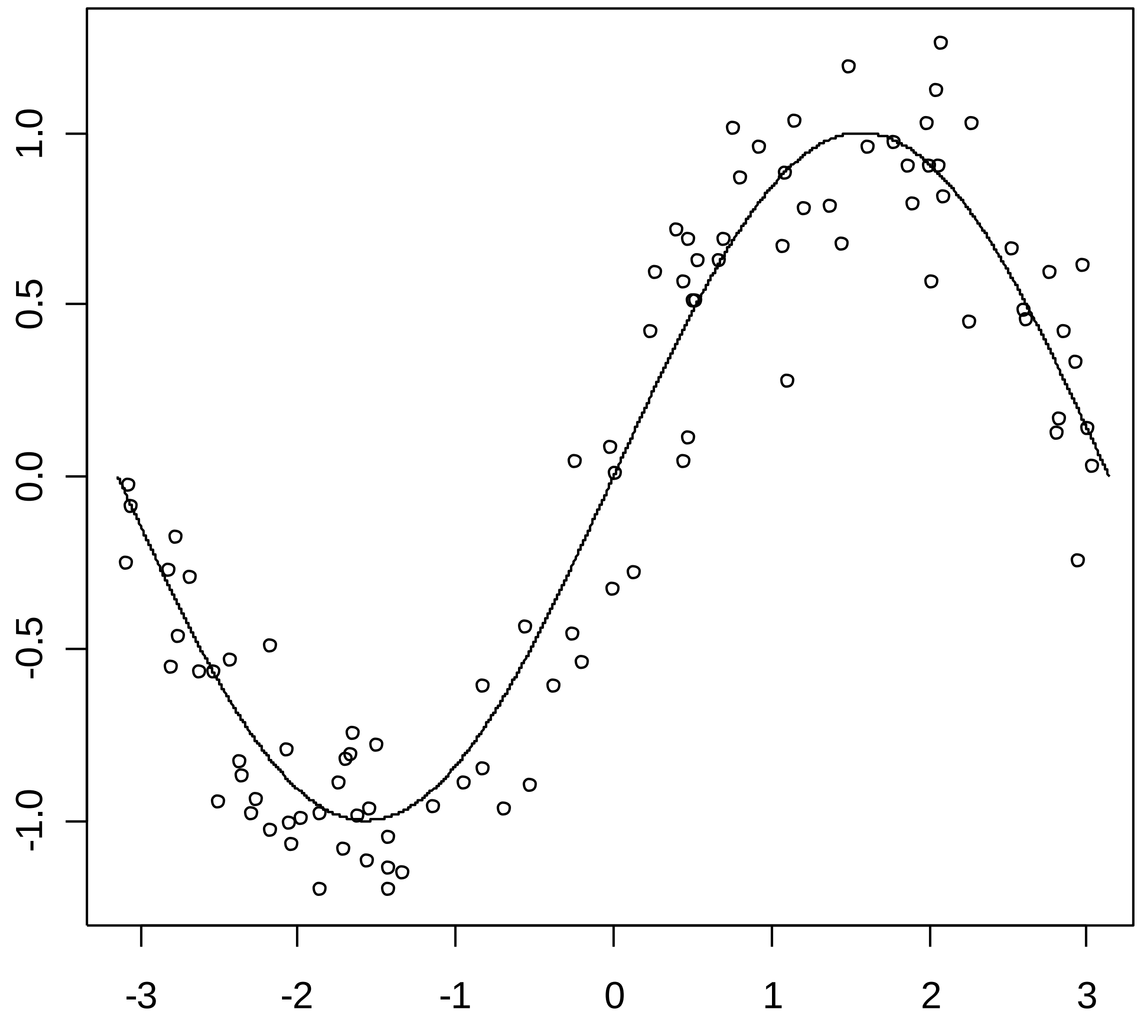
Lack of smoothness

Lack of smoothness of the prediction surface

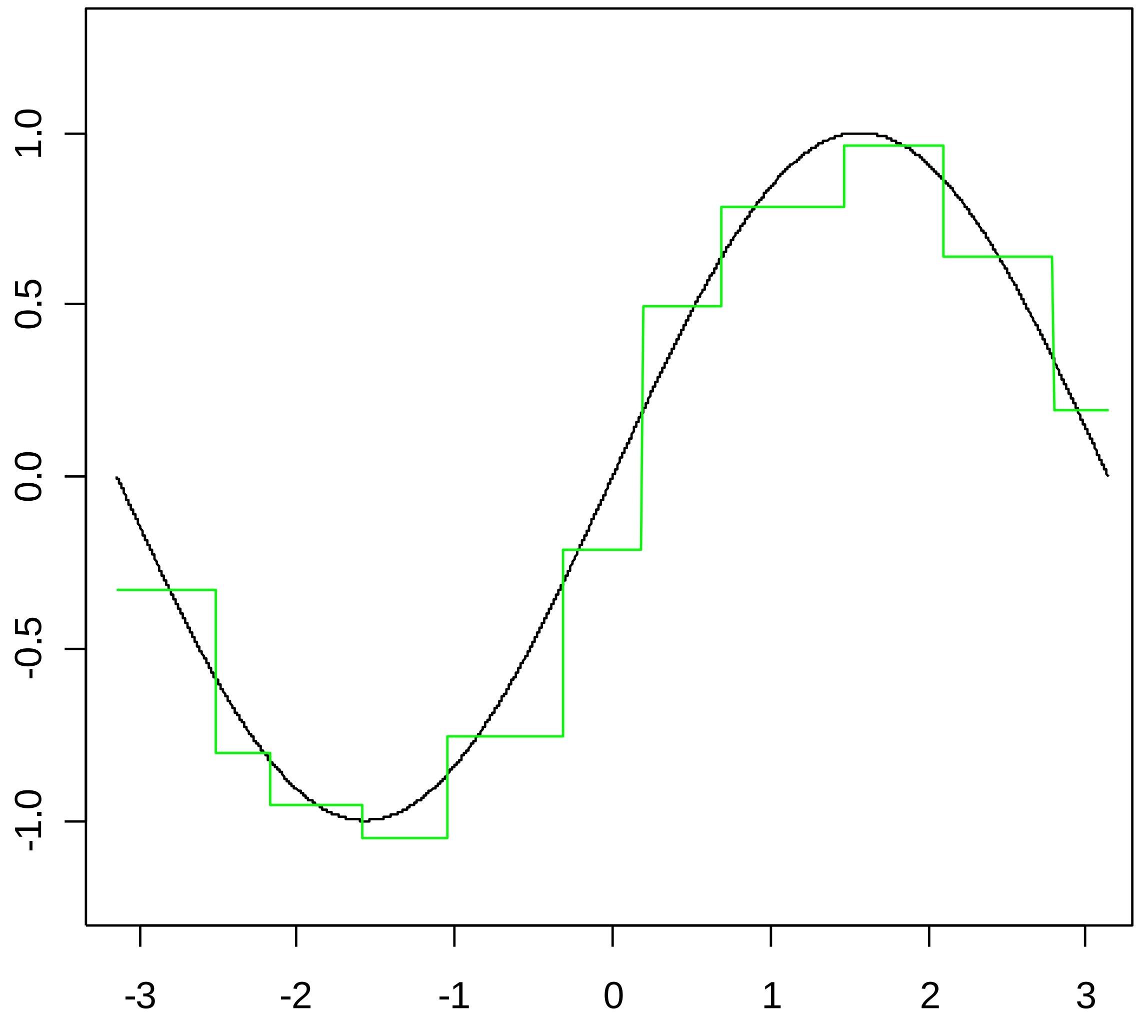
Can severly degrade performance in the regression setting



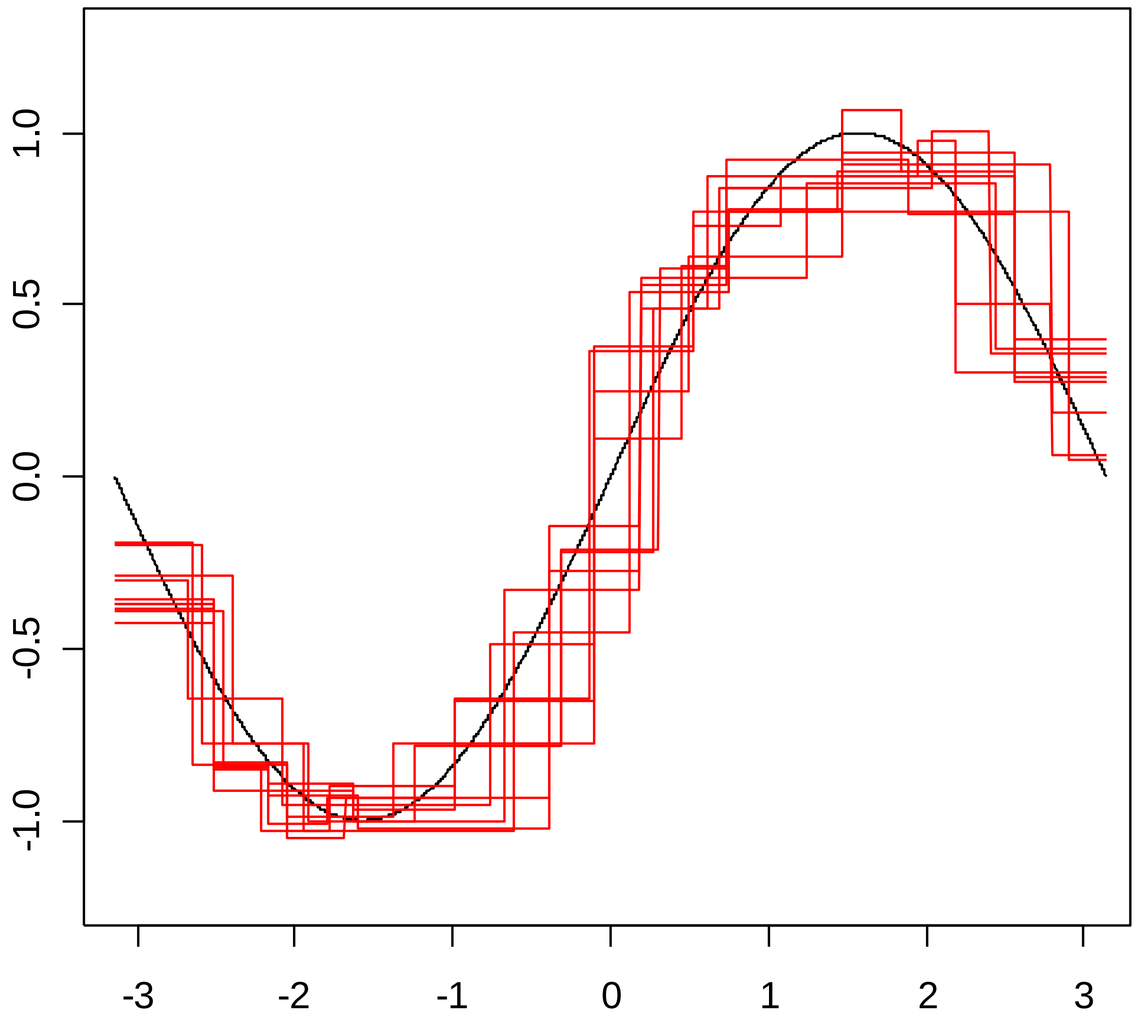
Data and Underlying Function



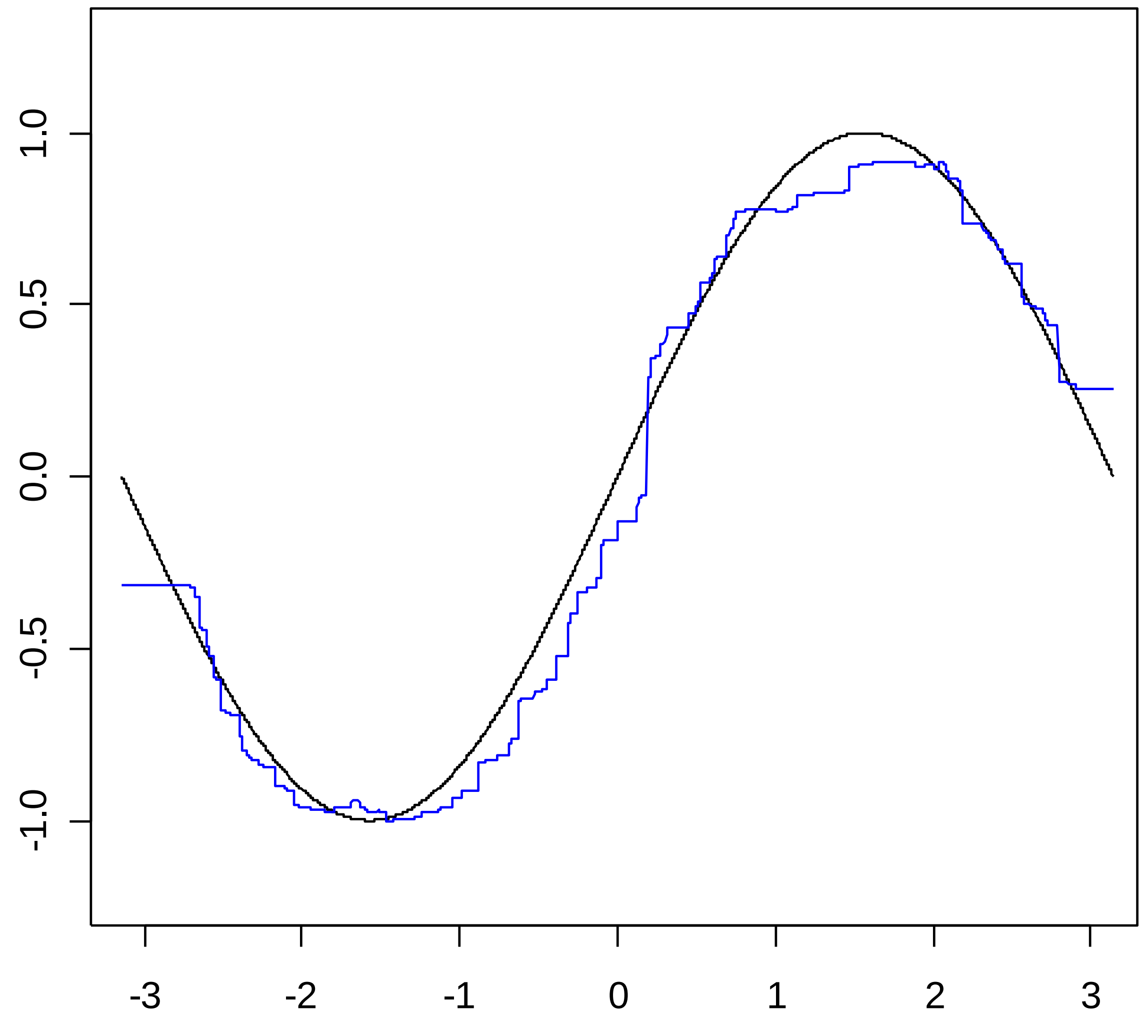
Single Regression Tree (all data)



10 Regression Trees (fit to boostrap samples)



Average of 100 Regression Trees (fit to bootstrap samples)



Decision trees

Advantages:

- Simple and easy to interpret

Disadvantages:

- Instable (high variance)
- Lack of smoothness

Ensemble methods

Combine multiple models to obtain better predictive performance

For example:

- Random forest
- Bagging
- Boosting

Random Forests

Grow a forest of **trees**. Grow each tree on an independent **bootstrap sample** from the training data. Bootstrap – sampling with replacement from the original data.

At each node:

1. Randomly select **m** variables out of all **M** possible variables (independently for each node).
2. Find the best split on the selected **m** variables.

Grow the trees to maximum depth.

Vote or average the trees to get predictions.

Bootstraped samples - we can get out-bag error estimates

Out-of-bag Data

For each tree in the forest, we select a bootstrap sample from the data.

The bootstrap sample is used to grow the tree.

The remaining data are said to be “out-of-bag” (about one-third of the cases).

The out-of-bag data serve as a test set for the tree grown on the bootstrap sample.

The out-of-bag error estimate

Think of a ***single case*** in the training set.

It will be out-of-bag in about one-third of the trees.

Each time it is out of bag, pass it down the tree and get a predicted class.

The RF prediction is the class that is chosen the most often. For each case, the RF prediction is either correct or incorrect.

- Average over the cases within each class to get a ***classwise*** out-of-bag error rate.
- Average over all cases to get an ***overall*** out-of-bag error rate.

The out-of-bag error Estimate

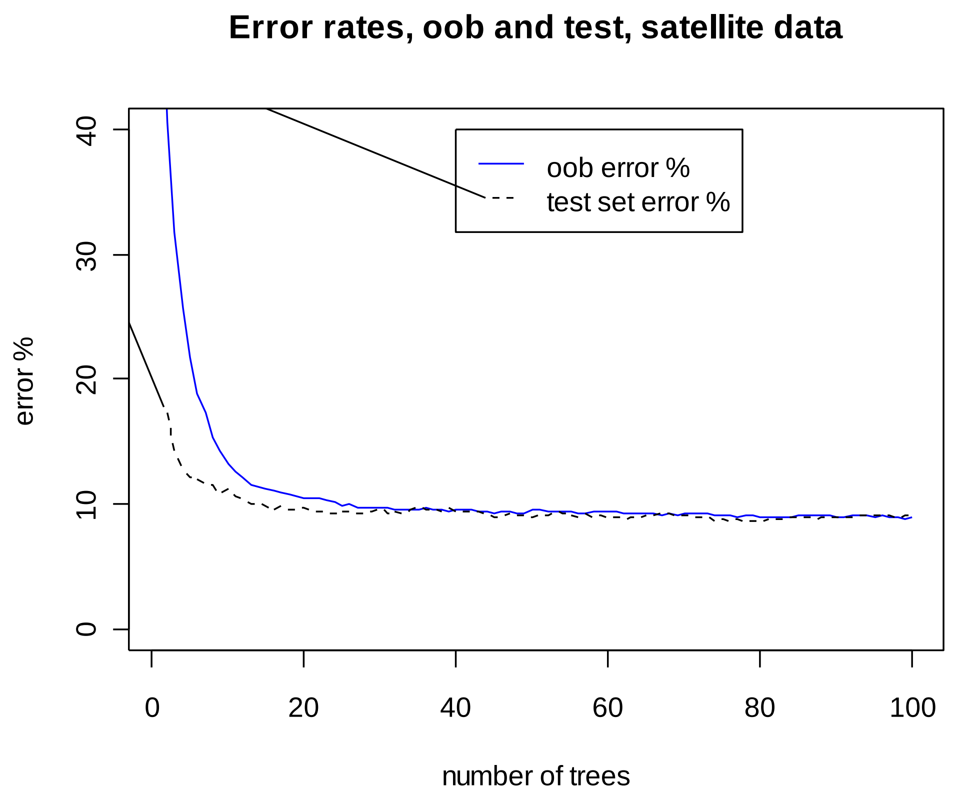
For example, suppose we fit 1000 trees, and a case is out-of-bag in 339 of them, of which:

283 say “class 1”

56 say “class 2”

The RF predictor is class 1.

The out-of-bag error rate (%) is the percentage of the time that the RF predictor is correct.



Training set: 4435 cases.
Test set: 2000 cases.
36 variables.
100 trees.

The out-of-bag error rate is larger at the beginning because there are not enough trees for good predictions.

Using out-of-bag Data to Choose **m**

The out-of-bag error rate is used to select **m**.

Here's how:

1. Start with **m** = √**M**.
2. Run a few trees, recording the out-of-bag error rate.
3. Increase **m**, decrease **m,** until you are reasonably confident you've found a value with minimum out-of-bag error rate.

Exercise 6 – Random Forest

Using the ranger method implemented in caret package, train a regression tree to predict gene expression from histone modifications. Report the accuracy of your model (hcp.model.rf)

Variable importance

In every tree grown in the forest, put down the oob cases and count the number of votes cast for the correct class.

Randomly permute the values of variable m in the oob cases and put these cases down the tree.

Subtract the number of votes for the correct class in the variable-m-permuted oob data from the number of votes for the correct class in the untouched oob data.

The average of this number over all trees in the forest is the raw importance score for variable m.

Variable importance for a single class 2 case

TREE	No permutation	Permute variable 1	…	Permute variable m
1	2	2	…	1
3	2	2	…	2
4	1	1	…	1
9	2	1	…	1
…	…	…	…	…
1000	2	2	…	1
% Error	10%	11%	…	35%

Standard error of variable importance

If the values of this score from tree to tree are independent, then the standard error can be computed by a standard computation.

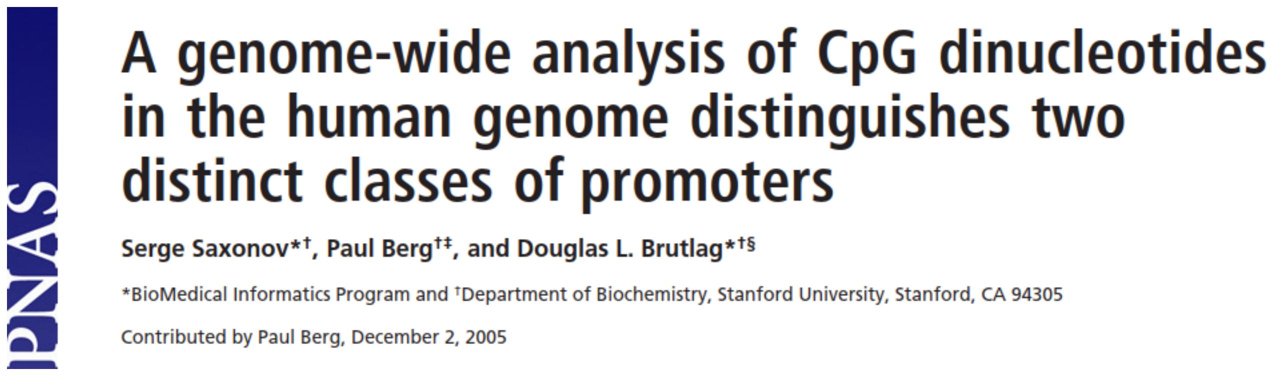
Divide the raw score by its standard error to get a z-score, and assign a significance level to the z-score assuming normality.

If the number of variables is very large, forests can be run once with all the variables, then run again using only the most important variables from the first run.

We can also calculate local variable importance.

Exercise 7 – variable importance of Random Forest models

Compare variable importance of random forest models used for predicting expression of genes with HCP or LCP promoters.



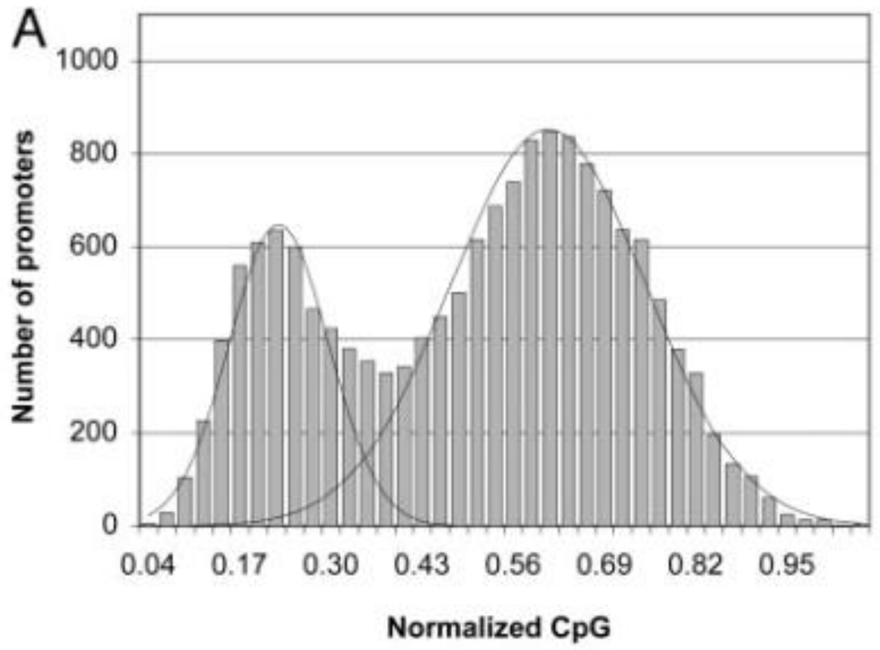
PNAS

A genome-wide analysis of CpG dinucleotides in the human genome distinguishes two distinct classes of promoters

Serge Saxonov*†, Paul Berg†‡, and Douglas L. Brutlag*†§

*BioMedical Informatics Program and †Department of Biochemistry, Stanford University, Stanford, CA 94305

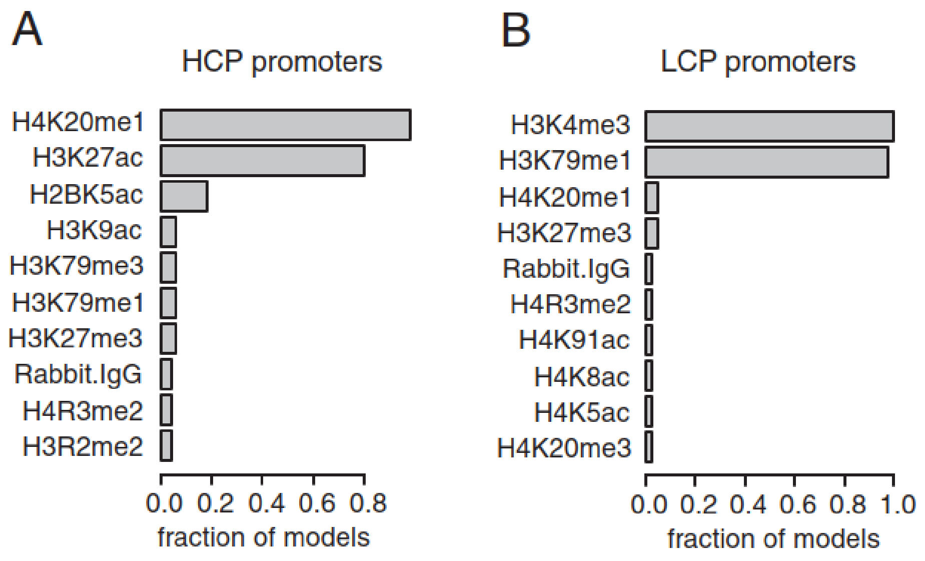
Contributed by Paul Berg, December 2, 2005



Saxonov et al., 2006, PNAS

Exercise 7 – variable importance of Random Forest models

Compare variable importance of random forest models used for predicting expression of genes with HCP or LCP promoters.



Karlić et al., 2010, PNAS

Proximities

Proximity of two items is the proportion of the time that they end up in the same node when they are simultaneously out-of-bag.

The proximities don’t just measure similarity of the covariates. Instead, they are based on the way the trees deal with the data.

Two cases that have quite different covariates might have high proximity if they differ on covariates that are ***not important*** in predicting class.

Two cases that have quite similar covariates might have small proximity if they differ on covariates that are ***important*** in predicting class.

Replacing Missing Values

Fast way: replace missing values for a given variable using the median of the non-missing values (or the most frequent, if categorical)

Better way (using proximities):

1. Start with the fast way.
2. Get proximities.
3. Replace missing values in case **n** by a weighted average of non-missing values, with weights proportional to the proximity between case **n** and the cases with the non-missing values.

Repeat steps 2 and 3 a few times (5 or 6).

Advantages

Excellent accuracy (captures also non-linear relationships).

Fast.

Does not overfit as we add more trees.

Handles
- thousands of variables
- many-valued categoricals
- extensive missing values
- badly unbalanced data sets.

Gives an internal unbiased estimate of test set error as trees are added to the ensemble

Disadvantages

Can’t represent our results in a tree structure

Variable importance measures can be misleading depending on the correlation of predictors or the scale (number of categories)

Similar approaches

Bagging – aggregate predictions of bootstrapped trees (without choosing random variables for training)

Boosting – grow trees sequentially to improve performance of previously collected trees

Boosting parameters:

- number of trees
- shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001.
- number of splits d in each tree, which controls the complexity
- of the boosted ensemble

Exercise 8 - Bagging

Using the "treebag" method implemented in caret package, train a regression tree to predict gene expression from histone modifications for genes with HCP promoters. Report the accuracy of your model (hcp.model.bag)

Exercise 9 - Boosting

Using the “bstTree" method implemented in caret package, train a regression tree to predict gene expression from histone modifications for genes with HCP promoters. Report the accuracy of your model (hcp.model.boost)

Exercise 10 – Compare accuracy of different models

Use the previously obtained Rsquared values (form postResample) to compare the accuracy of regression tree, random forest, bagging and boosting models.