# Classification And Regression Trees

*In R using* `rpart`

August 21, 2018

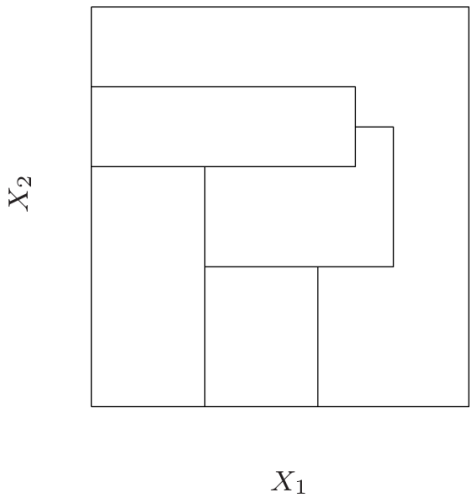
Torben Tvedebrink
tvede@math.aau.dk

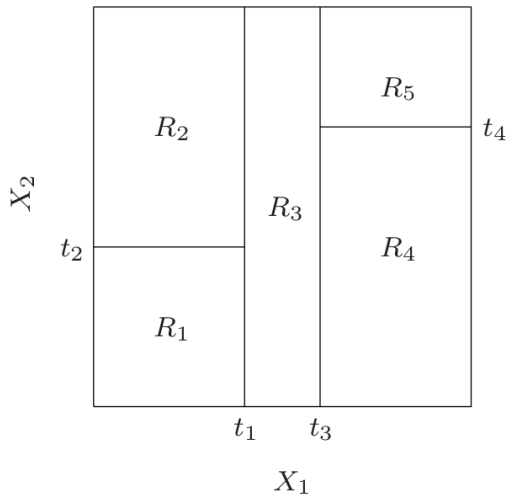
Data Science using R

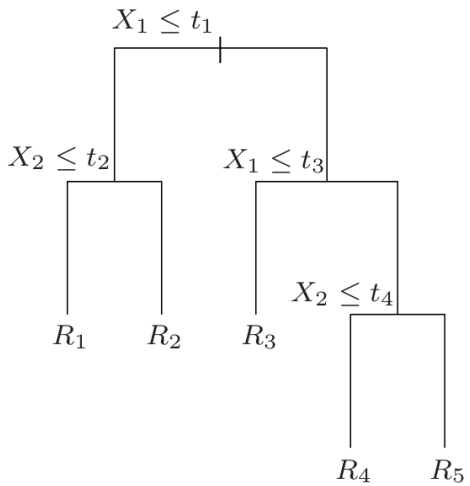
AALBORG UNIVERSITY
DENMARK

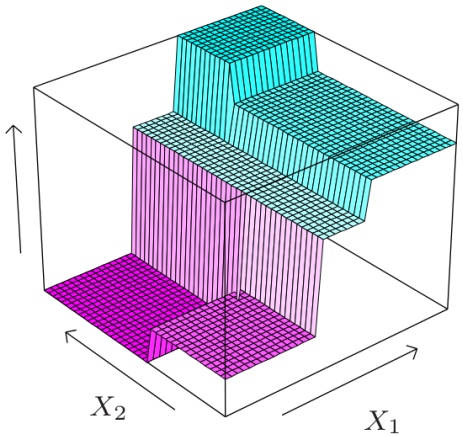
# CART: Classification And Regression Trees

Link: Introduction to `rpart`









Torben Tvedebrink
tvede@math.aau.dk

# CART: Regression



For regression the CART methodoloy fits a piecewise constant prediction for each region $R_j$,

$$\hat{Y}_{\text{CART}}(\boldsymbol{x}) = \sum_{j=1}^{R} \beta_j \mathbb{I}(\boldsymbol{x} \in R_j),$$

where $\beta_j$ is the constant level for region $R_j$.

Hence, the expression for $\hat{Y}$ can be determined if

a) the partition (i.e. the regions $R_1, \ldots, R_R$) are known

b) the estimated parameters $\beta_j$ are known

These are chosen such that they minimises the expected squared loss for future observations $(\boldsymbol{x}, y)$,

$$\mathbb{E}[(Y - \hat{Y})^2]$$


Torben Tvedebrink
tvede@math.aau.dk

# CART: Classification





Assume that $y \in \{0, 1\}$ and CART once again construcs a piecewise constant function

$$\hat{Y}_{\text{CART}}(\boldsymbol{x}) = \sum_{j=1}^{R} \beta_j \mathbb{I}(\boldsymbol{x} \in R_j),$$

where $\beta_j \in [0, 1]$. Standard classification uses

$$Y_{\text{CART}}(\boldsymbol{x}) = \begin{cases} 0, & \text{hvis } \hat{Y}_{\text{CART}} \leq 0.5 \\ 1, & \text{hvis } \hat{Y}_{\text{CART}} > 0.5 \end{cases}$$

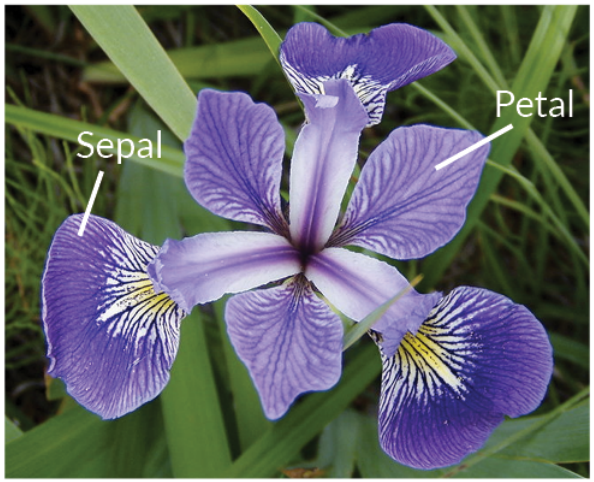
A good choice of $\hat{Y}_{\text{CART}}$ leads to a small mis-classification rate, $P(Y_{\text{CART}}(\boldsymbol{x}) \neq y)$.


Torben Tvedebrink
tvede@math.aau.dk

# Eksempel
## Iris data – three species








**Iris Versicolor**

**Iris Setosa**

**Iris Virginica**

```
> iris[c(1:2,51:52,101:102),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
51           7.0         3.2          4.7         1.4 versicolor
52           6.4         3.2          4.5         1.5 versicolor
101          6.3         3.3          6.0         2.5  virginica
102          5.8         2.7          5.1         1.9  virginica
```


Torben Tvedebrink
tvede@math.aau.dk

# Eksempel
## Iris data

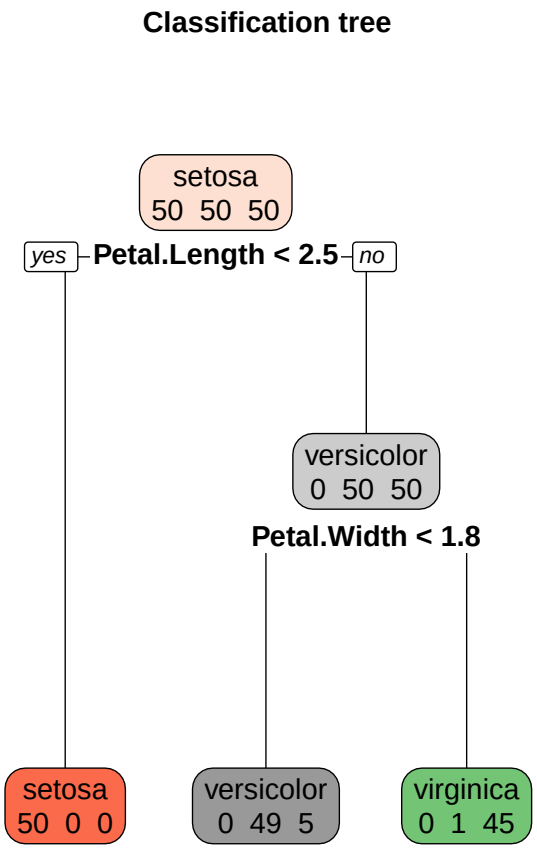


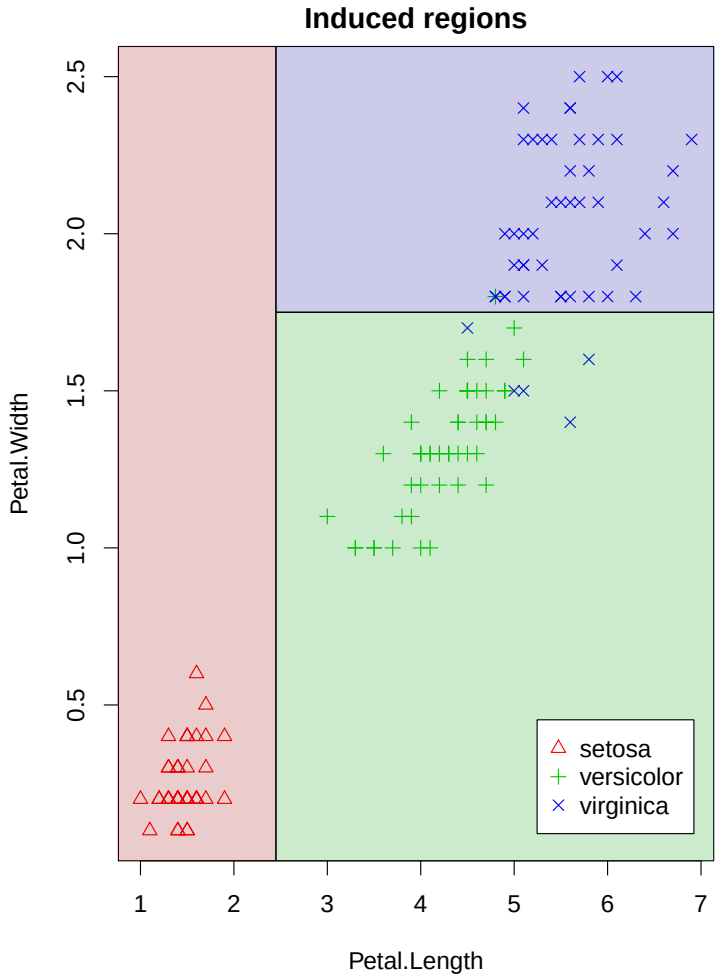
We can classify the species in the Iris dataset using CART classification.

```
library(rpart)

data(iris)

(cart.iris <- rpart(Species~.,data=iris))

n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 150 100 setosa (0.33 0.33 0.33)
  2) Petal.Length< 2.45 50   0 setosa (1.00 0.00 0.00) *
  3) Petal.Length>=2.45 100  50 versicolor (0.00 0.50 0.50)
    6) Petal.Width< 1.75 54   5 versicolor (0.00 0.91 0.09) *
    7) Petal.Width>=1.75 46   1 virginica (0.00 0.02 0.98) *
```


Torben Tvedebrink
tvede@math.aau.dk

# Example
## Iris data – Cont'd







Torben Tvedebrink
tvede@math.aau.dk

# Parameter estimation



From the model

$$\hat{Y}_{\text{CART}}(\boldsymbol{x}) = \sum_{j=1}^{R} \beta_j \mathbb{I}(\boldsymbol{x} \in R_j),$$

we have that when the partitions/regions $R_j$ are given, the MLE for $\beta_j$ is given by

$$\hat{\beta}_j = \frac{\sum_{i=1}^{n} y_i \mathbb{I}(\boldsymbol{x}_i \in R_j)}{\sum_{i=1}^{n} \mathbb{I}(\boldsymbol{x}_i \in R_j)} = \bar{y}_{R_j}.$$

where $\hat{\beta}_j$ for regression just is the average of the $y$s with $\boldsymbol{x} \in R_j$ and for classification the fraction of "$y = 1$"-samples.


Torben Tvedebrink
tvede@math.aau.dk

# Partitioning





Ideally we wants a partitioning which given the smallest expected loss (regression: sum of squares, classification: error rate).

The number of partitions is to vast, why an exhautive search is infeasible.

Hence, we use a greedy algorithm to search for partitions with good splits.

Note! The `r` in `rpart` stands for *recursive*. Hence, what applies to the root is used recursively down the tree.


Torben Tvedebrink
tvede@math.aau.dk

# Method to generate splits



In the training data we have $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})$ is $p$-dimensional.

For a numeric predictor vector $\boldsymbol{x}$ we search for the partition:

1. Start by $R_1 = \mathbb{R}^p$
2. Given $R_1, \ldots, R_r$, split each $R_j$ into $R_{j_1}$ and $R_{j_2}$ where

$$R_{j_1} = \{\boldsymbol{x} \in \mathbb{R}^p : \boldsymbol{x} \in R_j \text{ and } x_k \leq c\}$$
$$R_{j_2} = \{\boldsymbol{x} \in \mathbb{R}^p : \boldsymbol{x} \in R_j \text{ and } x_k > c\},$$

   and the variable $x_k$ with splitting points $c$ is chosen such

$$\arg\min_{k,c} \min_{\beta_1,\beta_2} \left( \sum_{i:\boldsymbol{x}_i \in R_{j_1}} (y_i - \beta_1)^2 + \sum_{i:\boldsymbol{x}_i \in R_{j_2}} (y_i - \beta_2)^2 \right)$$

   Let $R_{1_1}, R_{1_2}, \ldots, R_{r_1}, R_{r_2}$ be new partitions.
3. Repeat step 2. $d$ times to get a tree of depth $d$.


Torben Tvedebrink
tvede@math.aau.dk

# Model complexity






What size of tree is optimal?

We can grow the tree untill each observations has its own leaf (terminal node). This gives an error rate of null, but not very enlightning!.

Hence, stop before that, but when?


Torben Tvedebrink
tvede@math.aau.dk

# Example
## Pima indians



Female descendents from the Pima indians above 21 years of age and living near Phonix, Arizona, was included in a study. Each female was tested for diabetes according to WHO's criteria.

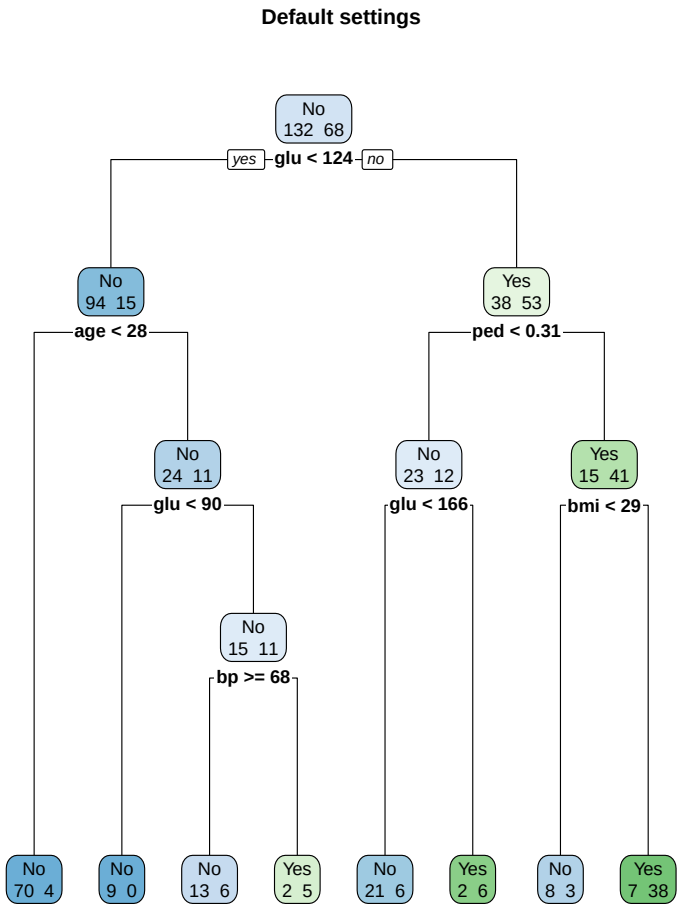
The variables in the data includes apart from diabetest status (`type`), information on

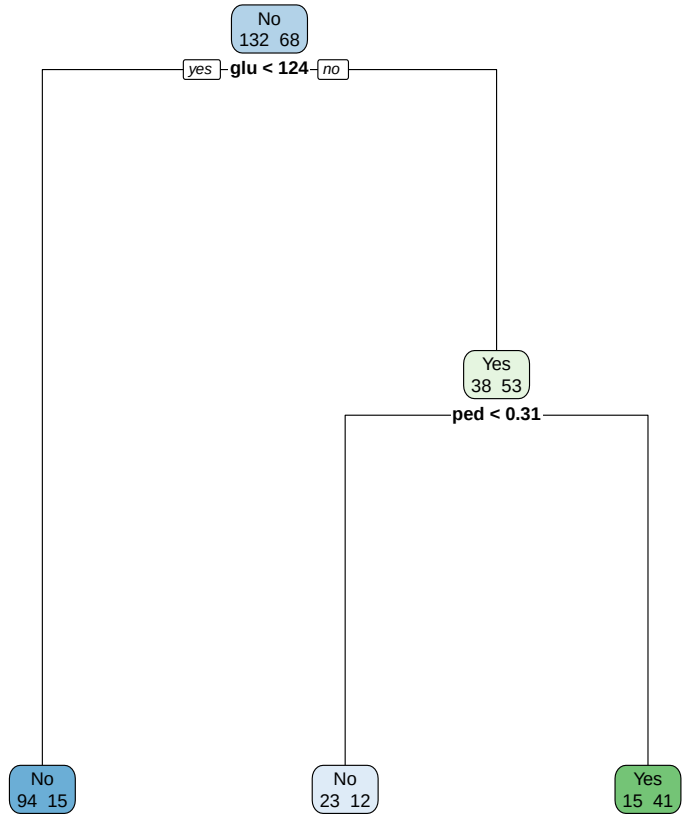
- number of pregnancies (`npreg`),
- plasma glucose concentration (`glu`)
- blood pressure (`bp`),
- triceps skin fold thickness (mm) (`skin`),
- BMI (`bmi`),
- diabetes pedigree function (`ped`) and
- age (`age`).


Torben Tvedebrink
tvede@math.aau.dk

# Two different trees

## Pima indians – Cont'd









Torben Tvedebrink
tvede@math.aau.dk

# Bias vs. variance



Why did I choose `rpart.control(cp=0.11)` in the analysis of the Pima indians? This *tuning parameter* decides the size of the tree (its complexity).

The larger the tree, the less bias but also a higher variance for the test data. Conversely, smaller trees gives larger bias, but little variance for test data.

In general, a bigger tree gives a better prediction for *training data*. However, an increased model complexity may result in a the model too specific for the training data (overfitting!), which makes it less applicable for test data and prediction for new data. It has a poor *generalisation* ability.




Torben Tvedebrink
tvede@math.aau.dk

# Choosing the *optimal* tree
## Tuning parameter $\alpha$



We wants to search for the *optimal* tree $T^*$, that minimises the *true* test error, $\text{Error}_{\text{Test}}$. This quantity is unknwon, but may be approximated using cross-validation.

The estimate/approximation is used to identify $T^*$, such that

$$T^* = \arg\min_T \text{Error}_{\text{Test}}(T)$$




Torben Tvedebrink
tvede@math.aau.dk

# Choosing the *optimal* tree
## Tuning parameter $\alpha$



We wants to search for the *optimal* tree $T^*$, that minimises the *true* test error, $\text{Error}_{\text{Test}}$. This quantity is unknwon, but may be approximated using cross-validation.

The estimate/approximation is used to identify $T^*$, such that

$$T^* = \arg\min_T \text{Error}_{\text{Test}}(T)$$

This, however, would require an exhaustive search over all possible trees $T$ – which obviously is infeasible.

Using a tuning parameter $\alpha$ the problem can be translated into a one-dimensional problem.


Torben Tvedebrink
tvede@math.aau.dk

# Pruning

The tuning parameter $\alpha$ penalises large trees,

$$\text{Error}_{\text{Train}}(T) + \alpha|T|, \tag{1}$$

where $|T|$ is the number of leafs in the tree.




Torben Tvedebrink
tvede@math.aau.dk

# Pruning

The tuning parameter $\alpha$ penalises large trees,

$$\text{Error}_{\text{Train}}(T) + \alpha|T|, \tag{1}$$

where $|T|$ is the number of leafs in the tree.

Two approaches:

- Grow the tree untill (1) increases.
- Grow a full tree and prune it untill (1) increases.






Torben Tvedebrink
tvede@math.aau.dk

# Selecting $\alpha$

What value of $\alpha$ should be used? Given $\alpha \in \mathbb{R}_+$, let $T_\alpha$ be the tree that minimises

$$T_\alpha = \arg\min_T \text{Error}_{\text{Train}}(T) + \alpha|T|$$




Torben Tvedebrink
tvede@math.aau.dk

# Selecting $\alpha$



What value of $\alpha$ should be used? Given $\alpha \in \mathbb{R}_+$, let $T_\alpha$ be the tree that minimises

$$T_\alpha = \arg\min_T \text{Error}_{\text{Train}}(T) + \alpha|T|$$

We wants $\alpha^*$ such that the resulting tree has the minimal test error

$$T_{\alpha^*} = \arg\min_{T_\alpha,\ \alpha\in\mathbb{R}_+} \hat{\text{Error}}_{\text{Test}}(T_\alpha),$$

where $\hat{\text{Error}}_{\text{Test}}$ is the estimate of the test error.


Torben Tvedebrink
tvede@math.aau.dk

# Selecting $\alpha$
Cont'd



We may plot the generalisation error $\hat{\text{Error}}_{\text{Test}}$ for the optimal tree using the criterion

$$\text{Error}_{\text{Train}}(T) + \alpha|T|$$

as a function of $\alpha$.

It holds that $T_\alpha$ is constant in intervals $I_1 = [0, \alpha_1]$, $I_2 = (\alpha_1, \alpha_2], \ldots, I_m = (\alpha_{m-1}, \infty]$. Hence, all values $\alpha' \in I_j$ gives the same tree, i.e. $\alpha_j$, $T_{\alpha'} \equiv T_{\alpha_j}$

Note, $T_0$ og $T_\infty$ are special cases – $T_0$ receives no penalty for its size (the full tree), $T_\infty$ gives the empty tree $T_\emptyset$.


Torben Tvedebrink
tvede@math.aau.dk

# How in `rpart`



To decide on $\alpha$, in `rpart` we use `printcp` or `plotcp`.

These functions use a rewritten version of the above:

$$\begin{aligned}\frac{\text{Error}_\alpha(T)}{\text{Error}_\infty(T)} &= \frac{\text{Error}(T) + \alpha|T|}{\text{Error}(T_\emptyset)} \\ &= \frac{\text{Error}(T)}{\text{Error}(T_\emptyset)} + \frac{\alpha}{\text{Error}(T_\emptyset)}|T| \\ &= \texttt{rel error} + \texttt{cp}|T|,\end{aligned}$$

where the error is relative to $T_\infty = T_\emptyset$ – i.e. the 'total' variance as we don't have any splits in $T_\infty$

The variable `cp` is short for 'complexity parameter'.


Torben Tvedebrink
tvede@math.aau.dk

# Choise of cp

There are (at least) two criteria to select $\alpha^*$ that decides the complexity of $T_{\alpha^*}$:

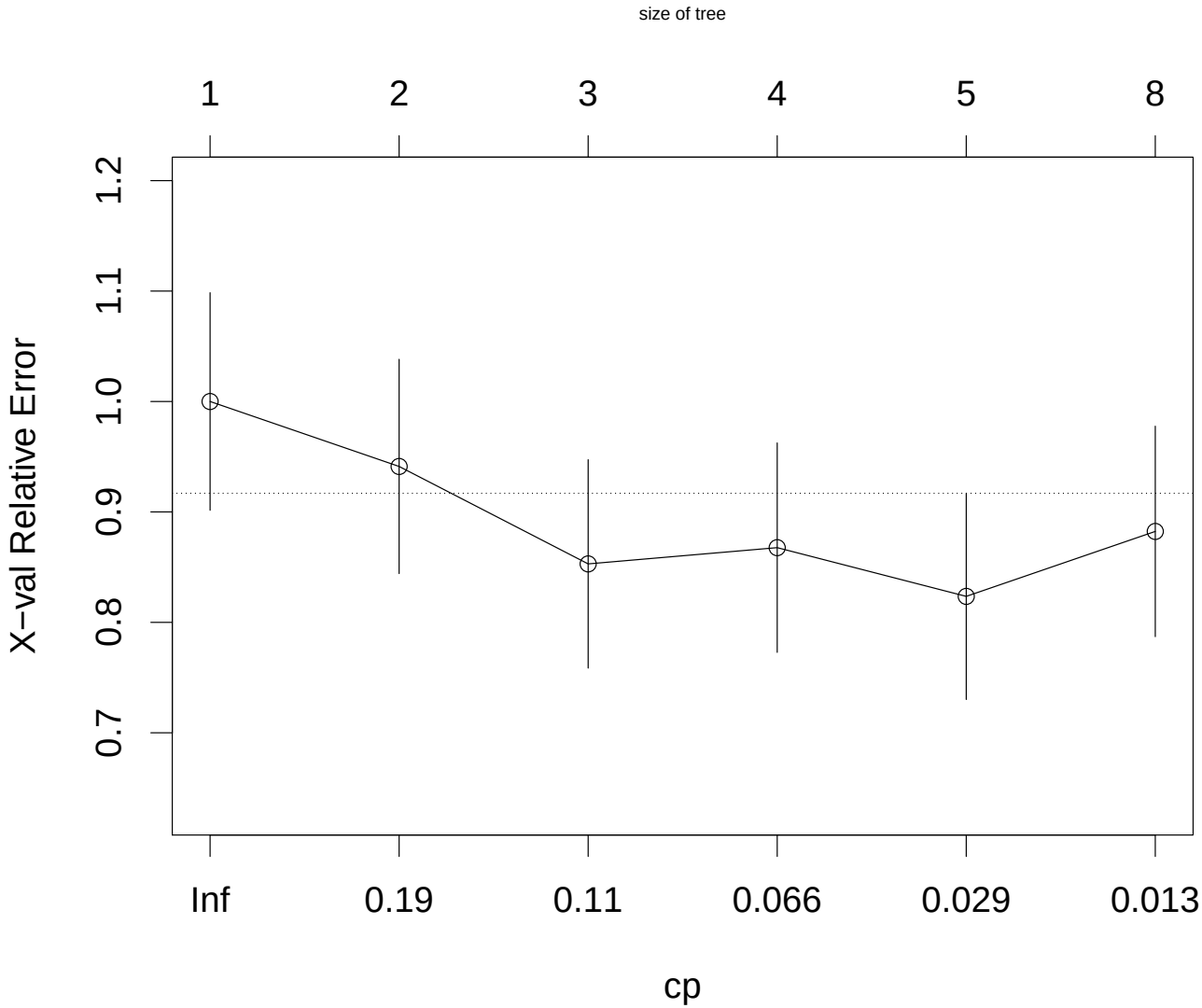
1. Choose cp where `xerror` (CV estimate of `rel error`) is smallest,
2. Choose cp giving `xerror` within one standard deviation of the smallest `xerorr`.

In the `plotcp`-plot the dotted line shows `xerror+xstd` relative to the cp-value with smallest `xerror`.

Note! `xerror` and `xstd` changes with the CV and is recomputed for each run of `rpart`.

In practice we use 2. since this gives the more parsimonious model (and we consider models within one standard deviation as equally good).

# Eksempel
## Pima indians – Cont'd







Torben Tvedebrink
tvede@math.aau.dk

# Eksempel
## Pima indianere – Cont'd

```
set.seed(13454)
pima.cp <- rpart(type~.,data=Pima.tr,cp=0.012)
printcp(pima.cp)

Classification tree:
rpart(formula = type ~ ., data = Pima.tr, cp = 0.012)

Variables actually used in tree construction:
[1] age bmi bp  glu ped

Root node error: 68/200 = 0.34

n= 200

        CP nsplit rel error  xerror     xstd
1 0.220588      0   1.00000 1.00000 0.098518
2 0.161765      1   0.77941 0.97059 0.097791
3 0.073529      2   0.61765 0.79412 0.092331
4 0.058824      3   0.54412 0.77941 0.091785
5 0.014706      4   0.48529 0.69118 0.088180
6 0.012000      7   0.44118 0.77941 0.091785
```

# Surrogates



A nice feature of the CART methodoloty are the so called *surrogates*. These are variables in the data that are not choosen as primary splitting variables, but assemples the splitting properties of the primary split.

They are in particularly important when *missing* observations exists in the primary split variables.


Torben Tvedebrink
tvede@math.aau.dk