

# Introduction to classification and regression trees, random forests and model-based recursive partitioning in R

Marjolein Fokkema

Department of Methods & Statistics  
Institute of Psychology  
Leiden University

# Course Overview

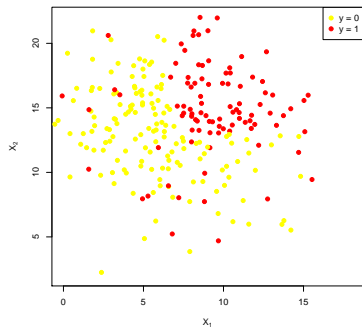
- ▶ Day 1: Single trees
  - ▶ Introduction to recursive partitioning methods (CART)
  - ▶ Unbiased recursive partitioning
  - ▶ Model-based recursive partitioning
  - ▶ Trees for clustered / multilevel data
- ▶ Day 2: Ensemble methods
  - ▶ Bagging, Boosting and Random Forests
  - ▶ Interpretation: Variable importance
  - ▶ Tuning parameters
  - ▶ Requests?

# Recursive partitioning

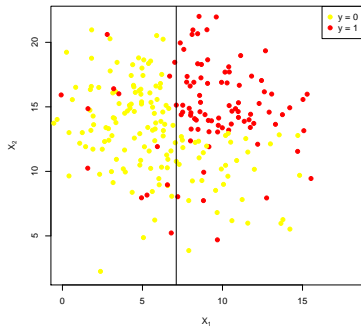
aim: predict  $Y$  from  $X_1, \dots, X_p$

how: find subgroups, characterized by values of  $X_1, \dots, X_p$ , that are most similar in terms of outcome  $Y$  (within each subgroup)

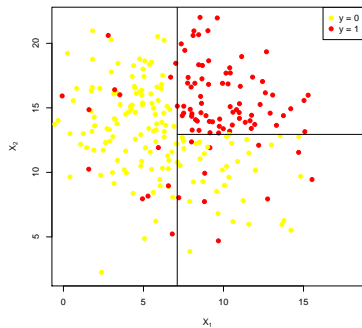
# Categorical outcome $Y$



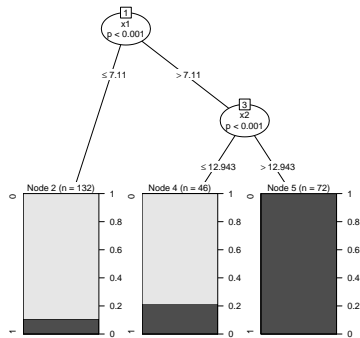
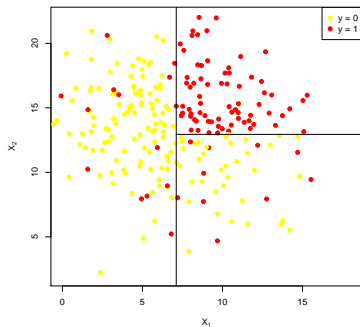
# Categorical outcome Y



# Categorical outcome $Y$



# Categorical outcome Y



# Classification and regression trees

aim: predict  $Y$  from  $X_1, \dots, X_p$

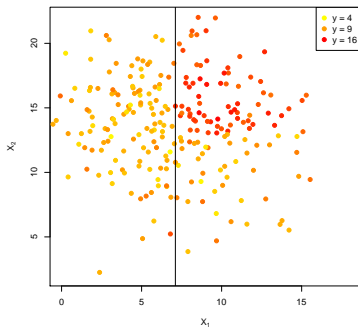
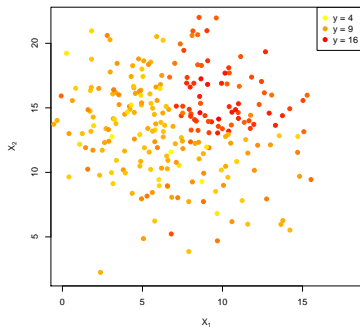
how: find subgroups, characterized by values of  $X_1, \dots, X_p$ , that are most similar in terms of  $Y$  (within each subgroup)

note: a partition can theoretically take any shape. To turn the partitioning into a feasible task, (most) RPMs:

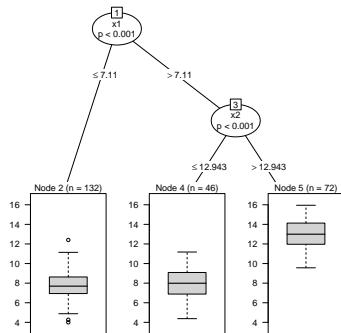
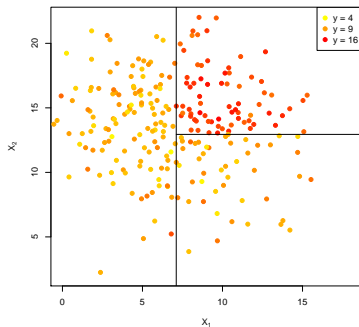
- ▶ split the data (variable space) recursively
- ▶ using one variable per split (rectangular areas)
- ▶ create binary splits only



# Continuous Y



# Continuous Y



# Questions

1. How is the tree built?
2. How can we make predictions from the tree?

# Questions

1. How is the tree built?
  - ▶ recursive partitioning (binary or multiway)
2. How can we make predictions from the tree?

# Questions

1. How is the tree built?
  - ▶ recursive partitioning (binary or multiway)
  - ▶ split selection based on impurity reduction (or association measure)
2. How can we make predictions from the tree?

# Questions

## 1. How is the tree built?

- ▶ recursive partitioning (binary or multiway)
- ▶ split selection based on impurity reduction (or association measure)

## 2. How can we make predictions from the tree?

- ▶ drop new observation down the tree
- ▶ terminal node yields prediction:
  - ▶ majority class (classification)
  - ▶ mean (regression)

# Tree building - multiway or binary

traditional recursive partitioning algorithms:

- ▶ multiway splitting (of categorical variables only): C4.5 ([Quinlan, 1993](#))  
implemented in Weka, available through RWeka
- ▶ binary splitting: CART ([Breiman et al., 1984](#))  
implemented in packages and functions `rpart` and `tree`

# Multiway splitting (C4.5)

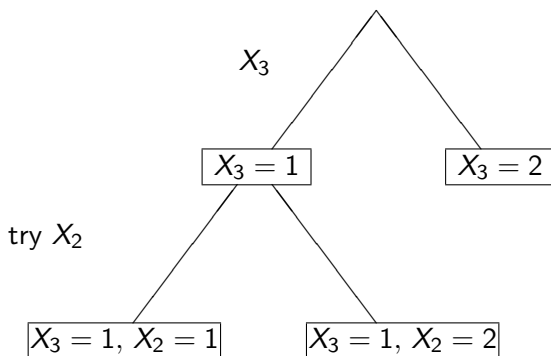
from the starting node

- ▶ produce as many nodes as categories
- ▶ select variable for splitting if most informative with respect to impurity reduction

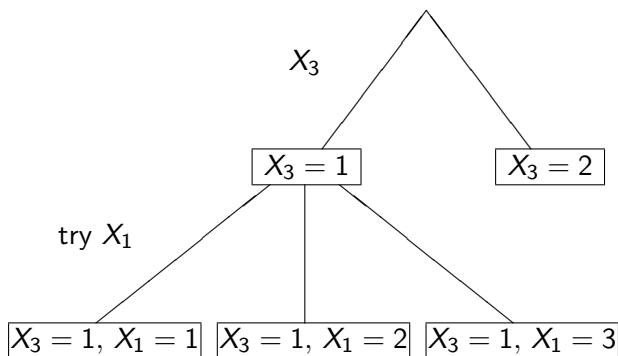
...



## Multiway splitting (C4.5)



## Multiway splitting (C4.5)



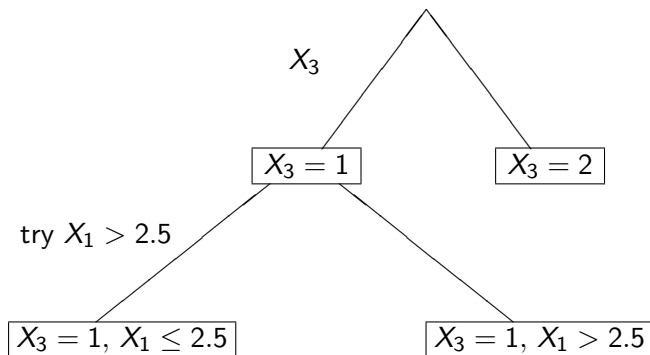
# Binary splitting

from the current node

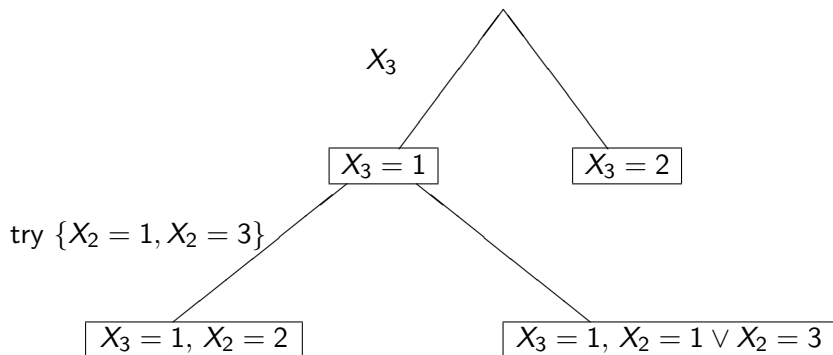
- ▶ try all possible cutpoints for all variables
- ▶ split using the best cutpoint = produce two nodes

...

## Binary splitting - continuous predictor variables



## Binary splitting - categorical predictor variables

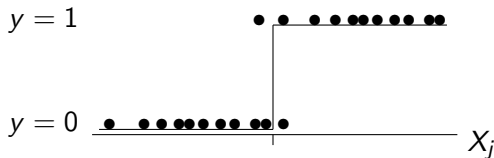


# Tree building - multiway or binary?

Any multi-way split can be represented as a series of binary splits.

# Binary splitting

continuous, ordinal



categorical

$AB|CD$  or  $A|BCD$  or  $AD|BC$  or  $C|ABD$  ...

note: to reduce number of possible splits, CART 'orders' the levels of categorical variables by the mean of the outcome (regression) or the proportion of class 1 observations (classification)

# Cutpoint selection in CART

for current node  $C$  and daughter nodes  $C_{L,c_j}$  and  $C_{R,c_j}$

- ▶ select the best cutpoint  $c_j^*$  within the range of each predictor variable  $X_j$  with respect to impurity reduction

$$c_j^* = \arg \max_{c_j} \Delta \mathcal{J} (C, C_{L,c_j}, C_{R,c_j})$$

(in regression: improvement in SSE instead of impurity reduction)



# Impurity reduction

impurity reduction = impurity before splitting -  
impurity after splitting (weighted mean over nodes)

# Impurity reduction

impurity reduction = impurity before splitting -  
impurity after splitting (weighted mean over nodes)

$$\Delta \mathfrak{I}(C, C_{L,c_j}, C_{R,c_j}) = \mathfrak{I}(C) - \left( \frac{n_{L,c_j}}{n} \mathfrak{I}(C_{L,c_j}) + \frac{n_{R,c_j}}{n} \mathfrak{I}(C_{R,c_j}) \right)$$

where  $n_{L,c_j}$  is the number of observations in  $C$  that are assigned to the left node

(e.g., for a continuous predictor,  $n_{L,c_j} = \sum_i \mathbb{I}(x_{ij} \leq c_j)$ )

# Impurity measures

- ▶ Gini index (used in CART)

$$\mathfrak{I}(C) = \sum_{k=0}^K \hat{p}_k (1 - \hat{p}_k)$$

- ▶ Shannon entropy (used in C4.5)

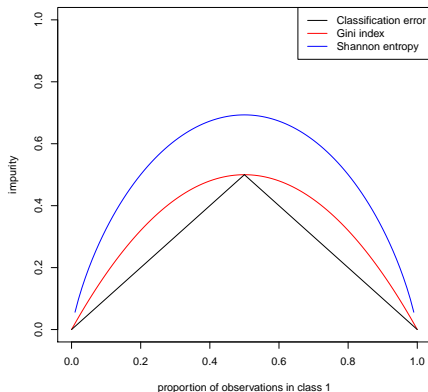
$$\mathfrak{I}(C) = - \sum_{k=0}^K \hat{p}_k \log \hat{p}_k$$

- ▶ Classification error (never or rarely used)

$$\mathfrak{I}(C) = 1 - \max_k \hat{p}_k$$

# Impurity measures

Impurity measures for  $K = 2$  response classes:



Impurity in a node is:

0, if all observations are of the same class.

max, if each class is held by the same number of observations.

# Classification and regression trees

- ▶ tree building:

# Classification and regression trees

- ▶ tree building:
  - ▶ recursively select splitting variable and cutpoint

# Classification and regression trees

- ▶ tree building:
  - ▶ recursively select splitting variable and cutpoint
  - ▶ stop splitting if

# Classification and regression trees

- ▶ tree building:
  - ▶ recursively select splitting variable and cutpoint
  - ▶ stop splitting if
    - ▶ the impurity reduction does not exceed a (user) pre-specified threshold



# Classification and regression trees

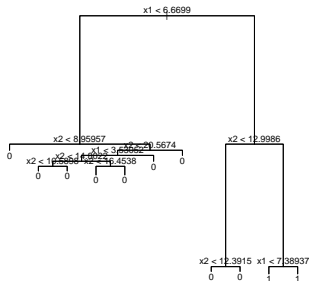
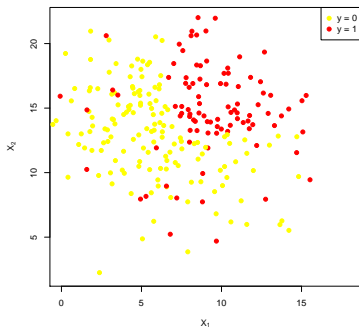
- ▶ tree building:
  - ▶ recursively select splitting variable and cutpoint
  - ▶ stop splitting if
    - ▶ the impurity reduction does not exceed a (user) pre-specified threshold
    - ▶ the number of observations in any node would be smaller than a (user) pre-specified number minimum number

# Classification and regression trees

- ▶ tree building:
  - ▶ recursively select splitting variable and cutpoint
  - ▶ stop splitting if
    - ▶ the impurity reduction does not exceed a (user) pre-specified threshold
    - ▶ the number of observations in any node would be smaller than a (user) pre-specified number minimum number
  - ▶ prune ([Hastie et al., 2001](#)) if necessary

# Pruning

Full tree likely to overfit:



# Pruning

- ▶ Find the optimal number of nodes  $|T|$  for tree  $T$
- ▶ Create a range of subtrees  $T_M$  through  $T_1$ ,
  - ▶  $T_M$  is the full tree,  $T_1$  is a tree with 1 node
  - ▶ In every subtree  $T_m$ , the node that least improved impurity is removed from tree  $T_{m+1}$
- ▶ Full tree  $T$  was build by minimizing  $\sum_{m=1}^M \sum_{x_i \in C_m} L(y_i - \hat{y}_i)$  in the training data
- ▶ Find optimal number of nodes  $|T|$  by minimizing  $\sum_{m=1}^{|T|} \sum_{x_i \in C_m} L(y_i - \hat{y}_i) + \alpha |T|$  in the training data
- ▶ Optimal value of  $\alpha$  is determined through  $k$ -fold CV

## Short detour: Model validation

aim: evaluate model performance and compare models

usually based on prediction error estimates

$$\frac{1}{n} \sum_{i=1}^n I(\hat{y}_i \neq y_i)$$

in classification

$$\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

in regression

# Naive approach

- ▶ fit model on training sample
- ▶ estimate error on same training sample

# Naive approach

- ▶ fit model on training sample
- ▶ estimate error on same training sample

bad idea!

# Naive approach

- ▶ error estimation would be overoptimistic
- ▶ especially for models that overfit, i.e. that
  - ▶ fit too closely to random variation in the training sample
  - ▶ are not generalizable to other samples from the same data generating process

⇒ rewards overfitting



# Test sample

test sample should not be identical to the training sample!

cross validation:

# Test sample

test sample should not be identical to the training sample!

cross validation:

- ▶ set aside part of the original sample as a test sample

# Test sample

test sample should not be identical to the training sample!

cross validation:

- ▶ set aside part of the original sample as a test sample
  - ▶ tradeoff between size of training and test sample

# Test sample

test sample should not be identical to the training sample!

cross validation:

- ▶ set aside part of the original sample as a test sample
  - ▶ tradeoff between size of training and test sample
  - ▶ better: more than one test sample

# Test sample

test sample should not be identical to the training sample!

cross validation:

- ▶ set aside part of the original sample as a test sample
  - ▶ tradeoff between size of training and test sample
  - ▶ better: more than one test sample
- ▶  $k$ -fold cross validation

# Cross validation (CV)

$k$ -fold CV: randomly separate original training sample into  $k$  subsets



- ▶ in  $k$  steps:
- ▶ fit model on training sample minus  $k$ th subset
- ▶ assess prediction error on  $k$ th subset (test sample)  
⇒ gives  $k$  error estimates
- ▶ average over  $k$  error estimates

# Cross validation (CV)

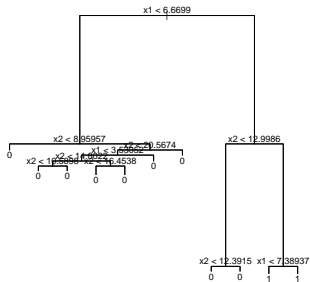
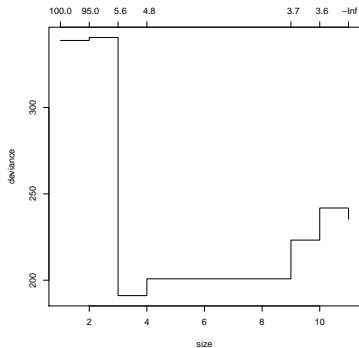
How many folds to use?

Extreme case:  $N$ -fold-CV (leave-one-out-CV, can be overly optimistic)

Rule of thumb:  $k = 10$  gives realistic estimates of prediction error  
[Kohavi \(1995\)](#)

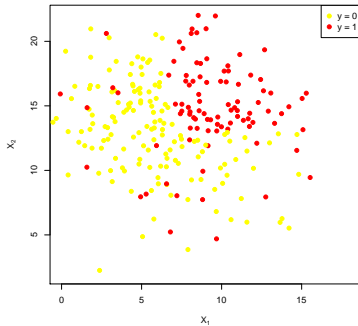
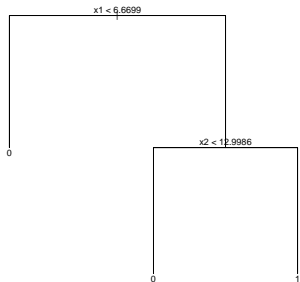
Note: Whenever you perform cross-validation, or random sampling, make sure you can exactly reproduce your results by setting the random seed (e.g, `set.seed(42)`).

## Back to pruning





# Prune tree



# Classification and regression trees

- ▶ missing values  
by means of surrogate variables ([Hastie et al., 2001](#))
- ▶ observation weights  
useful if classes are unbalanced, or if different costs are associated with different types of misclassifications

# Growing CARTs with package tree

tree options and other functions controlling tree growing and size:

<code>na.action =</code>	function to filter missing data. The default is <code>na.pass</code> as tree handles missing values by dropping them down the tree as far as possible.
<code>mincut =</code>	minimum number of observations to include in either child node. The default is 5.
<code>minsize =</code>	smallest allowed node size. The default is 10.
<code>mindev =</code>	The within-node deviance must be at least this times that of the root node for the node to be split. Default is .01.
<code>cv.tree()</code>	Performs $k$ -fold CV cost-complexity (pruning) parameter.
<code>prune.tree()</code>	Prunes tree to size found with <code>cv.tree</code> .

# Interpretation

- ▶ variables that appear in the tree are (supposed to be) more relevant than those that do not appear in the tree

# Interpretation

- ▶ variables that appear in the tree are (supposed to be) more relevant than those that do not appear in the tree
- ▶ variables that appear high up in the tree affect more observations than those that appear lower in the tree

# Interpretation

- ▶ variables that appear in the tree are (supposed to be) more relevant than those that do not appear in the tree
- ▶ variables that appear high up in the tree affect more observations than those that appear lower in the tree

Note: only descriptive (beware of causal attributions)!

## Comparison to linear models

	regression tree	linear model
model	$f(\mathbf{x}) = \sum_m \hat{y}_m I(\mathbf{x} \in C_m)$	$f(\mathbf{x}) = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \dots$

# Comparison to linear models

	regression tree	linear model
model	$f(\mathbf{x}) = \sum_m \hat{y}_m I(\mathbf{x} \in C_m)$	$f(\mathbf{x}) = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \dots$
fitting	recursive partitioning	least squares, closed form



## Comparison to linear models

	regression tree	linear model
model	$f(\mathbf{x}) = \sum_m \hat{y}_m \mathbb{I}(\mathbf{x} \in C_m)$	$f(\mathbf{x}) = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \dots$
fitting	recursive partitioning	least squares, closed form
$N < p$	can deal	cannot, need penalized regression

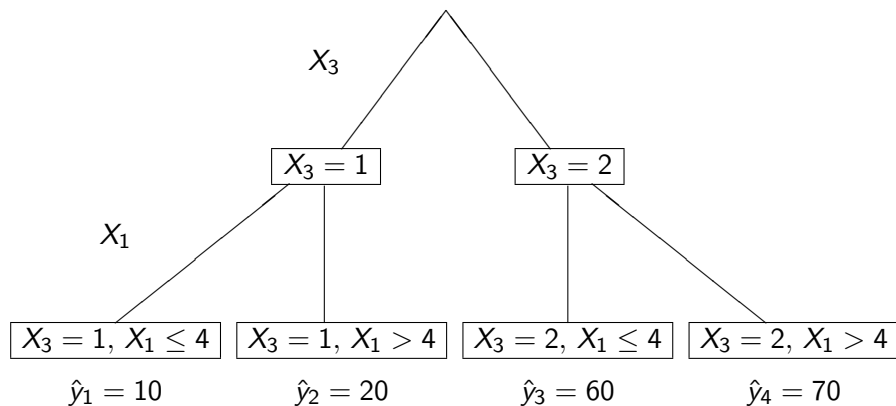
## Comparison to linear models

	regression tree	linear model
model	$f(\mathbf{x}) = \sum_m \hat{y}_m I(\mathbf{x} \in C_m)$	$f(\mathbf{x}) = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \dots$
fitting	recursive partitioning	least squares, closed form
$N < p$	can deal	cannot, need penalized regression
interpret.	easy, tree structure	$\hat{\beta}_j$ with sd

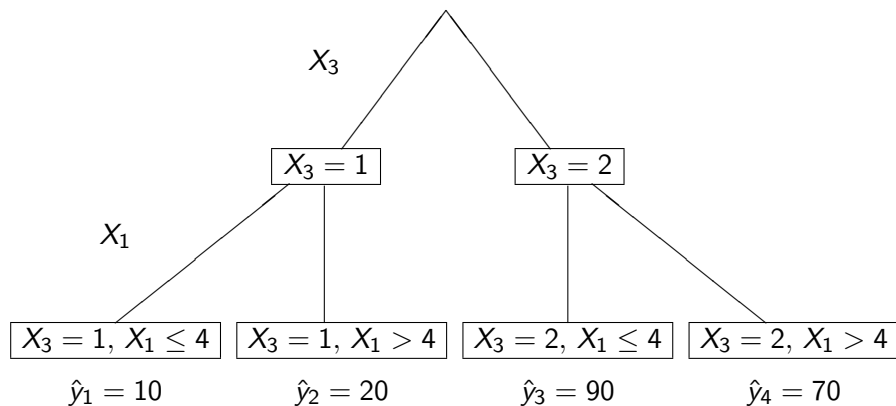
# Comparison to linear models

	regression tree	linear model
model	$f(\mathbf{x}) = \sum_m \hat{y}_m \mathbb{I}(\mathbf{x} \in C_m)$	$f(\mathbf{x}) = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \dots$
fitting	recursive partitioning	least squares, closed form
$N < p$	can deal	cannot, need penalized regression
interpret.	easy, tree structure	$\hat{\beta}_j$ with sd
main effect	?	$\hat{\beta}_1 \cdot x_1$
interaction	?	$\hat{\beta}_{1,2} \cdot x_1 \cdot x_2$

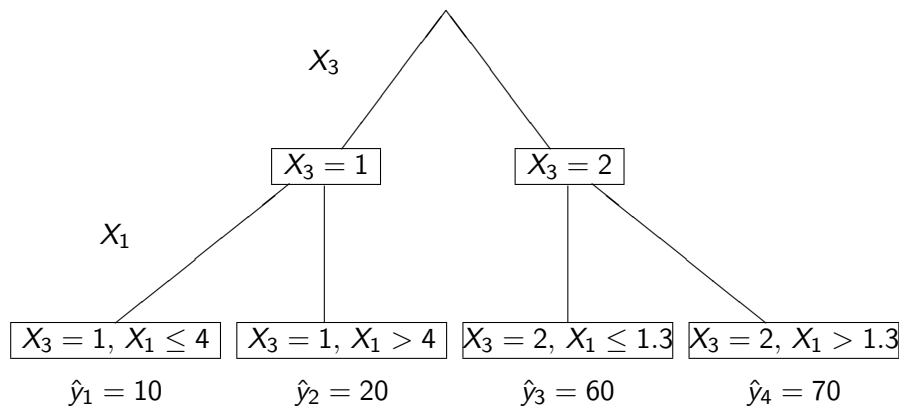
## Interpretation: Two main effects



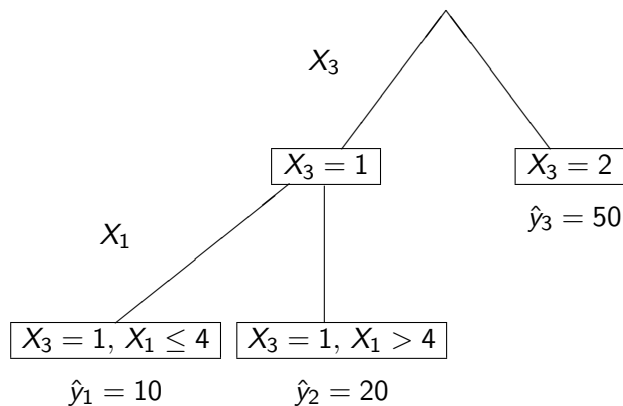
## Interpretation: Interaction



## Interpretation: Interaction



## Interpretation: Interaction



# Problems of CART

## I. variable selection bias



# Problems of CART

## I. variable selection bias

- ▶ continuous variables and variables with many categories, as well as variables with missing values, are preferred in original CART and C4.5 (cf., e.g., [White and Liu, 1994](#), [Loh and Shih, 1997](#), [Jensen and Cohen, 2000](#), [Kim and Loh, 2001](#), [Dobra and Gehrke, 2001](#))

# Problems of CART

## I. variable selection bias

- ▶ continuous variables and variables with many categories, as well as variables with missing values, are preferred in original CART and C4.5 (cf., e.g., [White and Liu, 1994](#), [Loh and Shih, 1997](#), [Jensen and Cohen, 2000](#), [Kim and Loh, 2001](#), [Dobra and Gehrke, 2001](#))
- ▶ due to multiple testing and biased entropy estimation ([Strobl et al., 2007](#))

# Problems of CART

## I. variable selection bias

# Problems of CART

## I. variable selection bias

- ▶ continuous variables and variables with many categories, as well as variables with missing values, are preferred in original CART and C4.5 (cf., e.g., [White and Liu, 1994](#), [Loh and Shih, 1997](#), [Jensen and Cohen, 2000](#), [Kim and Loh, 2001](#), [Dobra and Gehrke, 2001](#))

# Problems of CART

## I. variable selection bias

- ▶ continuous variables and variables with many categories, as well as variables with missing values, are preferred in original CART and C4.5 (cf., e.g., [White and Liu, 1994](#), [Loh and Shih, 1997](#), [Jensen and Cohen, 2000](#), [Kim and Loh, 2001](#), [Dobra and Gehrke, 2001](#))
- ▶ due to multiple testing and biased entropy estimation ([Strobl et al., 2007](#))

# Solution

## I. variable selection bias

# Solution

## I. variable selection bias

- ▶ unbiased variable selection approaches ([Loh and Shih, 1997](#), [Kim and Loh, 2001](#), [Dobra and Gehrke, 2001](#), [Shih, 2004](#), [Lausen et al., 2004](#), [Hothorn et al., 2006](#), [Strobl et al., 2007](#))

# Solution

## I. variable selection bias

- ▶ unbiased variable selection approaches ([Loh and Shih, 1997](#), [Kim and Loh, 2001](#), [Dobra and Gehrke, 2001](#), [Shih, 2004](#), [Lausen et al., 2004](#), [Hothorn et al., 2006](#), [Strobl et al., 2007](#))
  - ⇒ p-value of optimally selected ( $\chi^2$ , rank or Gini) statistic ([Shih, 2004](#), [Lausen et al., 2004](#), [Strobl et al., 2007](#))  
correct for optimal choice of cutpoint



# Solution

## I. variable selection bias

- ▶ unbiased variable selection approaches (Loh and Shih, 1997, Kim and Loh, 2001, Dobra and Gehrke, 2001, Shih, 2004, Lausen et al., 2004, Hothorn et al., 2006, Strobl et al., 2007)
  - ⇒ p-value of optimally selected ( $\chi^2$ , rank or Gini) statistic (Shih, 2004, Lausen et al., 2004, Strobl et al., 2007)  
correct for optimal choice of cutpoint
  - ⇒ separate variable and cutpoint selection (Loh and Shih, 1997, Kim and Loh, 2001, Hothorn et al., 2006)

# Solution

## I. variable selection bias

- ▶ package `party` or `partykit` for unbiased recursive partitioning ([Hothorn et al., 2006, 2017](#), [Hothorn and Zeileis, 2015](#))
  - ▶ function `ctree` for classification, regression and survival trees

# Solution

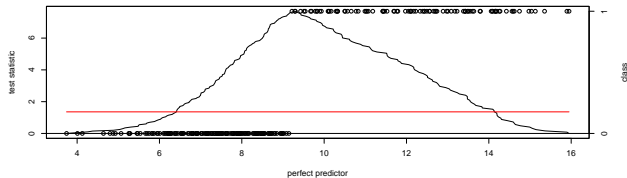
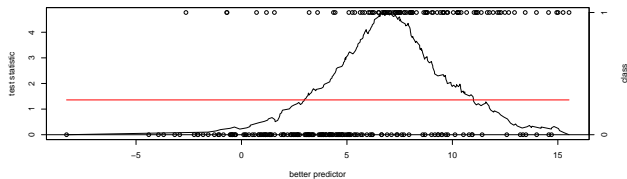
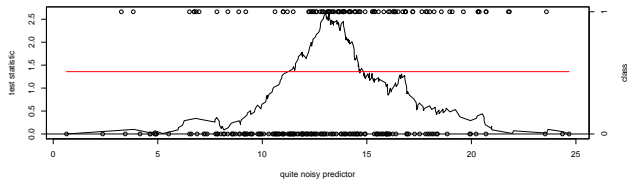
## I. variable selection bias

- ▶ package `party` or `partykit` for unbiased recursive partitioning ([Hothorn et al., 2006, 2017](#), [Hothorn and Zeileis, 2015](#))
  - ▶ function `ctree` for classification, regression and survival trees
- ▶ splitting based on p-values of conditional inference tests, that also serve as a stopping criterion
  - ⇒ separates variable and split selection
  - ⇒ no overfitting
  - ⇒ pruning not necessary

# Conditional inference trees (Hothorn et al., 2006)

1. Test the global null hypothesis of independence between any of the  $m$  covariates and the response. If null hypothesis can be rejected, select the covariate  $X_j^*$  with strongest association to  $Y$ . Otherwise, stop.
2. Select the splitting value for  $X_j^*$  that optimally separates the observations in terms of outcome  $Y$ .
3. Split the observations and repeat the procedure in each of the resulting daughternodes.

# Test statistics for selecting splitting value



# ctree function

ctree options controlling tree growing process:

<code>mincriterion =</code>	value of the test statistic or (1-p)-value that must be exceeded in order to implement a split.
<code>maxdepth =</code>	maximum depth of the tree. The default <code>maxdepth = Inf</code> means that no restrictions are applied to tree sizes.
<code>minsplit =</code>	the minimum sum of weights in a node in order to be considered for splitting.
<code>minbucket =</code>	the minimum sum of weights in a terminal node
<code>minprob =</code>	proportion of observations needed to establish a terminal node
<code>splittry =</code>	number of variables that are inspected for admissible splits if the best split doesn't meet the sample size constraint

# ctree function

ctree options controlling tree growing process:

<code>multiway =</code>	a logical indicating if multiway splits for all factor levels are implemented for unordered factors.
<code>maxsurrogate =</code>	number of surrogate splits to evaluate. Note that currently only surrogate splits in ordered covariables are implemented.
<code>majority =</code>	if TRUE, observations which can't be classified to a daughter node because of missing information are randomly assigned (following the node distribution). If FALSE, they go with the majority (the default in ctree).

# Model-based recursive partitioning

Rationale:

Global parametric model (e.g.,  $Y = X\beta + \epsilon$ ) may not fit the data well. When additional covariates are available, it may be possible to partition the data with respect to these covariates and find better-fitting models in each cell of the partition.

Zeileis et al. (2008)



# Model-based recursive partitioning

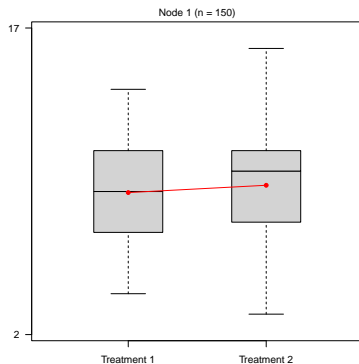
1. Fit a parametric model (e.g., estimate  $\hat{\beta}$  for a (G)LM) to all observations in the current node.
2. Test whether the parameter estimates are stable with respect to every ordering of  $U_1, \dots, U_q$ . If null hypothesis of parameter stability can be rejected, select covariate  $U_j^*$  associated with the highest instability. Otherwise, stop.
3. Select the splitting value for  $U_j^*$  that locally optimizes the loss function in each of the resulting daughter nodes.
4. Split the observations and repeat the procedure in each of the resulting daughter nodes.

## Example: Linear-model based recursive partitioning

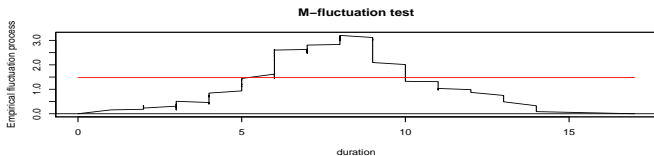
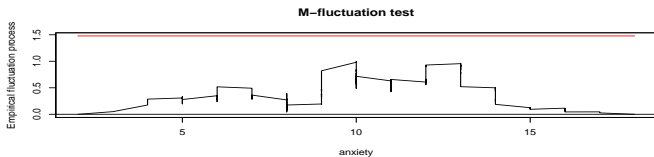
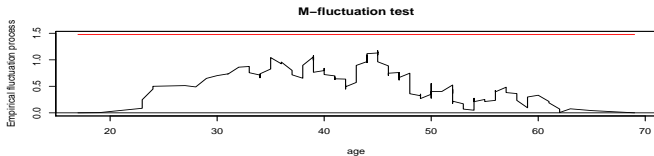
Dataset of  $N = 150$  patients receiving one of two treatments for depression.

- ▶  $X$ : treatment indicator
- ▶  $Y$ : post-treatment depressive symptom score
- ▶  $Z_1, \dots, Z_p$ : age, anxiety, duration

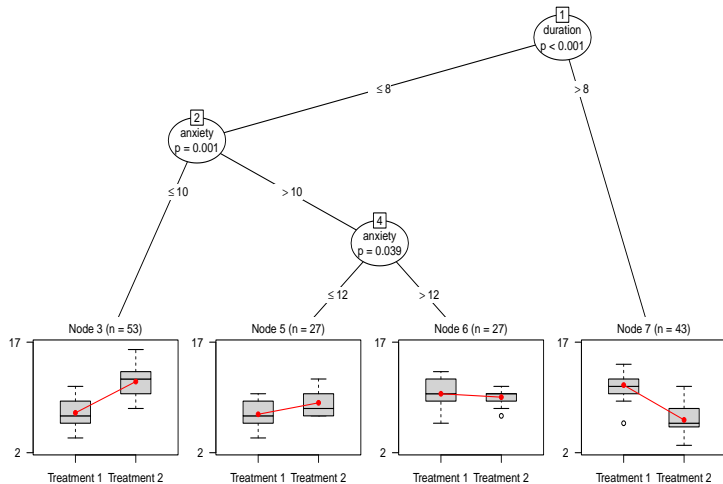
# Global linear model



# Parameter stability tests for linear model



# Linear model tree



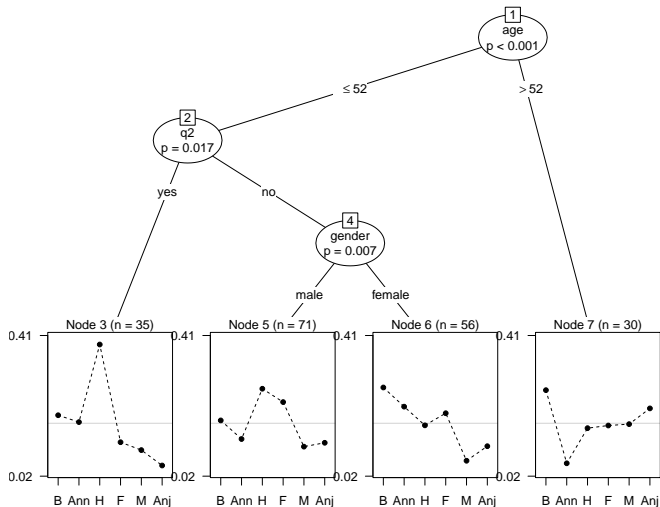
## Example: Bradley-Terry model based recursive partitioning

Dataset of  $N = 192$  respondents, who judged the attractiveness of the top six contestants on Germany's Next Topmodel 2007.

- ▶ preference: Preferences for all 15 paired comparisons from 6 contestants: Barbara, Anni, Hana, Fiona, Mandy, and Anja.
- ▶  $Z_1, \dots, Z_p$ : gender, age, q1, q2, q3
  - q1: Do you recognize the women on the pictures?
  - q2: Did you watch Germany's Next Topmodel regularly?
  - q3: Do you know who won Germany's Next Topmodel?

Further reference: [Strobl et al. \(2011\)](#) and R-package psychotree.

# Bradley-Terry tree



# Clustered / multilevel / longitudinal data

Lower-level observations nested within higher-level units  $\Rightarrow$

Linear mixed-effects model:  $y = X\beta + Ub + \epsilon$

Recursive partitioning of mixed-effects models:

- ▶ replace fixed-effects part  $X\beta$  by  $f(X)$ , found through recursive partitioning
- ▶ estimate random-effects parameters as usual



# Example: mixed-effects model based recursive partitioning

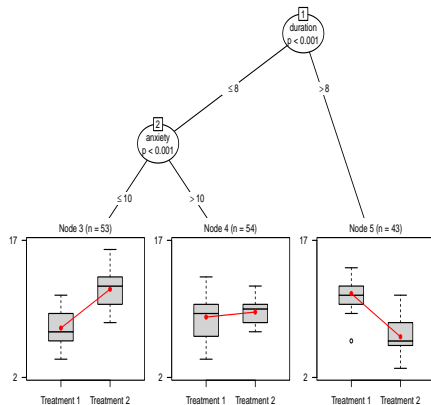
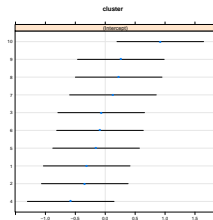
Dataset of  $N = 150$  patients receiving one of two treatments for depression.

- ▶  $X$ : treatment indicator
- ▶  $Y$ : post-treatment depressive symptom score
- ▶  $Z_1, \dots, Z_p$ : age, anxiety, duration
- ▶  $U$ : dummy indicators for treatment center ( $n = 10$ )

Further reference: [Fokkema et al. \(2015\)](#) and R-package `glmertree`.

# Linear mixed-effects model tree

$$Y = X\beta_j + Ub + \epsilon$$



# References I

- L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall, New York, 1984.
- A. Dobra and J. Gehrke. Bias correction in classification tree construction. In C. E. Brodley and A. Pohoreckij Danyluk, editors, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2001)*, Williams College, Williamstown, MA, USA, pages 90–97. Morgan Kaufmann, 2001.
- M. Fokkema, N. Smits, A. Zeileis, T. Hothorn, and H. Kelderman. Detecting treatment-subgroup interactions in clustered data with generalized linear mixed-effects model trees. *Working Papers in Economics and Statistics*, 2015.
- T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.
- T. Hothorn and A. Zeileis. partykit: A modular toolkit for recursive partytioning in r. *Journal of Machine Learning Research*, 16:3905–3909, 2015.
- T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.
- T. Hothorn, K. Hornik, C. Strobl, and A. Zeileis. party: A laboratory for recursive part(y)itioning. 2017. URL <http://CRAN.R-project.org/package=party>. R package version 1.2-3.
- D. D. Jensen and P. R. Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.
- H. Kim and W. Loh. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96:589–604, 2001.
- R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 14, pages 1137–1145. Stanford, CA, 1995.
- B. Lausen, T. Hothorn, F. Bretz, and M. Schumacher. Assessment of optimal selected prognostic factors. *Biometrical Journal*, 46(3):364–374, 2004.
- W. Loh and Y. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7:815–840, 1997.
- J. R. Quinlan. *C4.5: Programms for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, 1993.
- Y.-S. Shih. A note on split selection bias in classification trees. *Computational Statistics and Data Analysis*, 45(3): 457–466, 2004.

# References II

- C. Strobl, A.-L. Boulesteix, and T. Augustin. Unbiased split selection for classification trees based on the Gini Index. *Computational Statistics & Data Analysis*, 52(1):483–501, 2007.
- C. Strobl, F. Wickelmaier, and A. Zeileis. Accounting for individual differences in bradley-terry models by means of recursive partitioning. *Journal of Educational and Behavioral Statistics*, 36(2):135–153, 2011.
- A. White and W. Liu. Bias in information based measures in decision tree induction. *Machine Learning*, 15(3):321–329, 1994.
- A. Zeileis, T. Hothorn, and K. Hornik. Model-based recursive partitioning. *Journal of Computational and Graphical Statistics*, 17(2):492–514, 2008.