DGPS 2018 Pre-Conference-Workshop

An Introduction to Machine Learning for Psychologists in R

Florian Pargent & Clemens Stachl

16.09.2018

Agenda

- MORNING
  - Introduction
  - Basic Concepts
    - Regression + Classification
    - Performance Evaluation
    - Bias-Variance Tradeoff
    - Overfitting
    - Training & Test Data/Resampling
    - **Hands-on: Resampling**
    - (Nested Resampling & Variable Selection)
- LUNCH
- AFTERNOON
  - Decision Trees & Random Forests + **Hands-on: Training**
  - **Hands-on: Benchmark Analysis**
  - Variable Importance + Partial Dependence Plots
  - Take Home Message

Predictive Modeling/Machine Learning

Predictive Modeling:

- **Goal:** Make data-driven predictions/decisions
- Utilize flexible algorithms to detect patterns in data
- Accuracy of predictions is criterion for model selection
- Examples:
  - predict rent for new flats
  - recognize individual differences in user data
  - identify faces/objects in images
  - recognize words/language in audio data

Two Modeling Cultures (Breiman and others 2001)

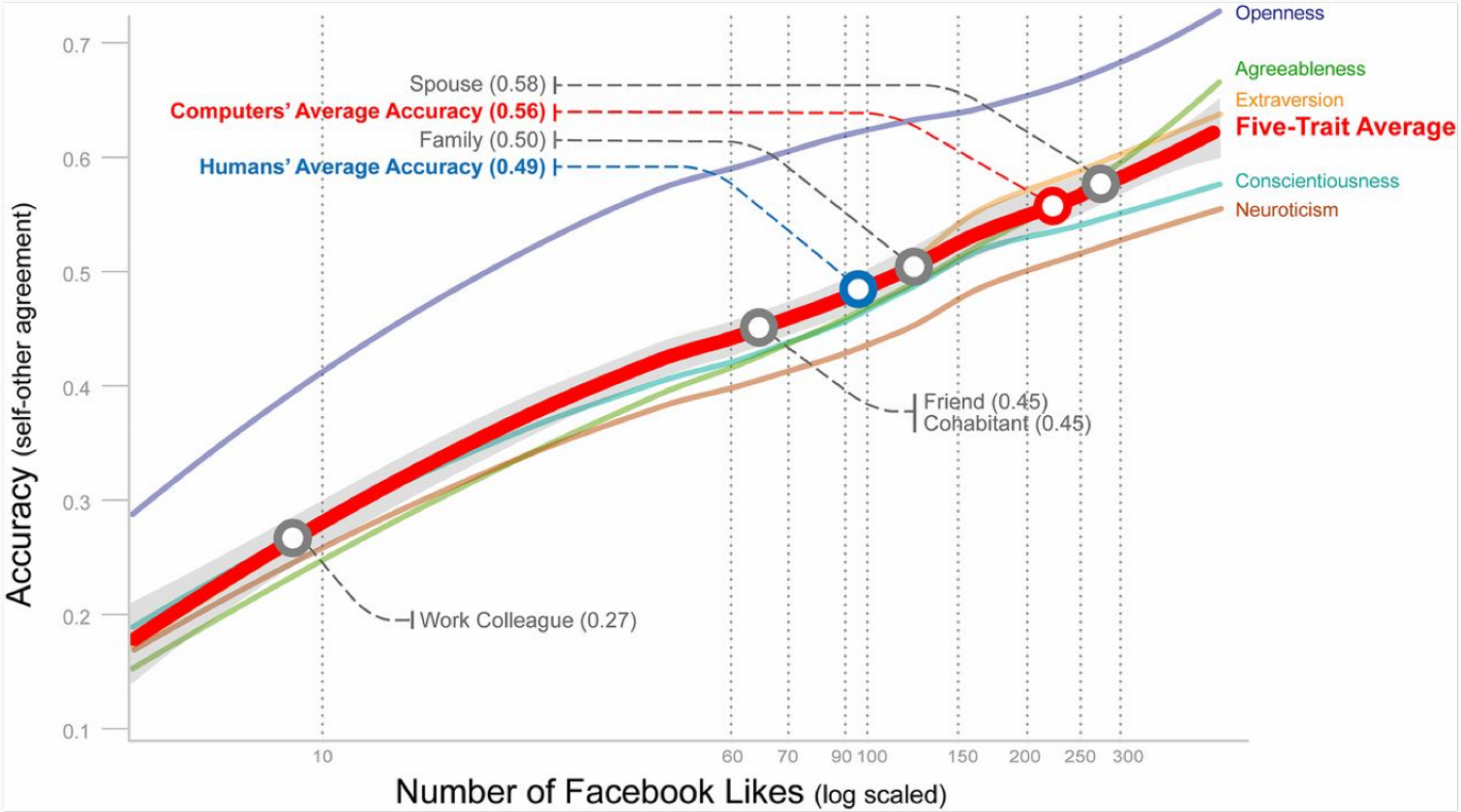
- Classical
  - data can be described with a stochastic model
  - "we assume a linear relationship. . . "
  - model-evaluation: goodness-of-fit indices (R^2, AIC, etc.), effect direction, residuals
  - sometimes good for explanation
  - often bad for prediction
- Algorithmic
  - data is created by an unknown, potentially complex process
  - find a function that can predict the target with high accuracy
  - model-evaluation = prediction accuracy
  - often good for prediction
  - often (still) bad for explanation

Explanation vs. Prediction

- Psychology has heavy focus on explanation (Yarkoni and Westfall 2017)
- Models often not adequately validated
- "This has led to irrelevant theory and questionable conclusions . . . " (Breiman and others 2001)
- High-dimensional data - complex relationships, hard to hypothesize
- Create new measures, reflect on and improve existing theories
- See if theories predict relevant criteria (Shmueli 2010)

Applications I (Youyou, Kosinski, and Stillwell 2015)

Computer-based personality judgment accuracy (y axis), plotted against the number of Likes available for prediction (x axis).



Wu Youyou et al. PNAS 2015;112:4:1036-1040

PNAS

Applications II

Work from our group at LMU Munich:

- Gender prediction from driving behavior (Stachl and Bühner 2015)
- Predictive modeling with psychological panel data (Pargent and Albert-von der Gönna 2018)
- Digital footprints of sensation seeking (Schoedel et al. 2018)

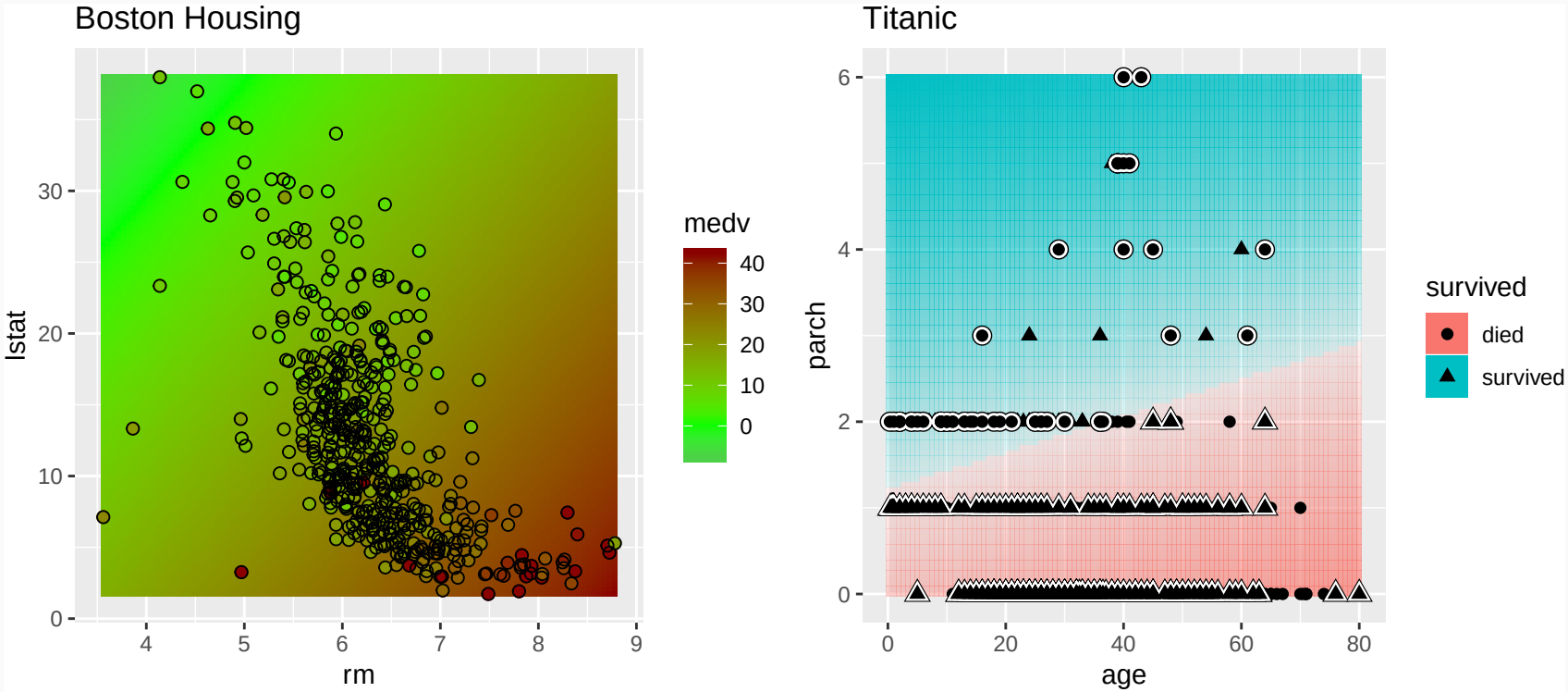
Check out the symposia we contribute to

- *Monday 16:45* (C1 – Big Data: The new Big Promise for research in applied psychology?)
- *Thursday 10:00* (K25 – Prädiktionen in der Psychologie - Verhalten, Persönlichkeit, Emotionen, Leistung, Orte und Person-Kultur Passung)

Basics

Regression and Classification

- Regression: continuous target
- Classification: categorical/binary target



Performance Measures

Model quality is evaluated based on prediction accuracy

- Quantify accuracy/prediction error
- Ultimate question: how well can new data be predicted based on a fitted model?
- Different measures for evaluation

  -> Relevant for model selection and model performance

Regression – Performance Measures

- Mean squared error (MSE)
- Quantification of a "typical" deviation from the true value

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

$$R^2 = 1 - \frac{\textit{residual sum of squares}}{\textit{total sum of squares}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

Classification – Performance Measures

- Mean misclassification error ($\widehat{=}$ MSE)
- Proportion of misclassified cases

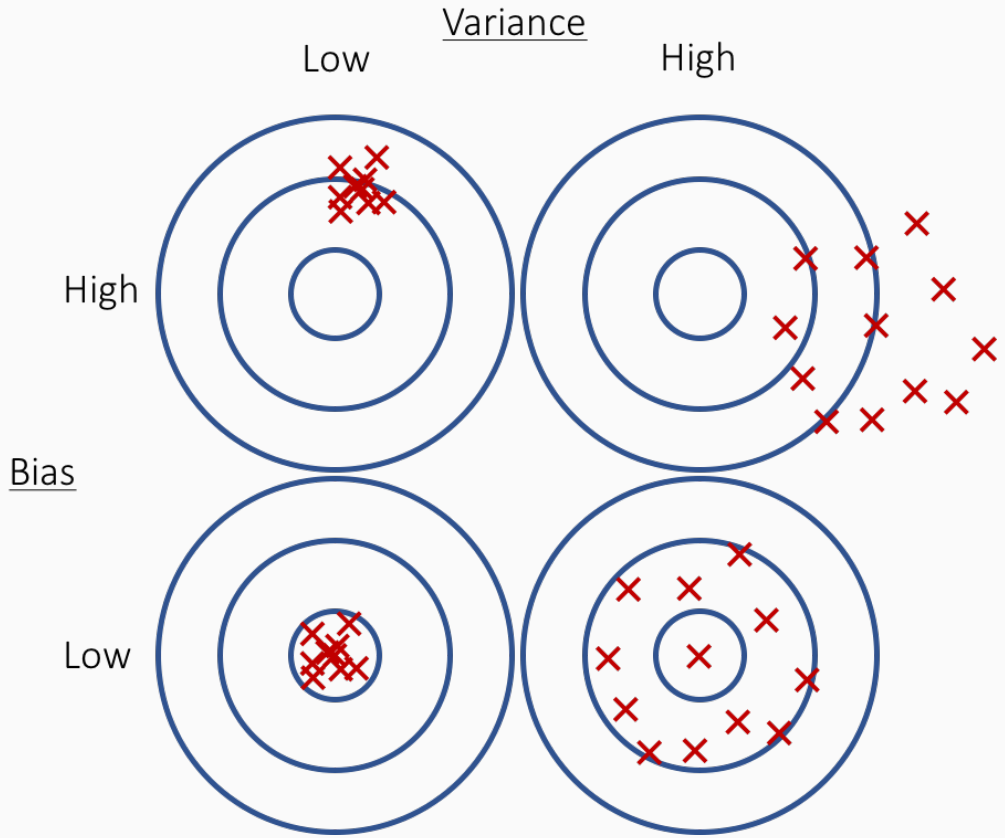
$$MMCE = \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

Classification - Performance Measures

Confusion Matrix		Truth y_i	
		1	0
Prediction $\hat{y}_i$	1	TP	FP
	0	FN	TN

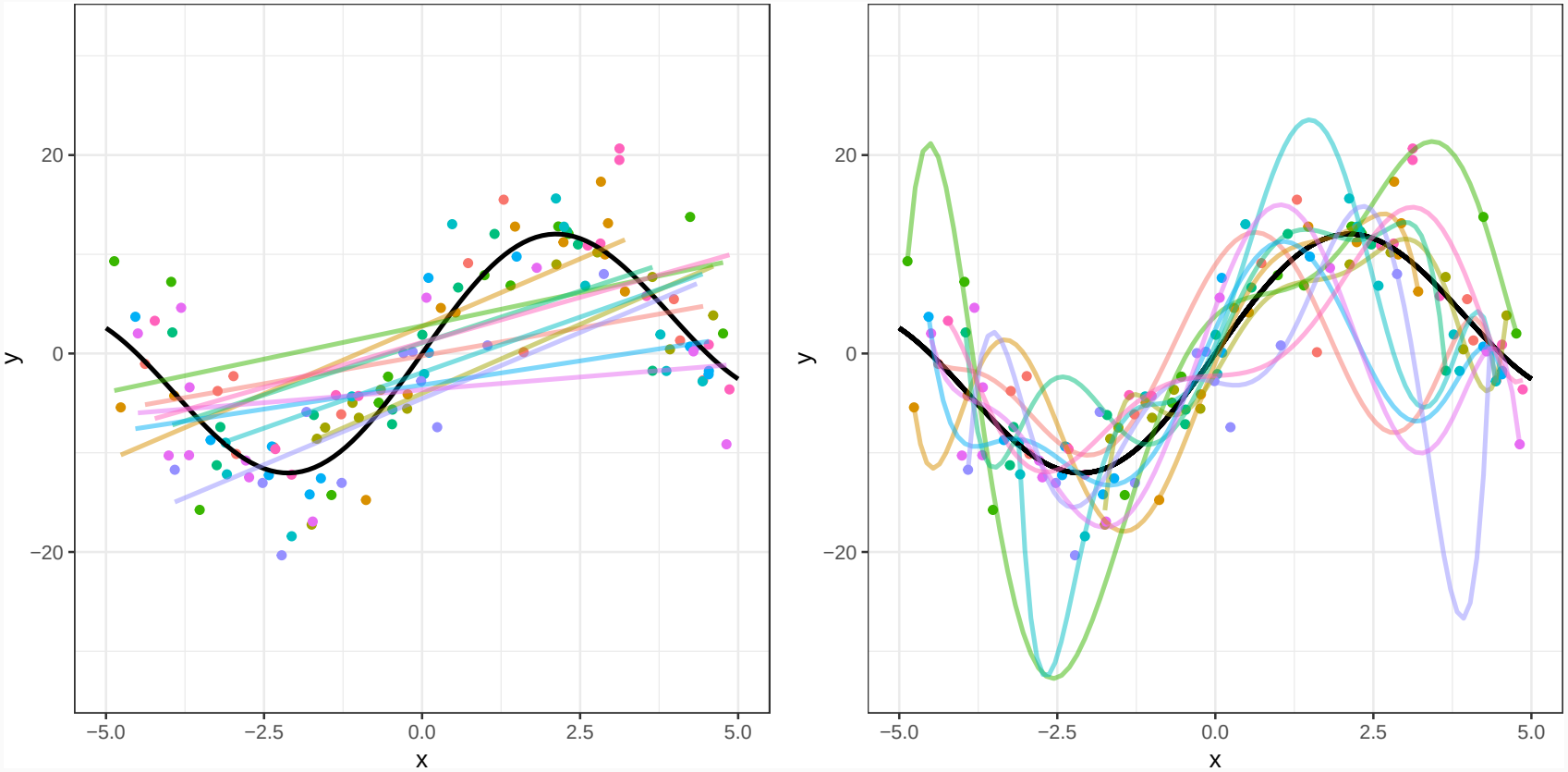
- Sensitivity = TP/(TP+FN)
- Specificity = TN/(TN+FP)
- Positive predictive value = TP/(TP+FP)
- Negative predictive value = TN/(TN+FN)

Prediction Error = f(Bias, Variance, Noise)



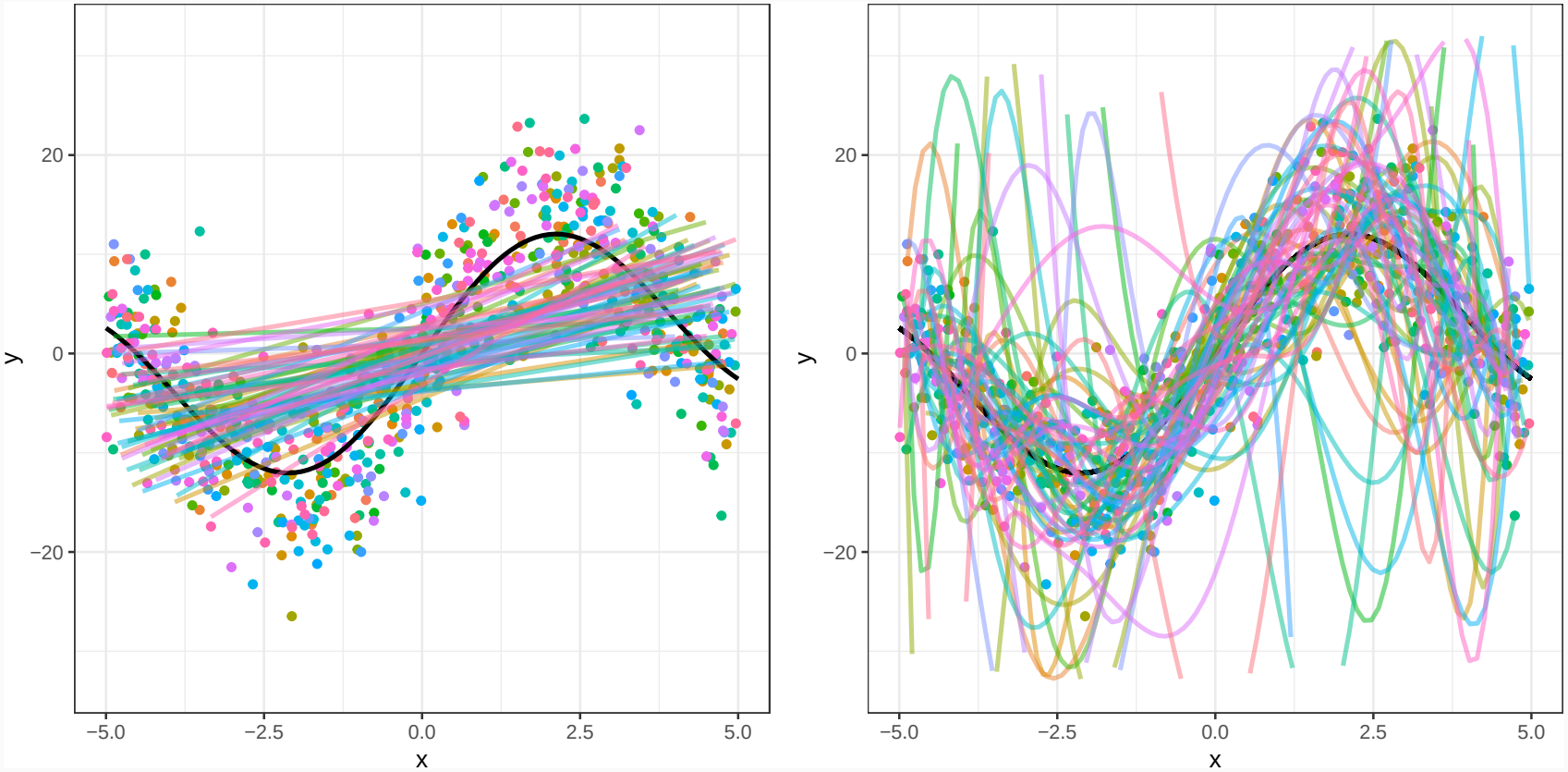
- *Bias:* deviation of the average prediction from the true value
- *Variance:* variability of predictions based on different samples
- *Noise:* irreducible error of the true population model
- **Goal:** find a predictive model with low bias **AND** low variance

Bias - Variance Tradeoff I



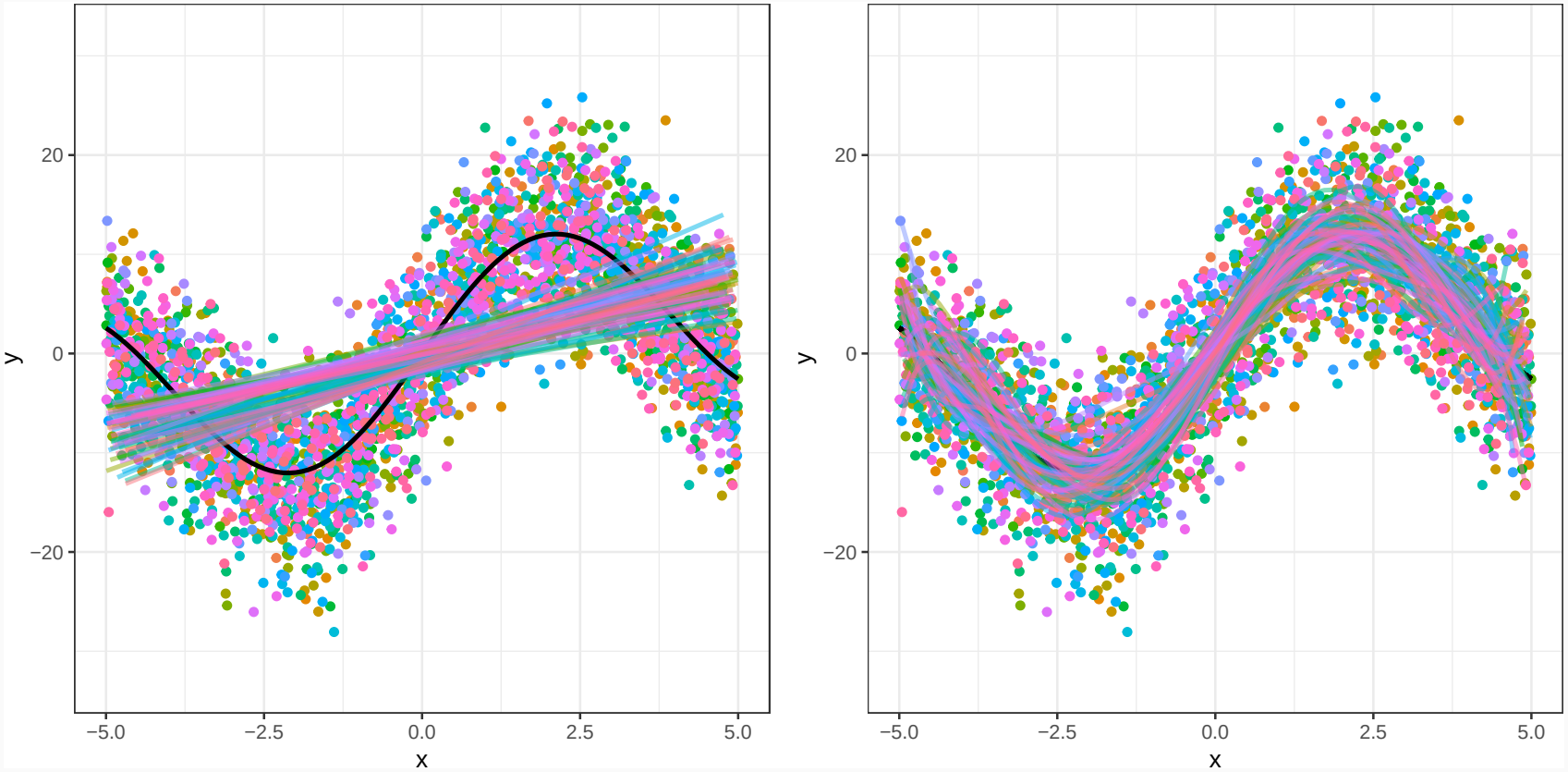
- Ten samples with N = 12 from the population model (black)
- Fit unflexible (left), flexible (right) models within each sample

Bias - Variance Tradeoff II (increase number of samples)



- Bias: high for unflexible, low for flexible models
- Variance: low for unflexible, high for flexible models

Bias - Variance Tradeoff III (increase sample size)

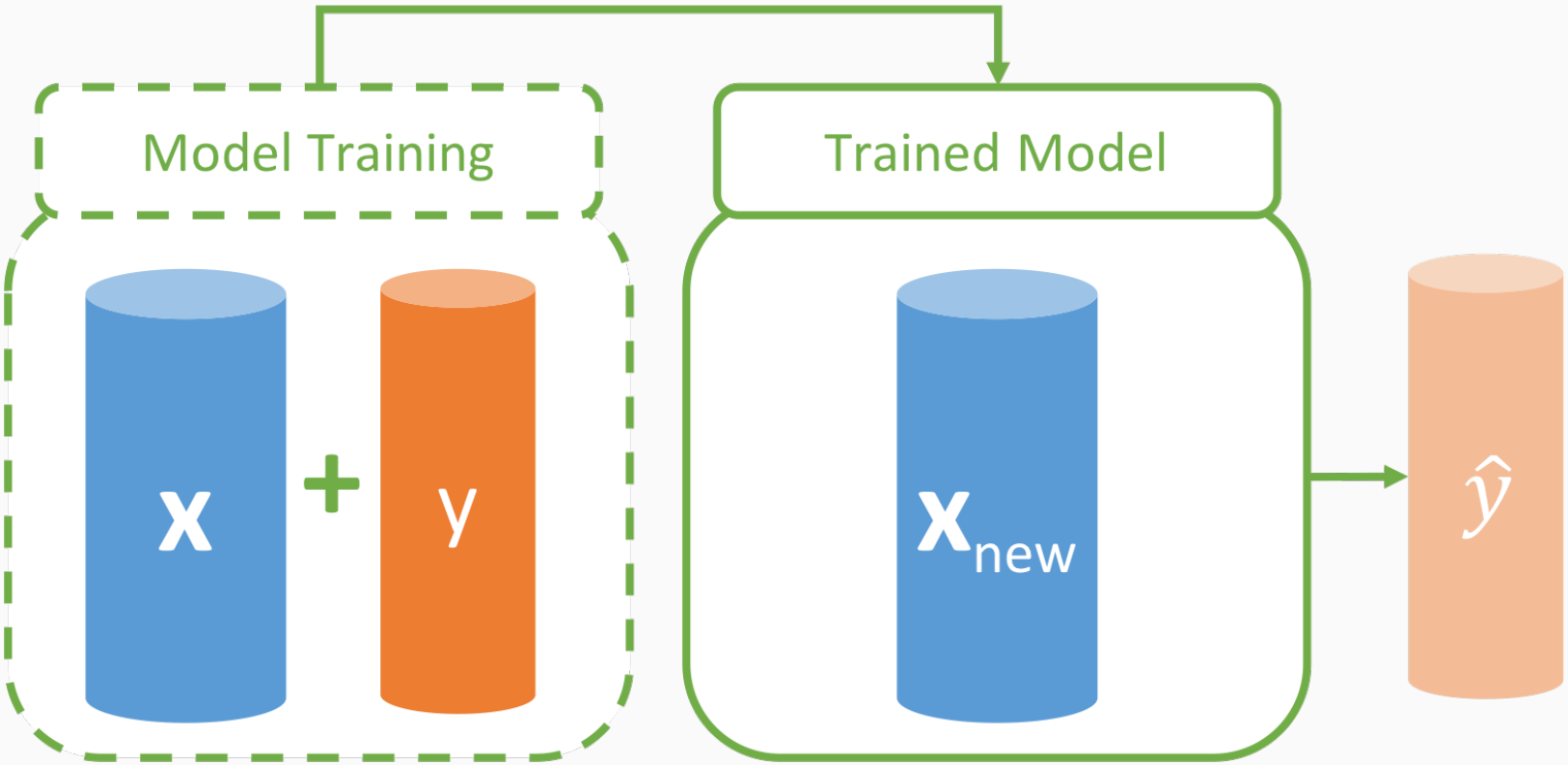


- Bigger sample size reduces variance
- More accurate predictions with flexible models

Training Set and Test Set

- How well does our model predict new data from the same population?
  - Option 1: get new data ;-)
  - Option 2: use prediction error in-sample :-(
  - Option 3: use available data in a smart way :-)
- Strict separation of model training and model evaluation:
  - **Training set**
    - algorithm is trained on this data
  - **Test set**
    - trained algorithm is evaluated

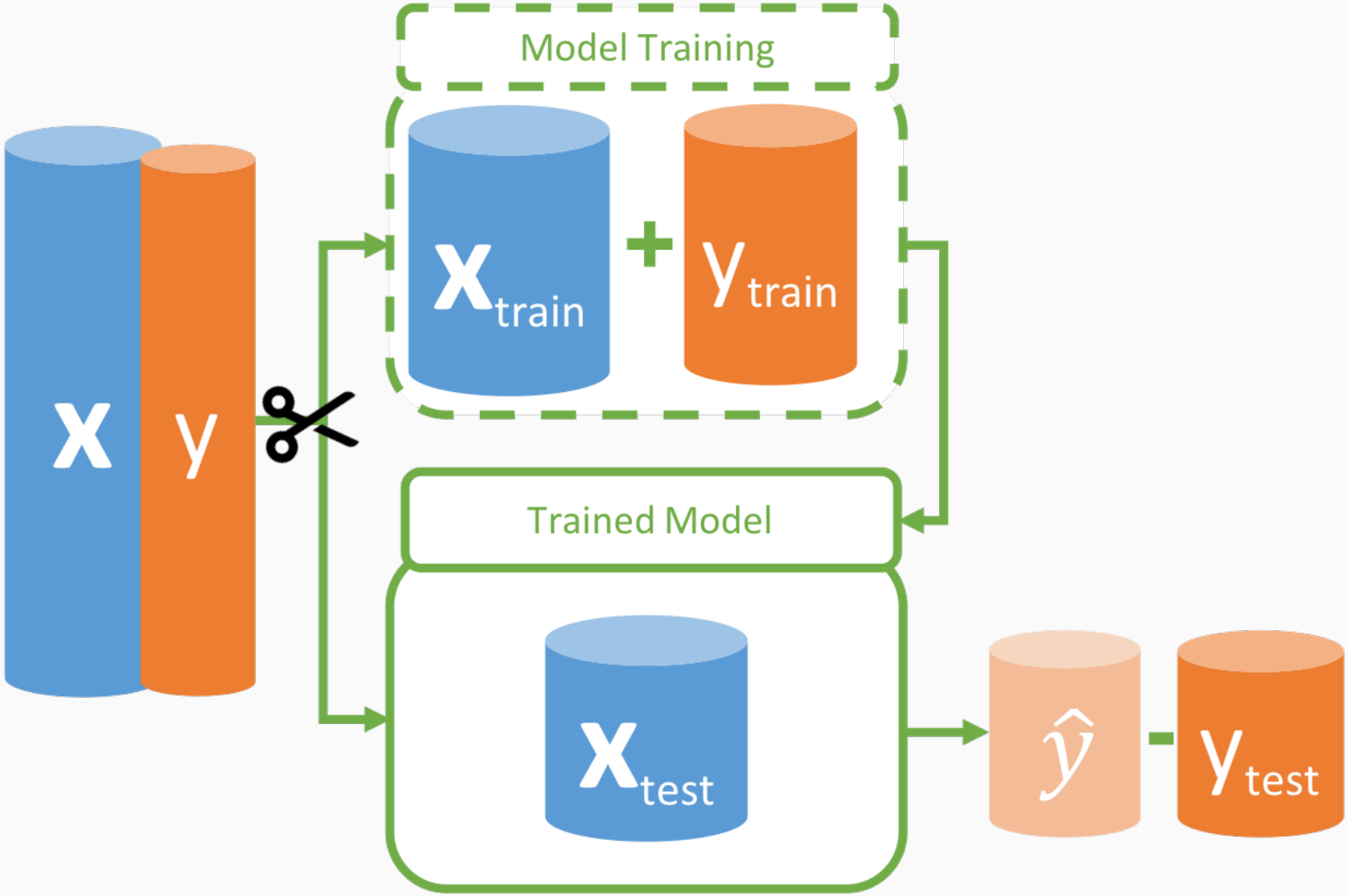
General Idea of Model Evaluation I



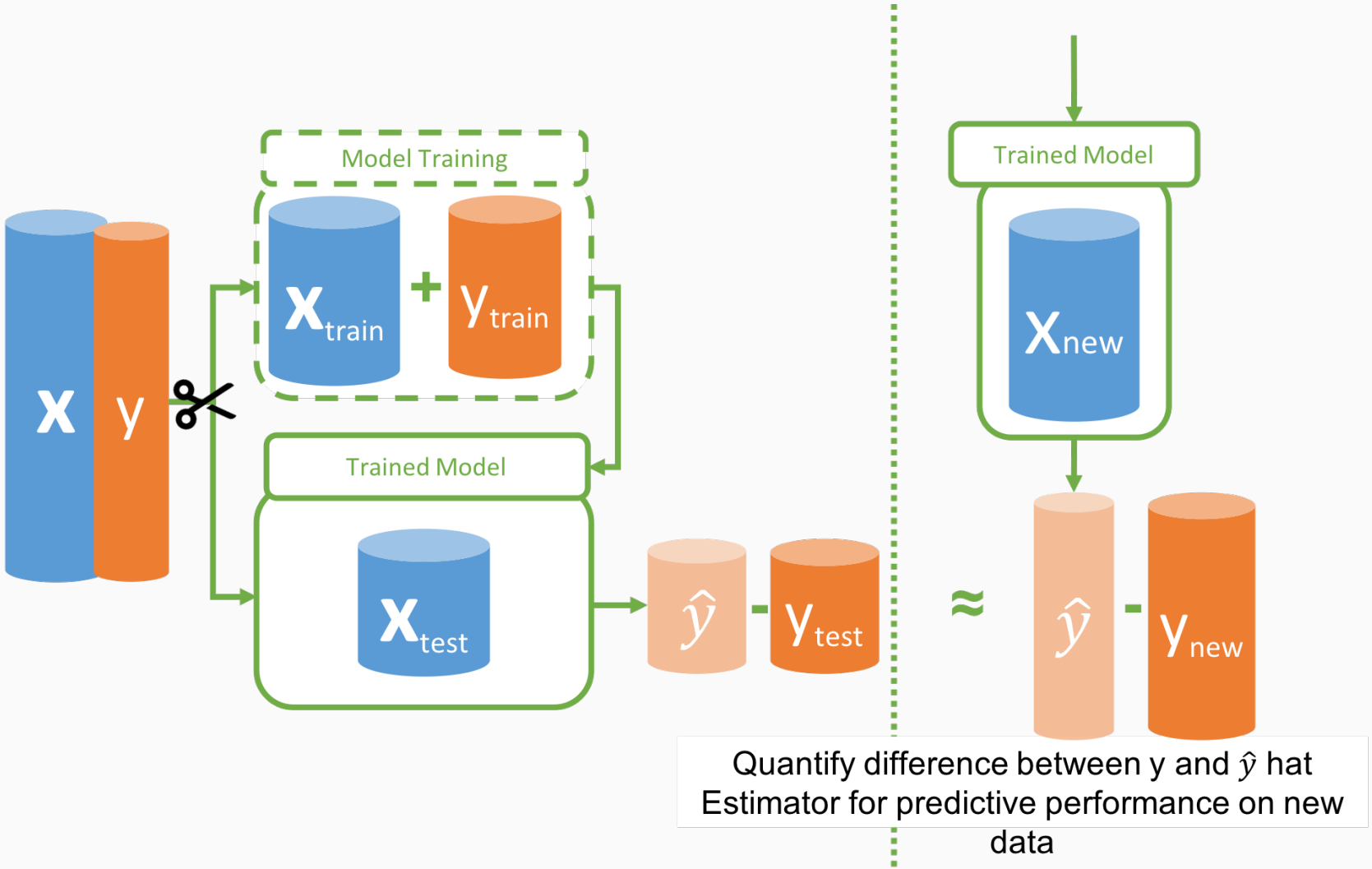
Learn to associate patterns in **X** with values of y

Model evaluation: how good does my model perform on new data?

General Idea of Model Evaluation II



General Idea of Model Evaluation III



Why Do We Have to Separate Training and Test Data?

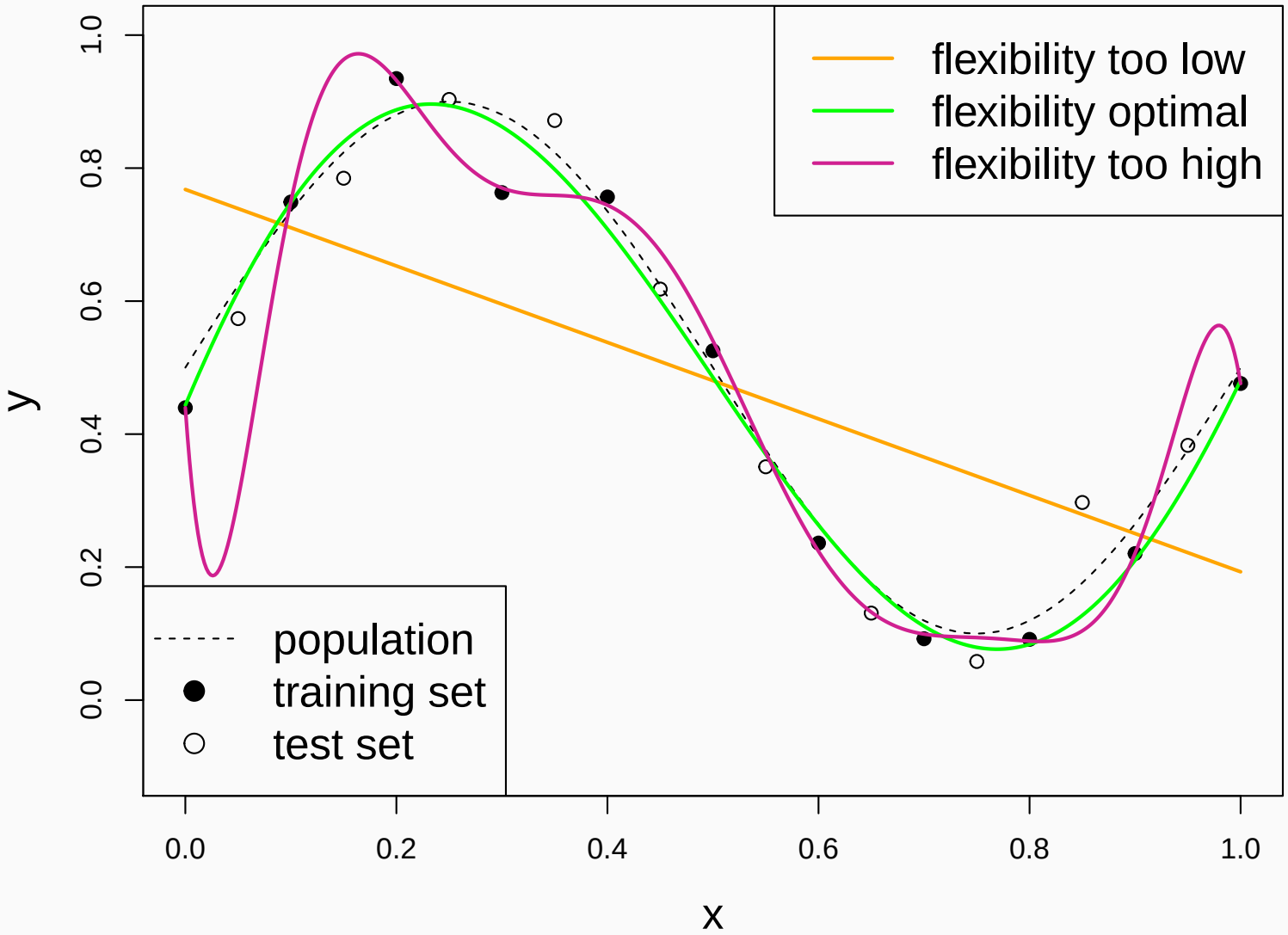
To avoid getting fooled by **Overfitting**:

- Model adjusts to a set of given data points too closely
- Sample specific patterns are learned ("fitting the noise")
- Can be compared to "learning something by heart"

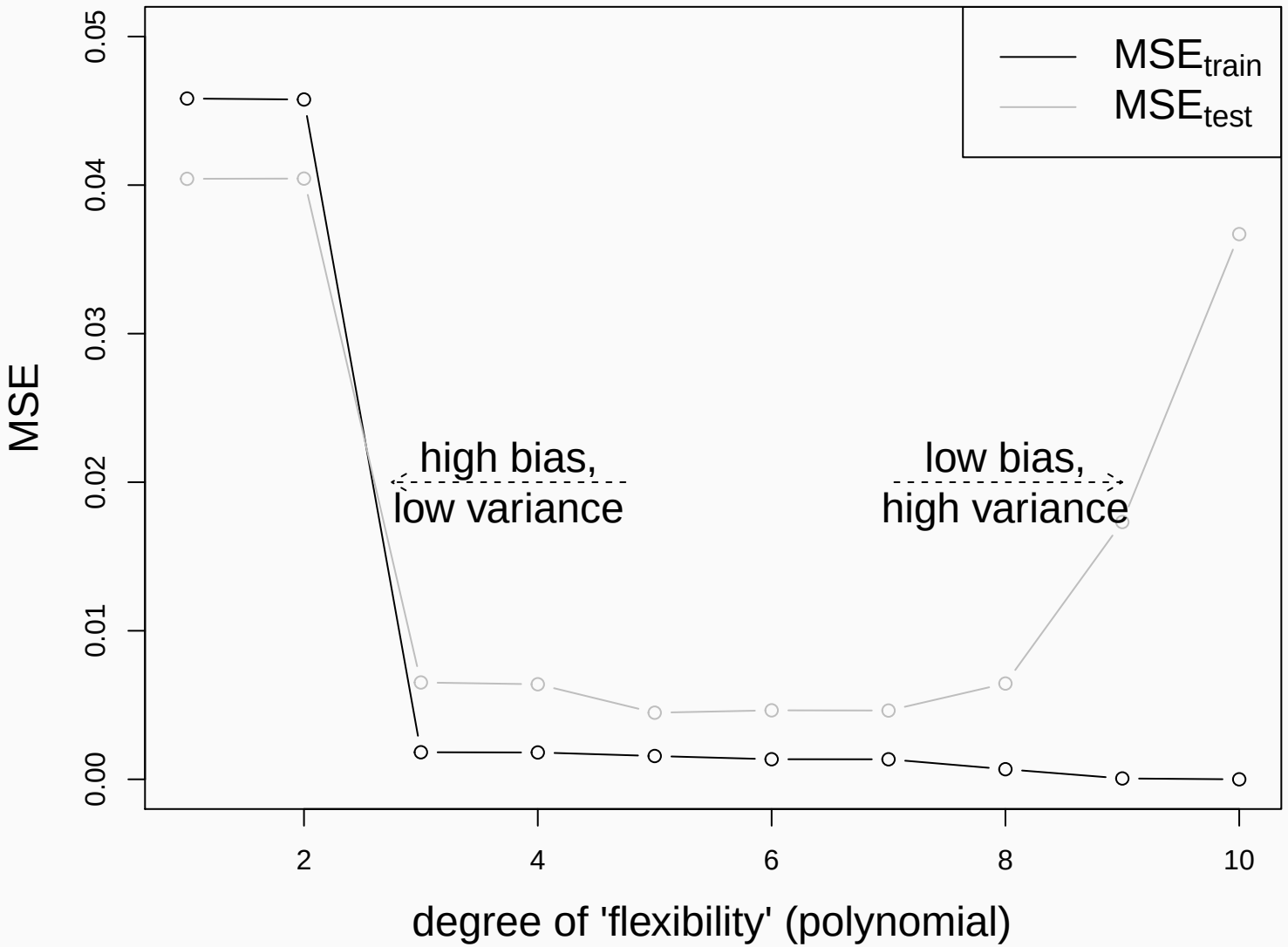
Many flexible algorithms predict training data (almost) perfectly:

> *Training ("in-sample") performance is useless to judge the model's performance on new data ("out-of-sample")!*

Example of Overfitting/Model Evaluation I



Example of Overfitting/Model Evaluation II



Training Set vs. Test Set – Dilemma

- Training set
  - as large as possible, else performance is underestimated
- Test set
  - as large as possible, else high variance in performance estimates
- *Validation set (only if you have many many observations!)*
  - *evaluation of the best final model on unused data*
- Rules of thumb
  - training set
    - large enough to learn well
    - rule: 2/3
  - test set
    - large enough for stable performance evaluation
    - rule: 1/3

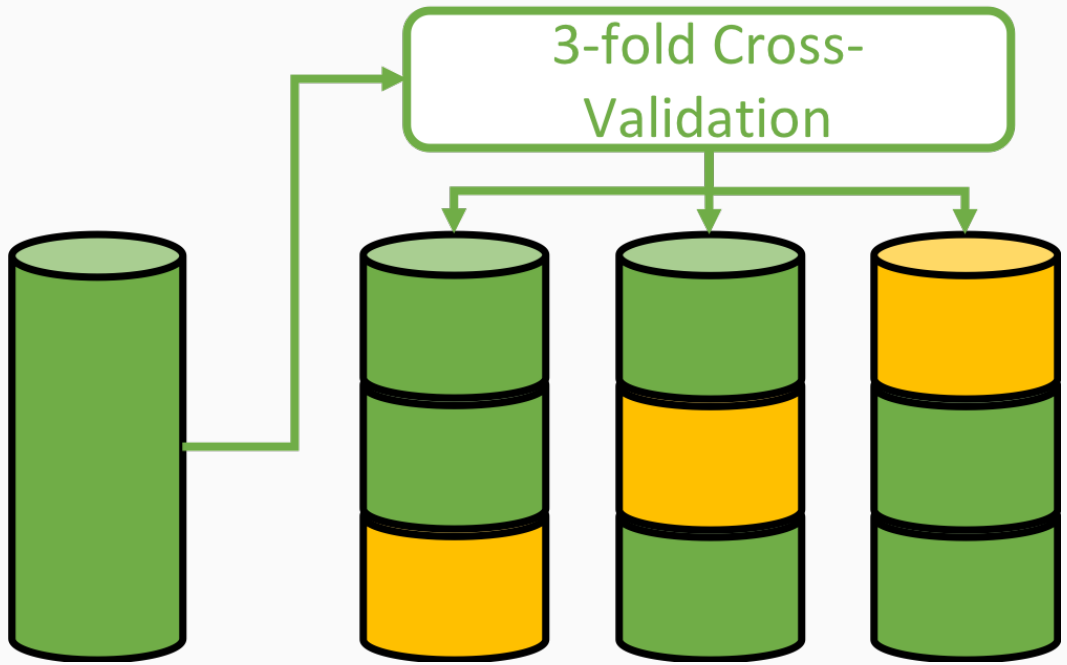
Resampling – Smart Recycling

- Optimized partitioning of the dataset:
  - several splits in training/test sets
  - aggregation of results
- Resampling methods:
  - **cross-validation (CV)**
  - repeated CV
  - leave-one-out CV
  - bootstrap
  - subsampling

(Bischl et al. 2012)

Cross-Validation (CV)

- **Bias reduction** via big training sets
- **Variance reduction** via aggregation
- Random partitioning in k equally sized parts (often 5 or 10)
- Each part test set once, remaining parts combined training set
- Average the estimated prediction error from all *folds*



Machine Learning in R

- **mlr** package (Bischl et al. 2016):
  - standardized interface for machine learning
  - detailed tutorial at https://mlr-org.github.io/mlr/

- Alternatives:
  - **caret** package (Kuhn and Johnson 2013)
  - **tidymodels** package (Kuhn and Wickham 2018)

Hands-on: Resampling

DATA: The PhoneStudy Communication Dataset

- Questionnaire data: collected in an initial lab session
- Mobile sensing data: recorded for 30 days on the smartphone

Prediction Task:

- **609 participants:** bundled from smaller studies (e.g., Stachl et al. 2017)
- **95 predictors:** aggregated communication behavior; small subset of available sensor variables
- **Continuous target:** extraversion facet Sociability; measured by the BFSI (Arendasy, Sommer, and Feldhammer 2011)

Classical Analysis (In-sample Performance)

Load the tidy **phonedata** dataframe via the *preprocessing.R* script.

```
source("preprocessing.R")
```

Compute a multiple linear regression for the target variable **Soci** (sumscore of the Sociability facet). Use all other variables as predictors. Check out the in-sample R^2.

```
summary(lm(Soci ~ ., data = phonedata))$r.squared
```

```
[1] 0.29
```

Compute In-sample Performance the mlr Way

Load **mlr** - package.

```
library(mlr)
```

Create a *learner* object. Checkout the full list of **mlr**'s learners: https://mlr-org.github.io/mlr/articles/integrated_learners.html

```
lrn <- makeLearner("regr.lm")
```

Create a *task* object which contains the necessary data.

```
task_Soci <- makeRegrTask("Sociability",
                          data = phonedata,
                          target = "Soci")
```

Train the *learner* on the *task*.

```
mod <- train(lrn, task = task_Soci)
```

Make *predictions* on the same data used for training.

```
insample_pred <- predict(object = mod, task = task_Soci)
```

Compute in-sample *performance* (R^2 and MSE).

```
insample_perf <- performance(insample_pred, measures = list(rsq, mse))
insample_perf
```

```
  rsq   mse
 0.29 25.82
```

Cross-Validation

Create a description of the *resampling* - strategy:
We use 10-fold *cross-validation* here.

```
rdesc <- makeResampleDesc("CV", iters = 10)
```

Set a *seed* to make your analysis reproducible.

```
set.seed(1)
```

Perform *cross-validation*.

```
res <- resample(learner = lrn,
                task = task_Soci,
                resampling = rdesc,
                measures = list(rsq, mse))
```

```
Resampling: cross-validation

Measures:             rsq   mse

[Resample] iter 1:    -0.05 48.49

[Resample] iter 2:    0.17  23.12

[Resample] iter 3:    -0.30 42.52

[Resample] iter 4:    0.08  36.98

[Resample] iter 5:    -0.18 47.01

[Resample] iter 6:    -0.06 37.72
```

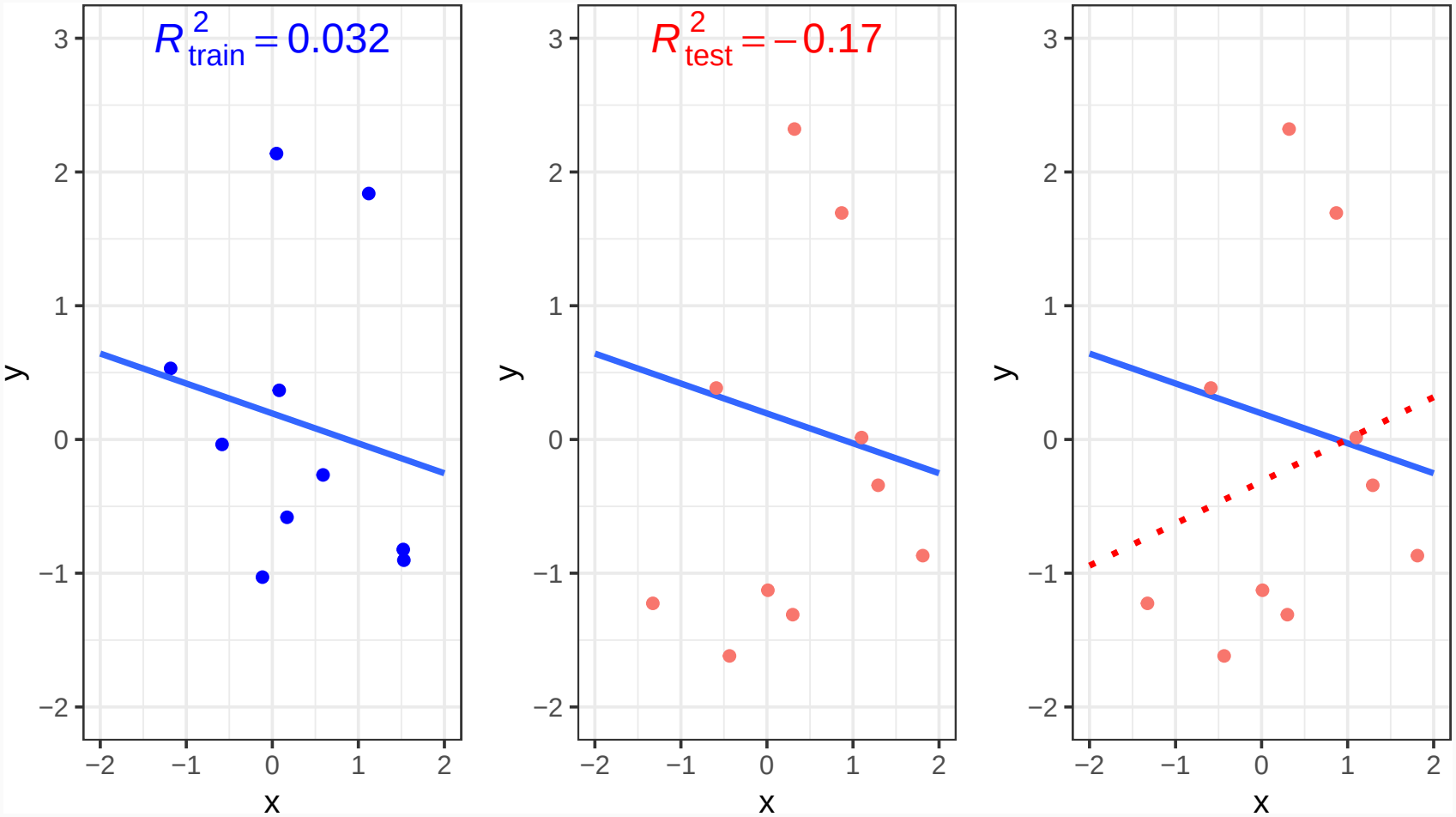
Compare your out-of-sample estimate with the in-sample performance.

```
crossval_perf <- res$aggr
matrix(c(insample_perf, crossval_perf), nrow = 2,
       dimnames = list(c("R^2", "MSE"),
                       c("in-sample", "cross-validation")))
```

```
    in-sample cross-validation
R^2      0.29           -0.049
MSE     25.82           36.617
```

Resampling shows that the prediction does not work well . . .

Negative R^2



- Train model on training data (positive R^2_{train})
- Predict test data with trained model (negative R^2_{test})

Tuning & Model Selection

Selection of the best model

Decisions in predictive modeling:

- Algorithm/type of model (e.g. linear model vs. random forest)
- Hyperparameters (e.g. *mtry* in random forests)
- Variable selection (the best predictors only?)
- Pre-processing (transformations, dimensionality-reduction, imputation)

To avoid overfitting, implement those decisions into the resampling process (i.e. repeat each time the model is trained)
-> **Simulate what happens in model application!**

If the decision itself requires resampling
-> **Nested Resampling**

Nested Resampling

- Inner loop: tuning, pre-processing, variable selection
- Outer loop: evaluation of model performance
- **Goal:** separate data used in model training from data used in model evaluation

EXKURS: Variable selection

- Common pitfall in (psychological) research:
  - correlate all predictors with the target in the complete dataset
  - choose only highly correlated predictors for modeling
- **Problem:** The decision of which variables to select is based on the complete dataset (training set + test set)
  –> **Overfitting**

Don't fool yourself! This shares similarities with:

- *multiple testing*
- *p-hacking*
- *HARKING*

Simulation - Variable Selection Done Wrong

ATTENTION: This is a recipe for overfitting

Simulate completely random data and define an arbitrary target.

```
set.seed(1)
data <- matrix(rnorm(100000), nrow = 100, ncol = 1000)
data <- as.data.frame(data)
colnames(data)[1000] <- "Target"
```

Compute correlations of all predictor variables with the target. Only retain the ten most correlated predictors, remove the others.

```
target_cors <- cor(data)[-1000, "Target"]
ten_best_vars <- names(sort(abs(target_cors),
                            decreasing = TRUE)[1:10])
library(mlr)
task_10 <- makeRegrTask(id = "ten_best",
                        data = data[,c(ten_best_vars, "Target")],
                        target = "Target")
```

Let's perform a simple regression analysis using those top-ten "best" variables to predict the target values, using the complete dataset.

```
lm <- train("regr.lm", task = task_10)
pred <- predict(lm, task = task_10)
performance(pred, measures = rsq)
```

```
 rsq
0.44
```

Oh woooow: a positive R^2, although we know that there really is no relationship between the target and those variables.

Let's try to improve on that and use some intense, repeated cross-validation.

```
rdesc <- makeResampleDesc("RepCV", folds = 10, reps = 10)
res_overfit <- resample("regr.lm", task = task_10,
                        resampling = rdesc, measures = rsq)

res_overfit
```

```
Resample Result
Task: ten_best
Learner: regr.lm
Aggr perf: rsq.test.mean=0.20
Runtime: 0.944939
```

Amazing: R^2 is still positive! This must be an important discovery!

. . . No it's not - we still overfitted our model :-(

What did we do wrong?

Solution: to obtain a realistic estimate of the predictive performance of our algorithm, the variable selection has to be integrated into the resampling process.

Let's create a learner that will choose the top-ten variables in each CV iteration respectively (those can be different).

```
lrn <- makeFilterWrapper(learner = "regr.lm",
                         fw.method = "linear.correlation", fw.abs = 10)
```

Run the resampling with the **wrapped learner** on the complete dataset, using all variables.

```
task_1000 <- makeRegrTask(id = "all_vars",
                          data = data, target = "Target")
res_nested <- resample(lrn, task = task_1000, resampling = rdesc,
                       measures = rsq)
```

```
res_nested
```

```
Resample Result
Task: all_vars
Learner: regr.lm.filtered
Aggr perf: rsq.test.mean=-0.54
Runtime: 4.79282
```

Result: negative R^2 – we did worse than predicting with the target mean of the respective test sets and ignoring all predictors.

LUNCHTIME

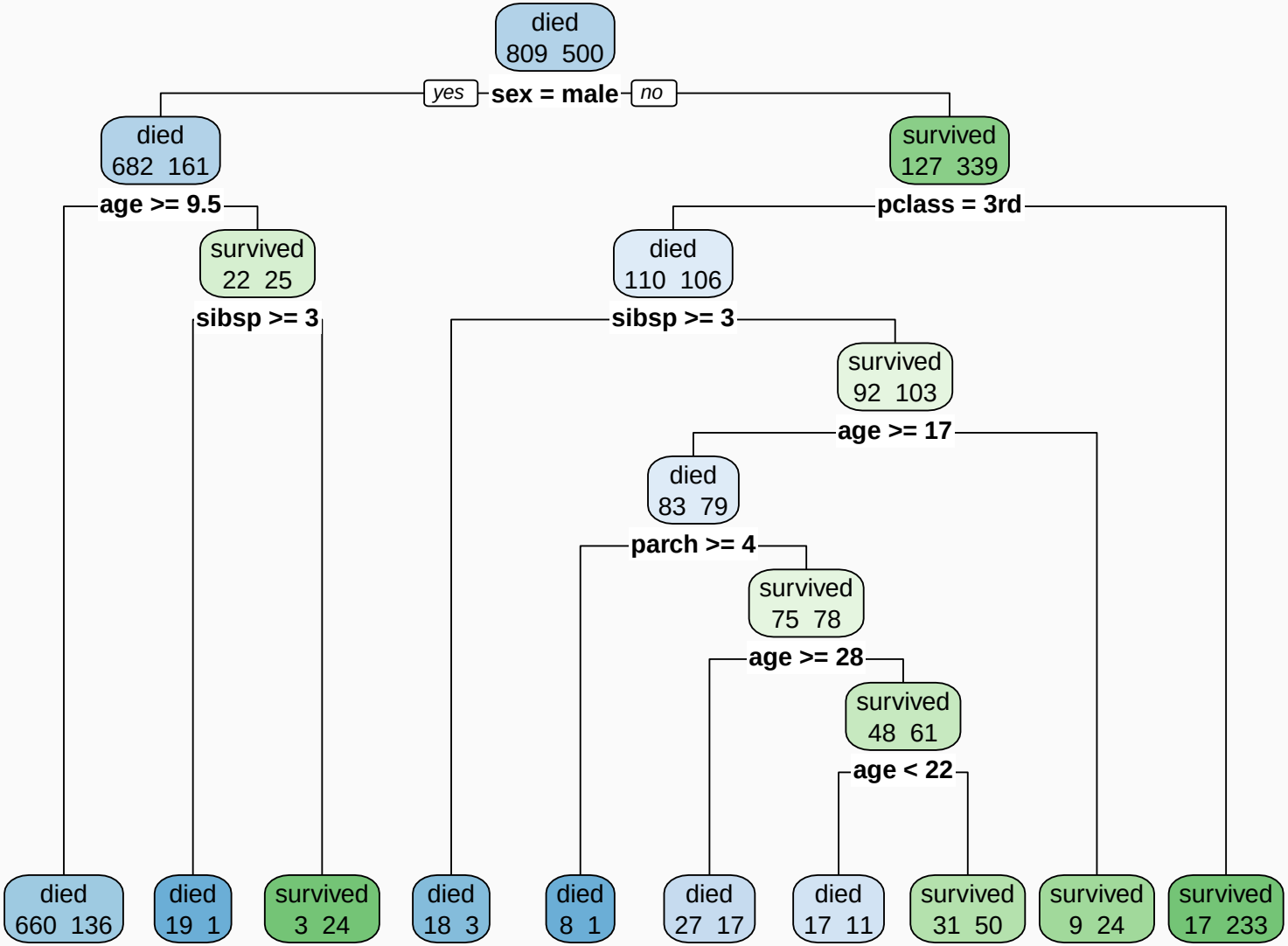
Save your workspace (then you don't have to reenter your task!)

Agenda Afternoon

- Decision Trees & Random Forests
- **Hands-on: Train a Random Forest**
- **Hands-on: Perform a Benchmark Experiment**
- Variable Importance + Partial Dependence Plots
- Take Home Message

Decision Trees and Random Forest

Example: Titanic dataset



Classification and Regression Trees (Breiman et al. 1984)

- Use predictor variables to iteratively partition the data space into single nodes
- Constant prediction within nodes (mean or majority vote)
- Optimization criterion determines split-variables and split-points simultaneously
- **Goal:** maximize "purity" within nodes
- Node-purity is defined by an **impurity** function, for example:
  - **MSE** (regression) = variance in target
  - **MMCE** (classification) = proportion of smaller class

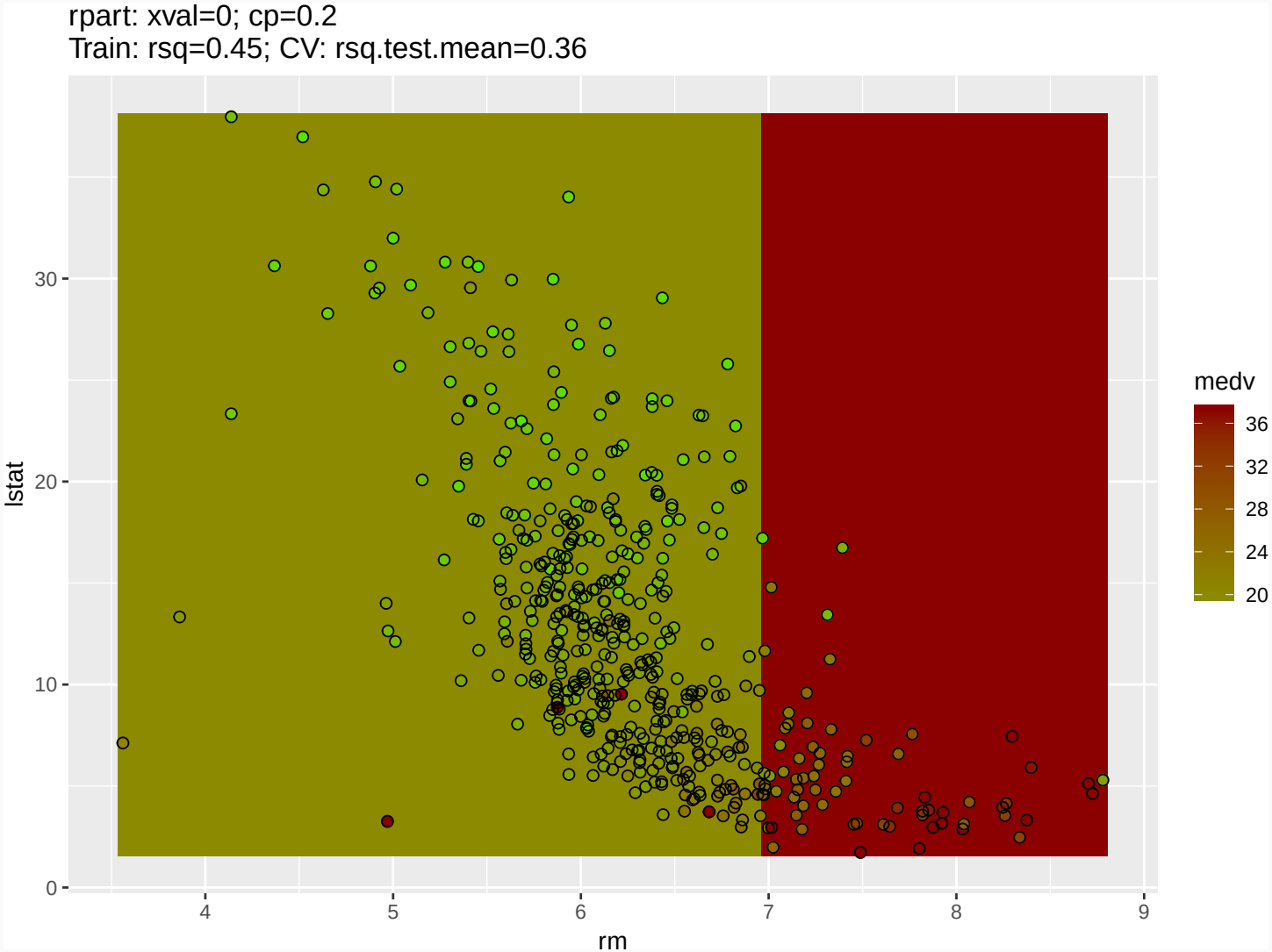
CART - Tree Growing Algorithm

- Start with constant predictions in the **root** node
- Calculate impurity for all combinations of split-variables and split-points
- Choose combination with a maximum reduction of node-impurity compared to the **parent** node (while accounting for the size of the **child** nodes)

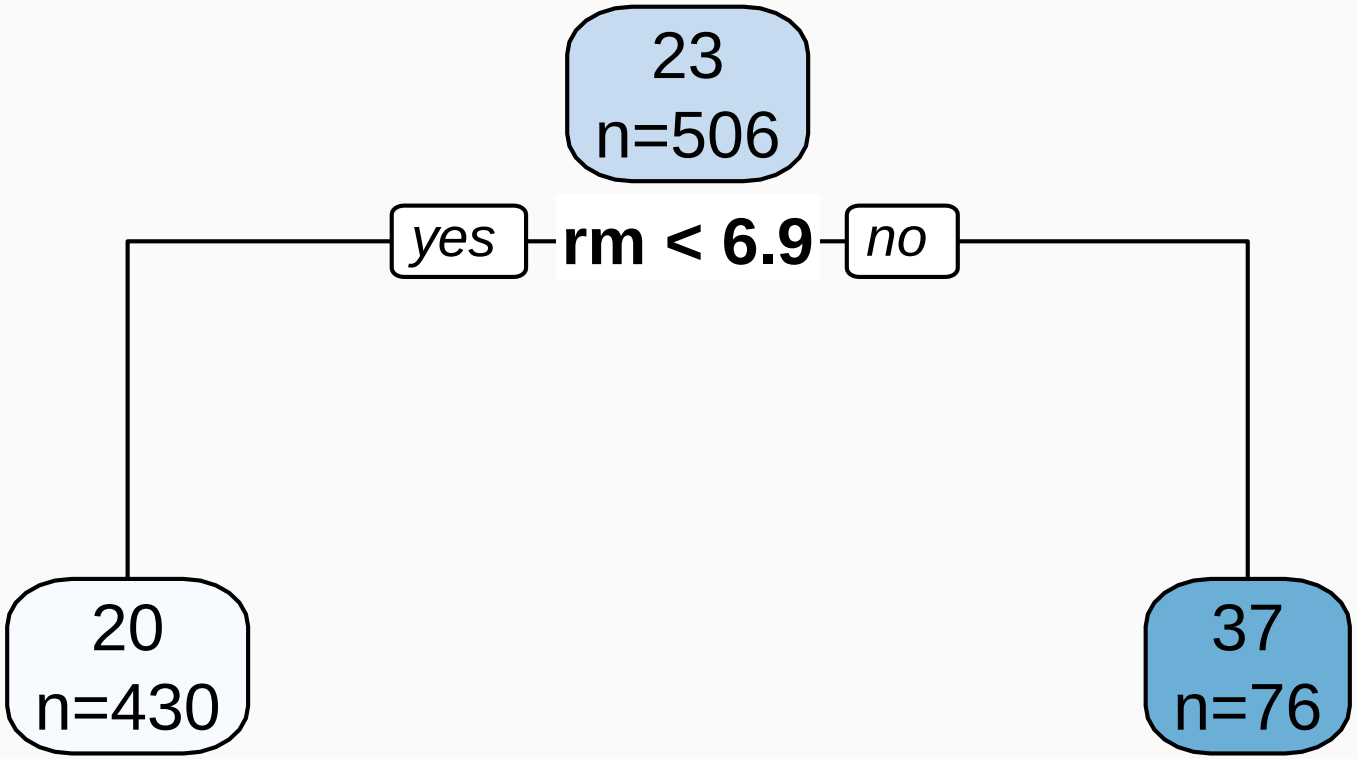
$$IR = Imp(P) - \frac{|L|}{|P|} Imp(L) - \frac{|R|}{|P|} Imp(R)$$

- Iterative splitting until a pre-defined **stopping-criterion** is met
- **Please note**: CART uses the **Gini - impurity** in classification

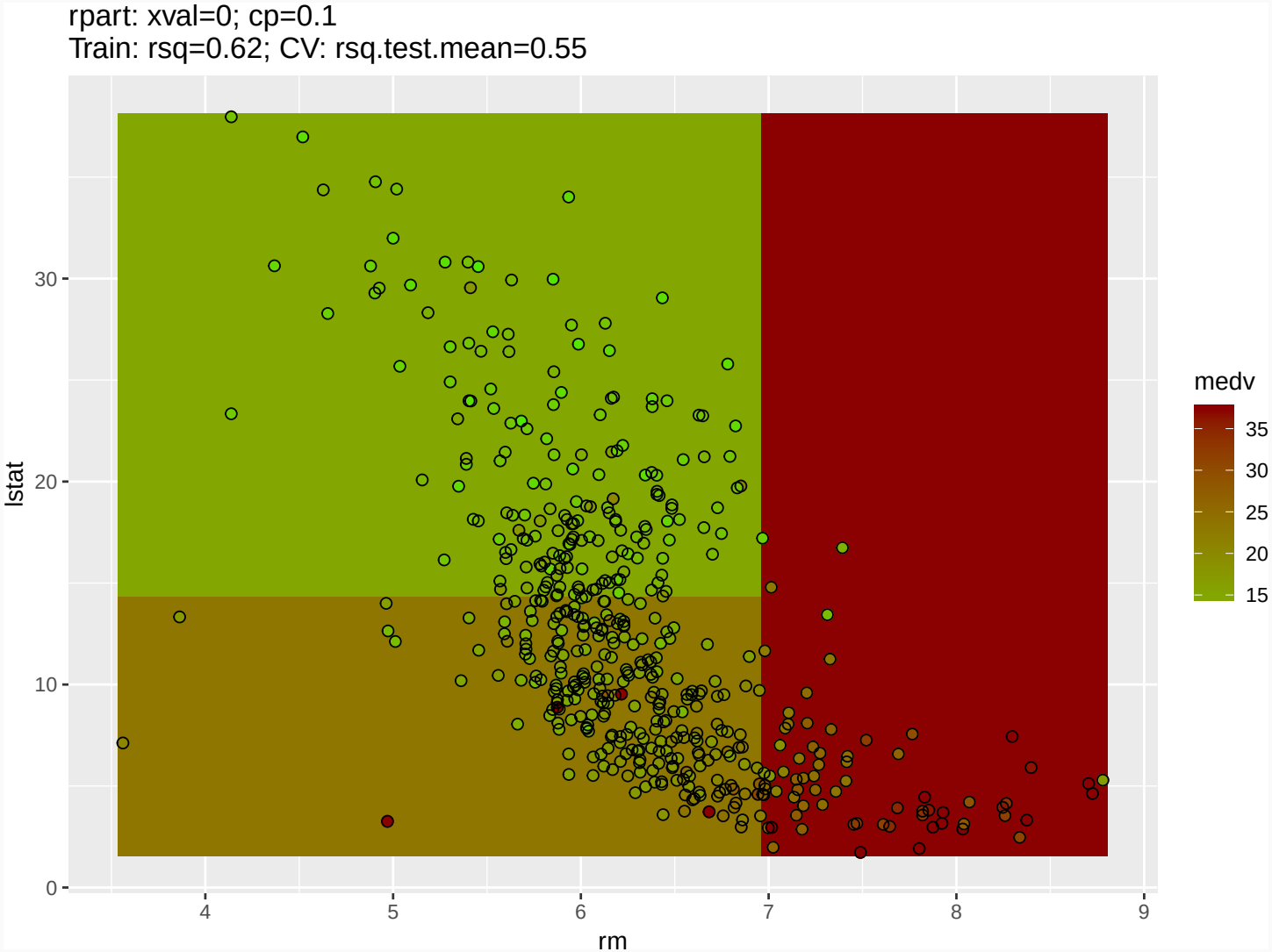
Example: 1 Split



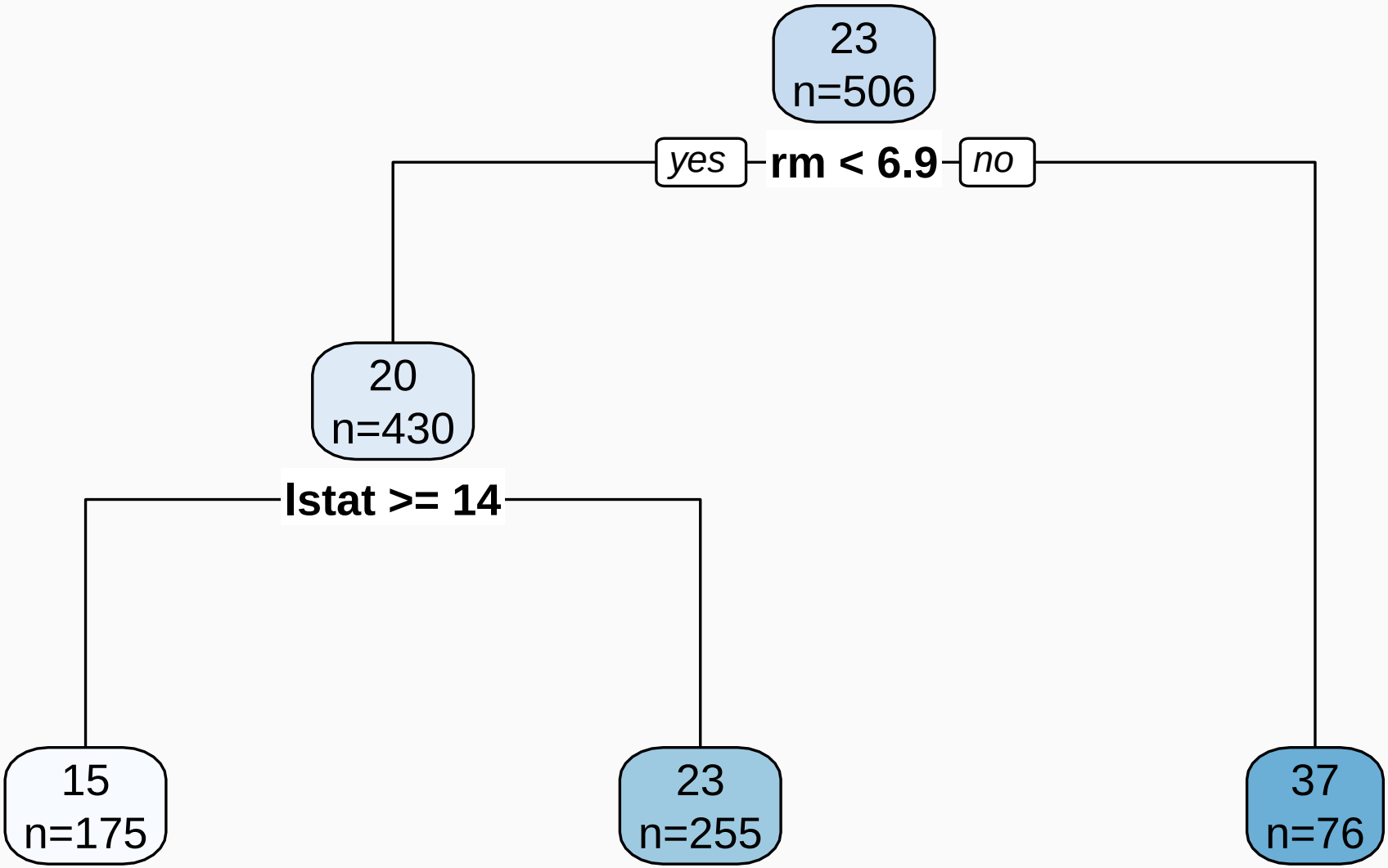
Example: 1 Split



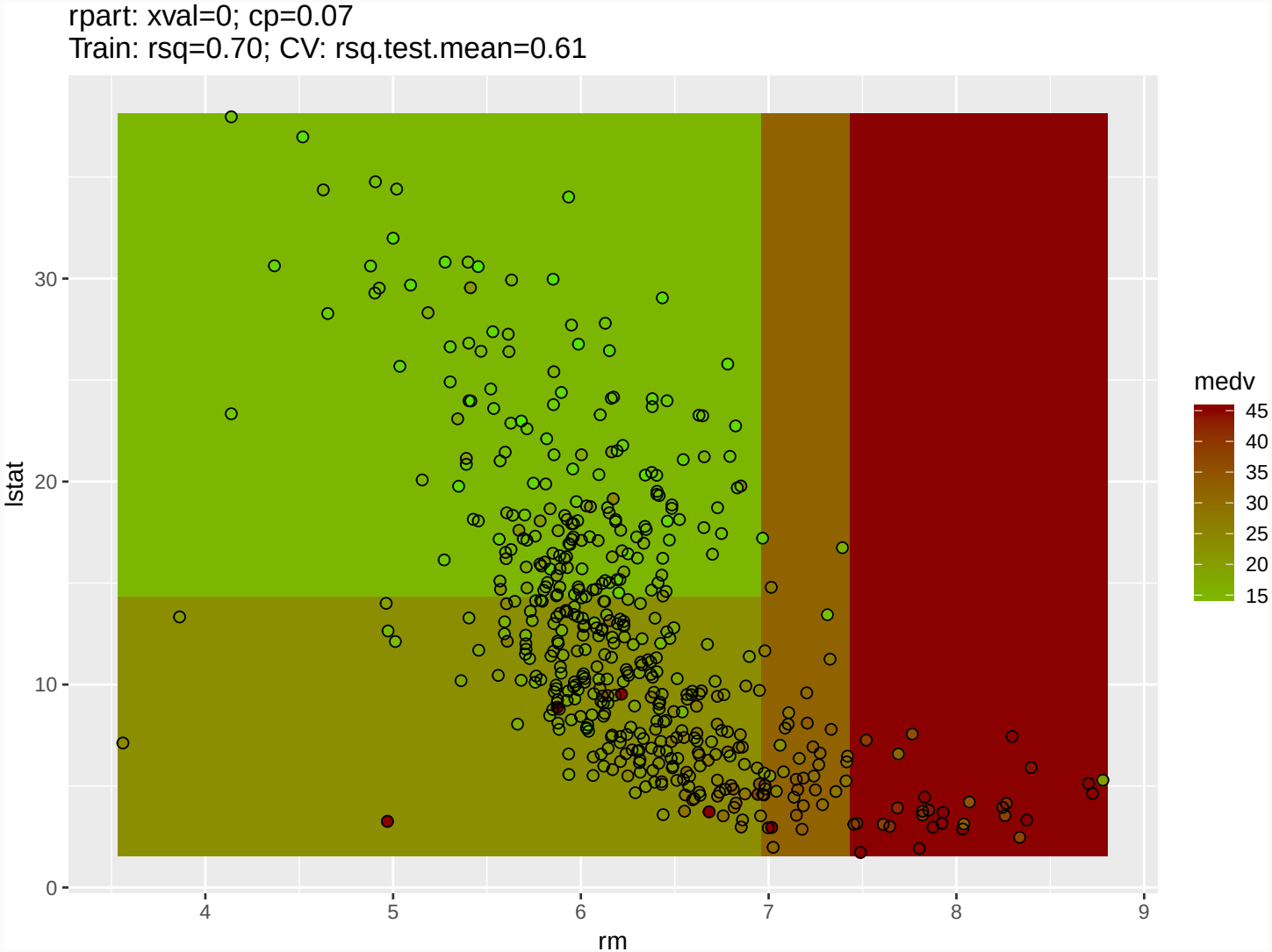
Example: 2 Splits



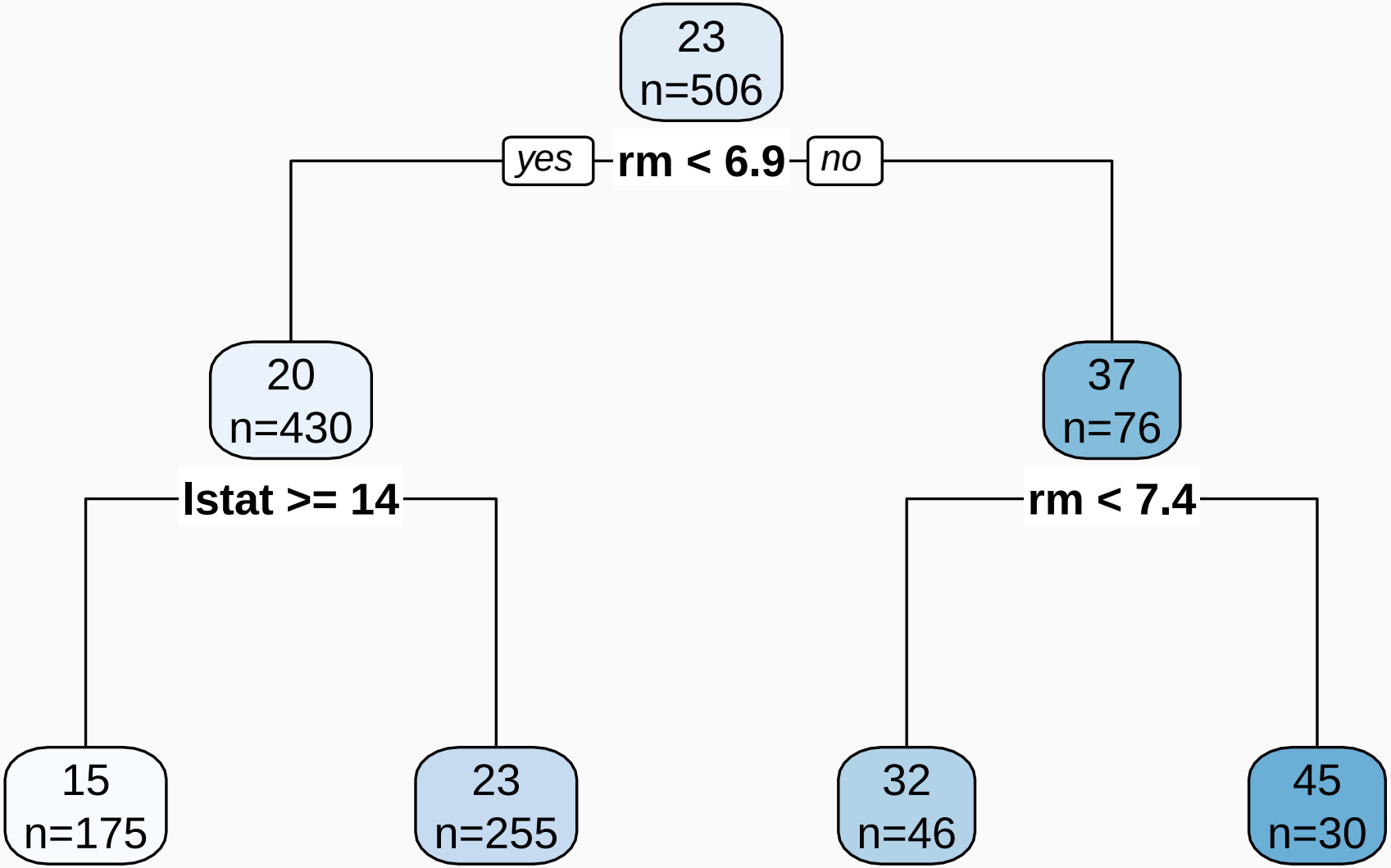
Example: 2 Splits



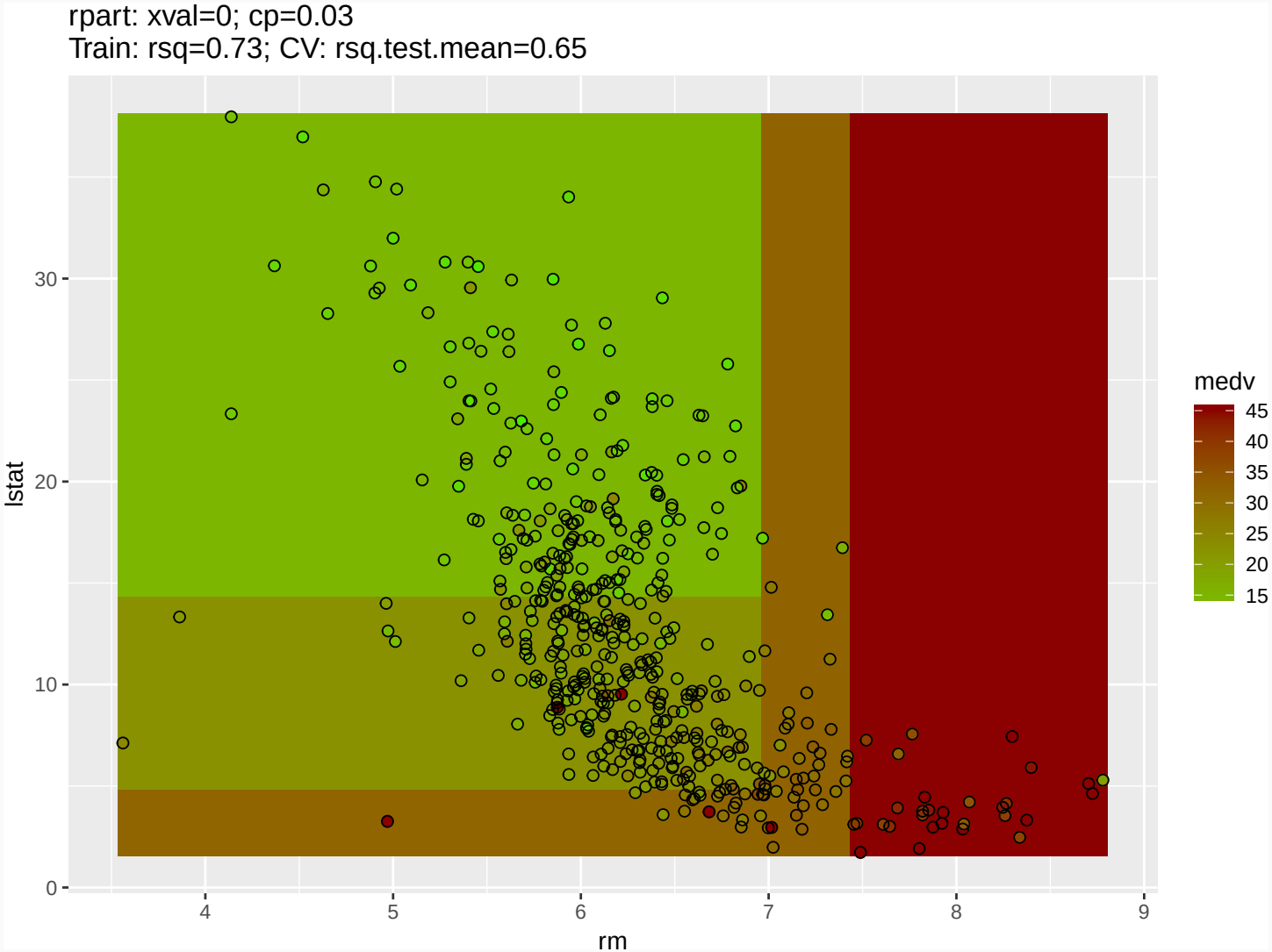
Example: 3 Splits



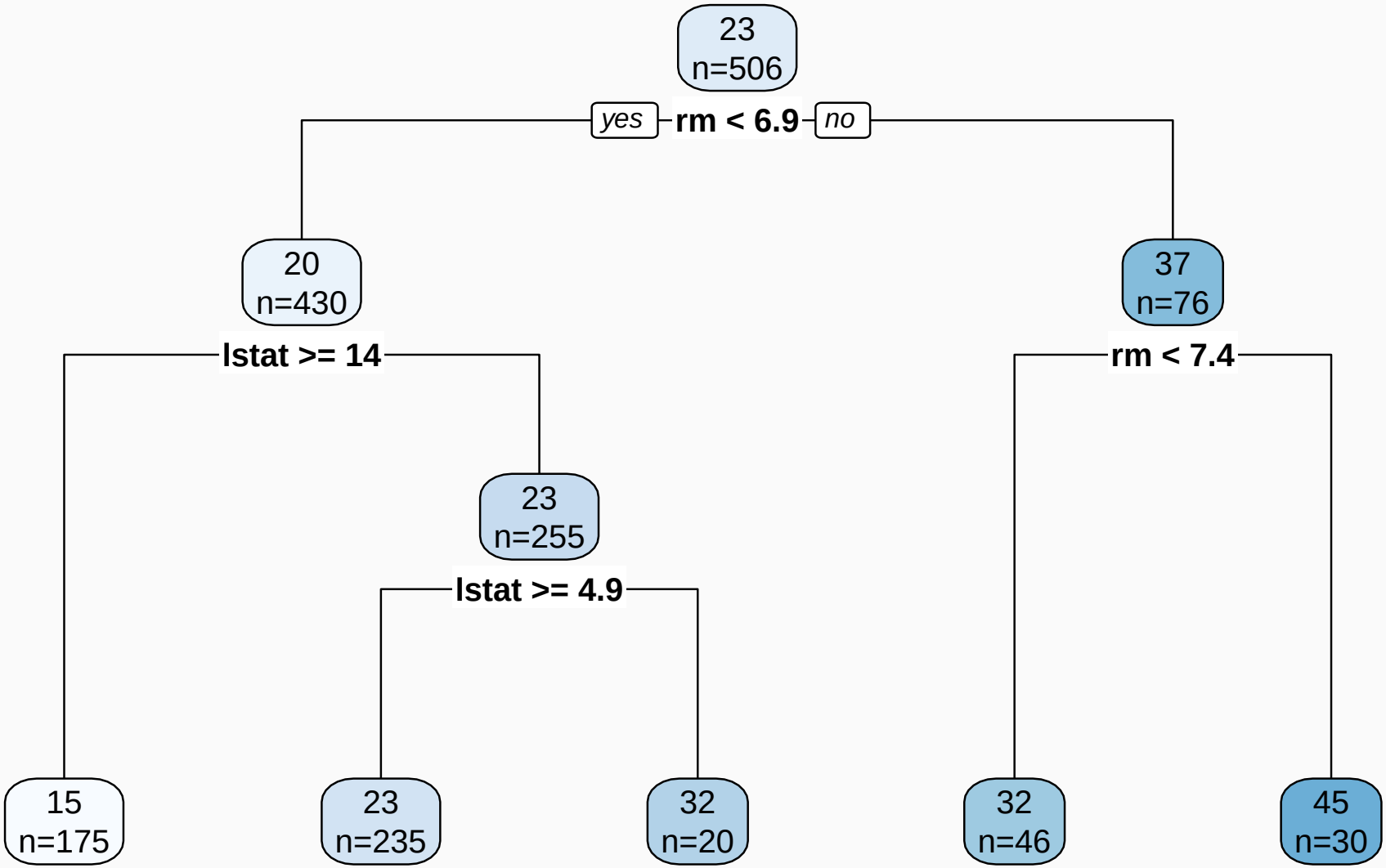
Example: 3 Splits



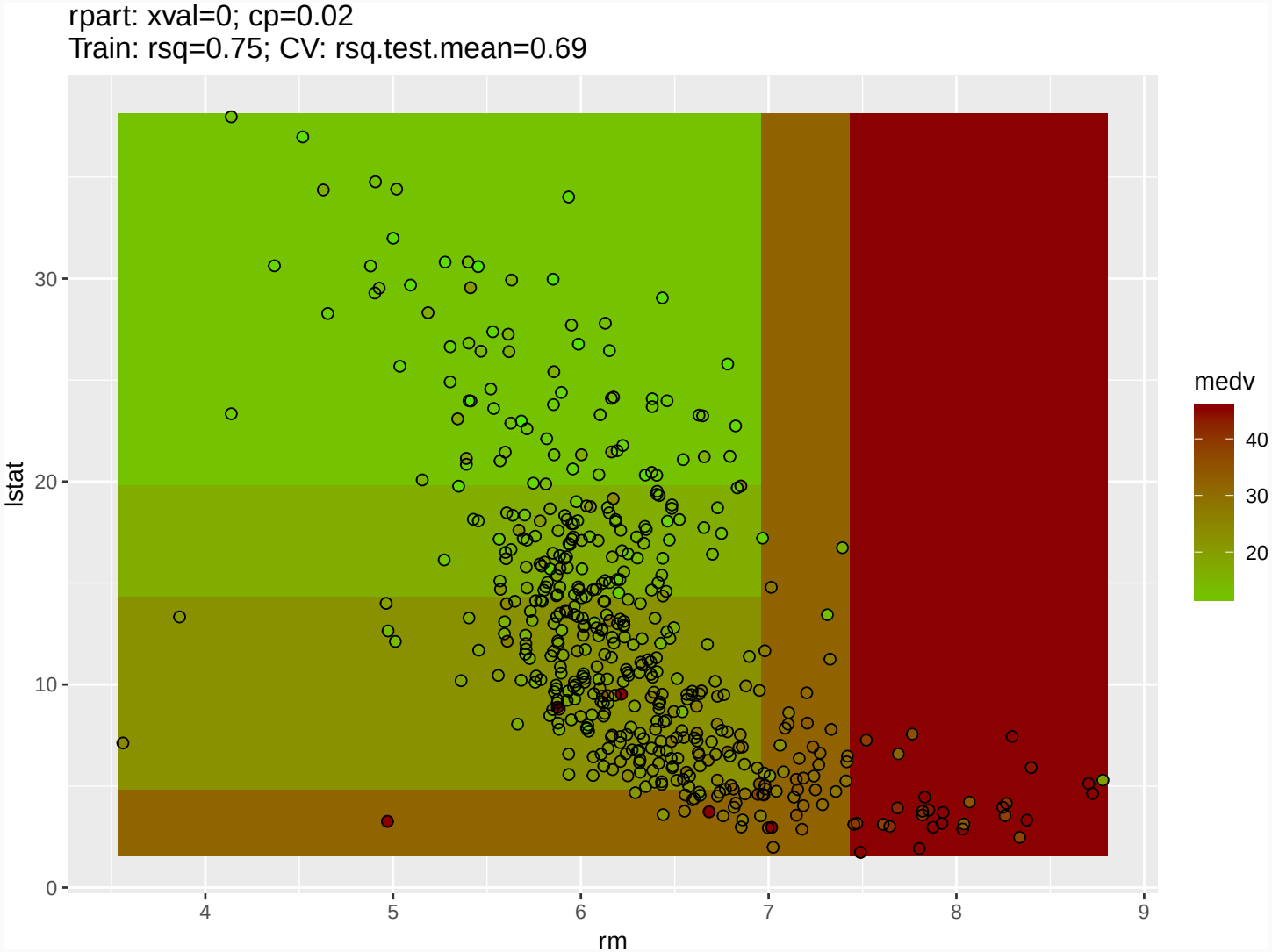
Example: 4 Splits



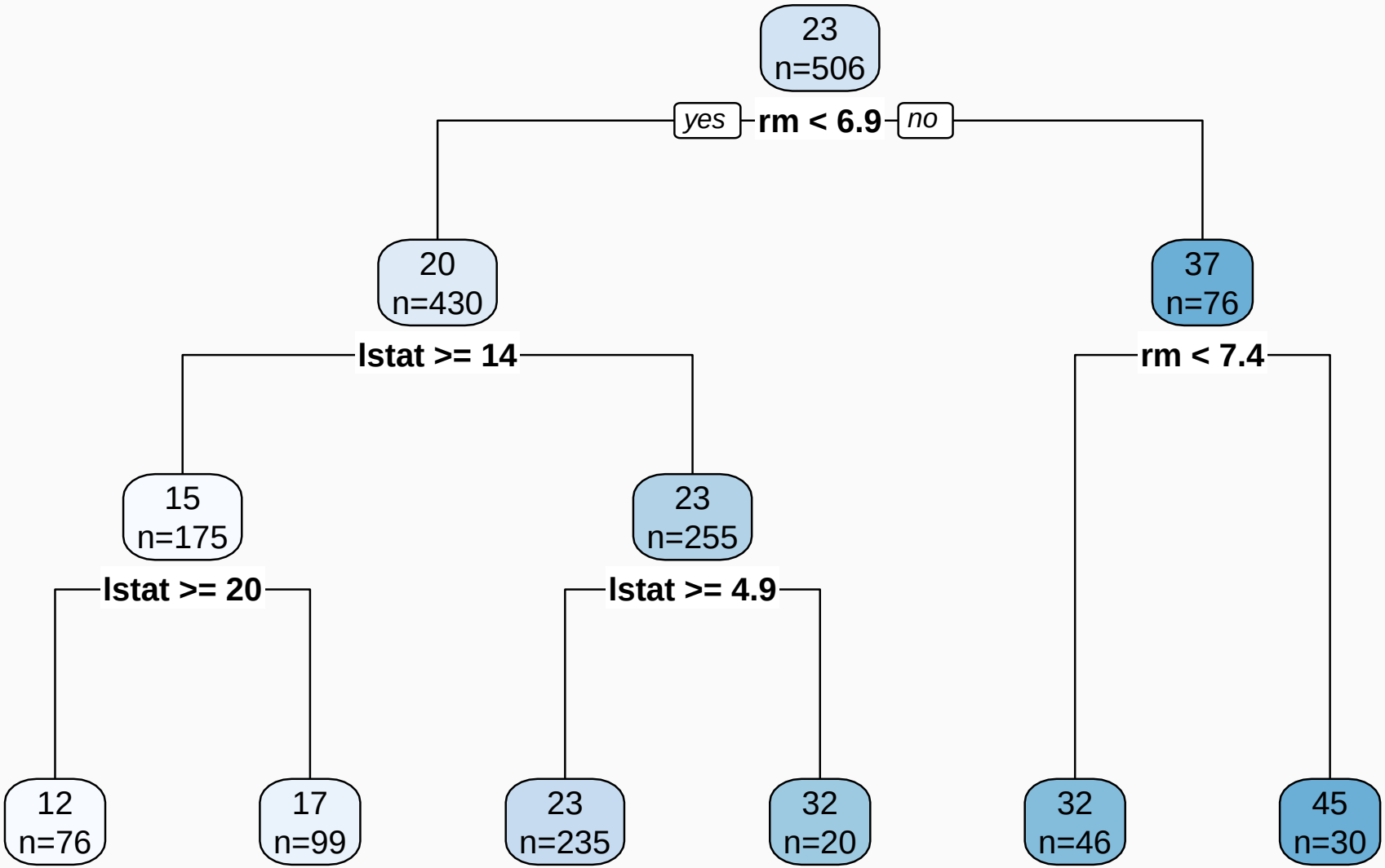
Example: 4 Splits



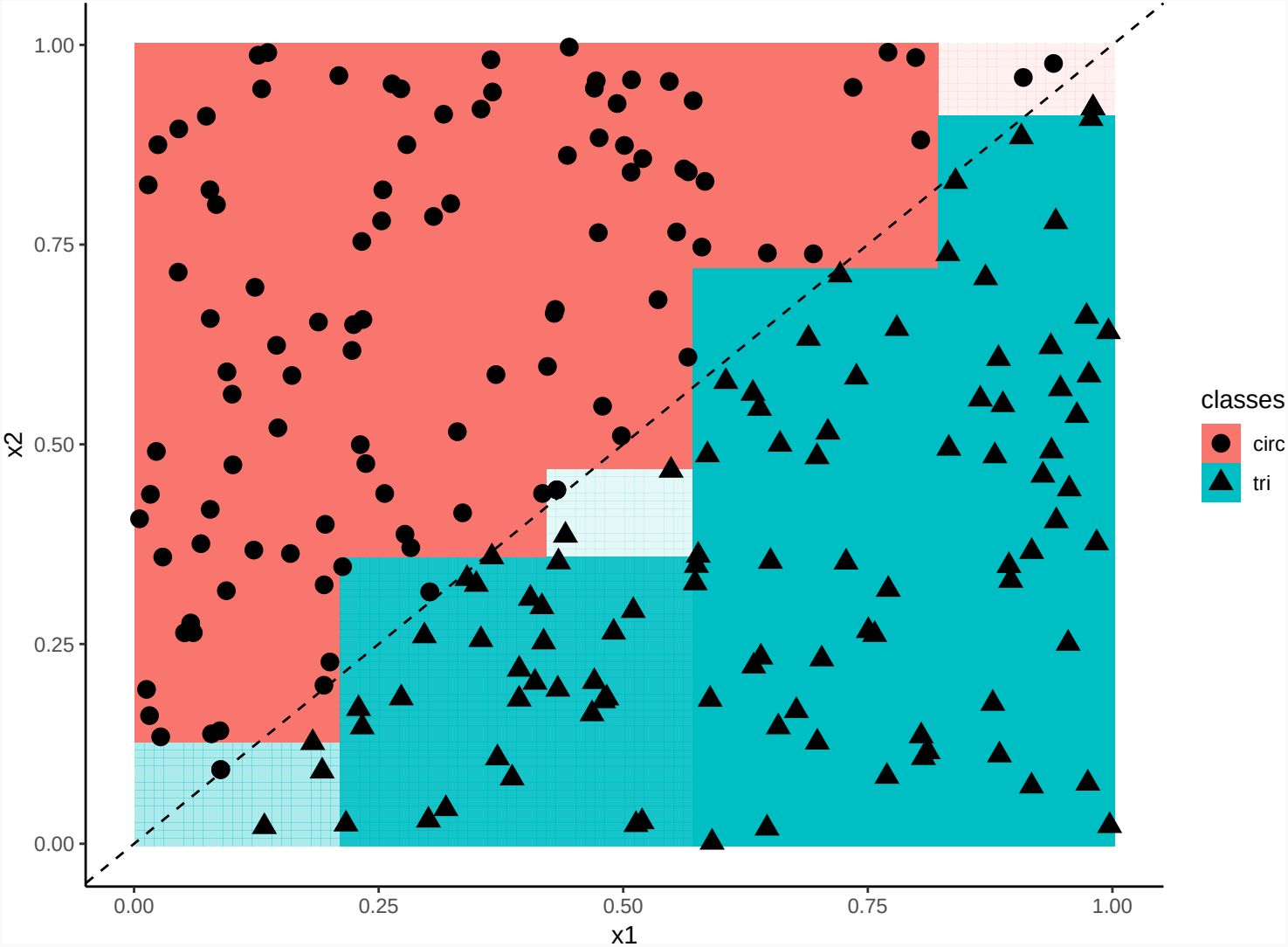
Example: 5 Splits



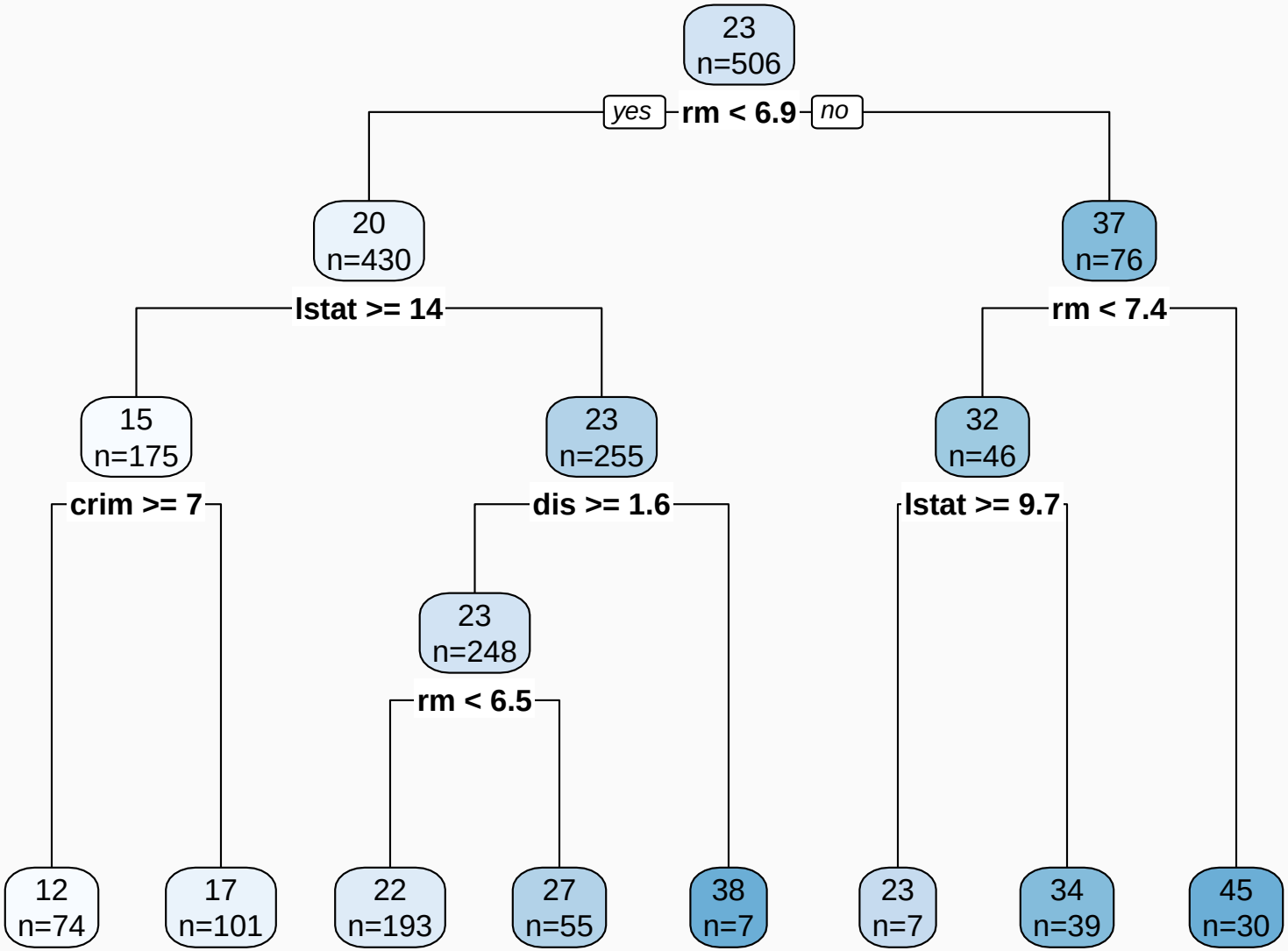
Example: 5 Splits



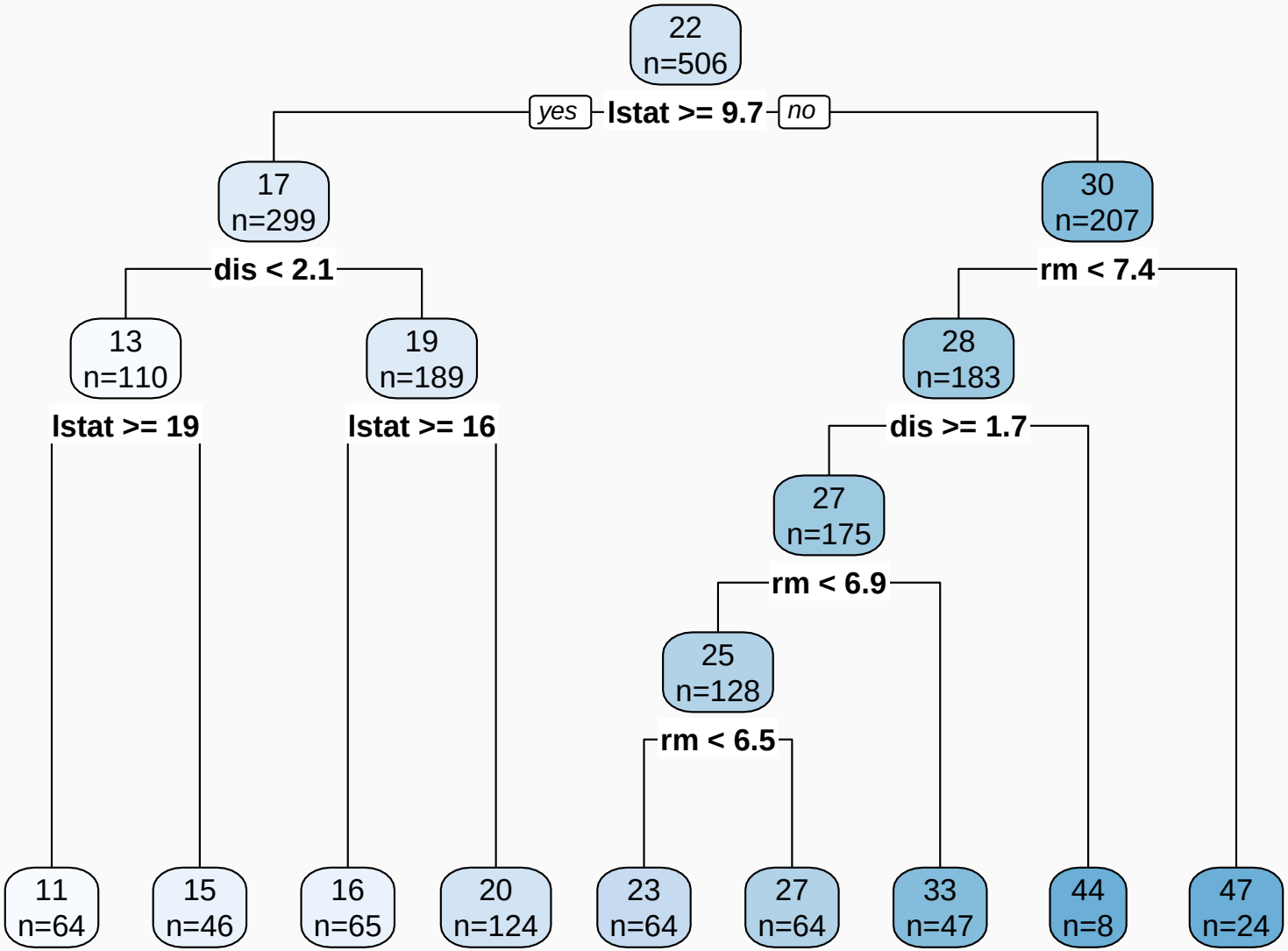
Trees Have Problems with Linearity



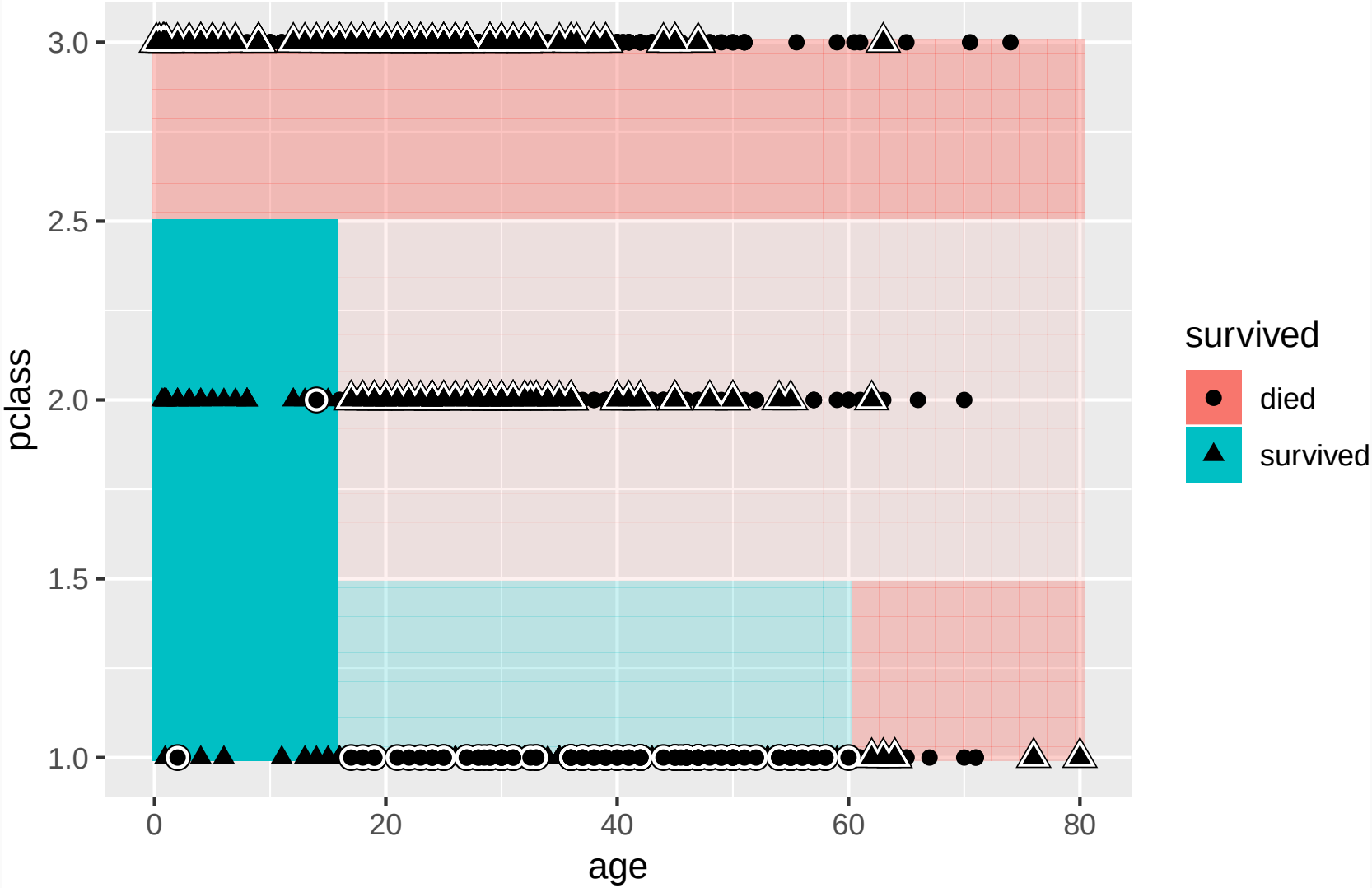
Instability of Trees (Boston Housing)



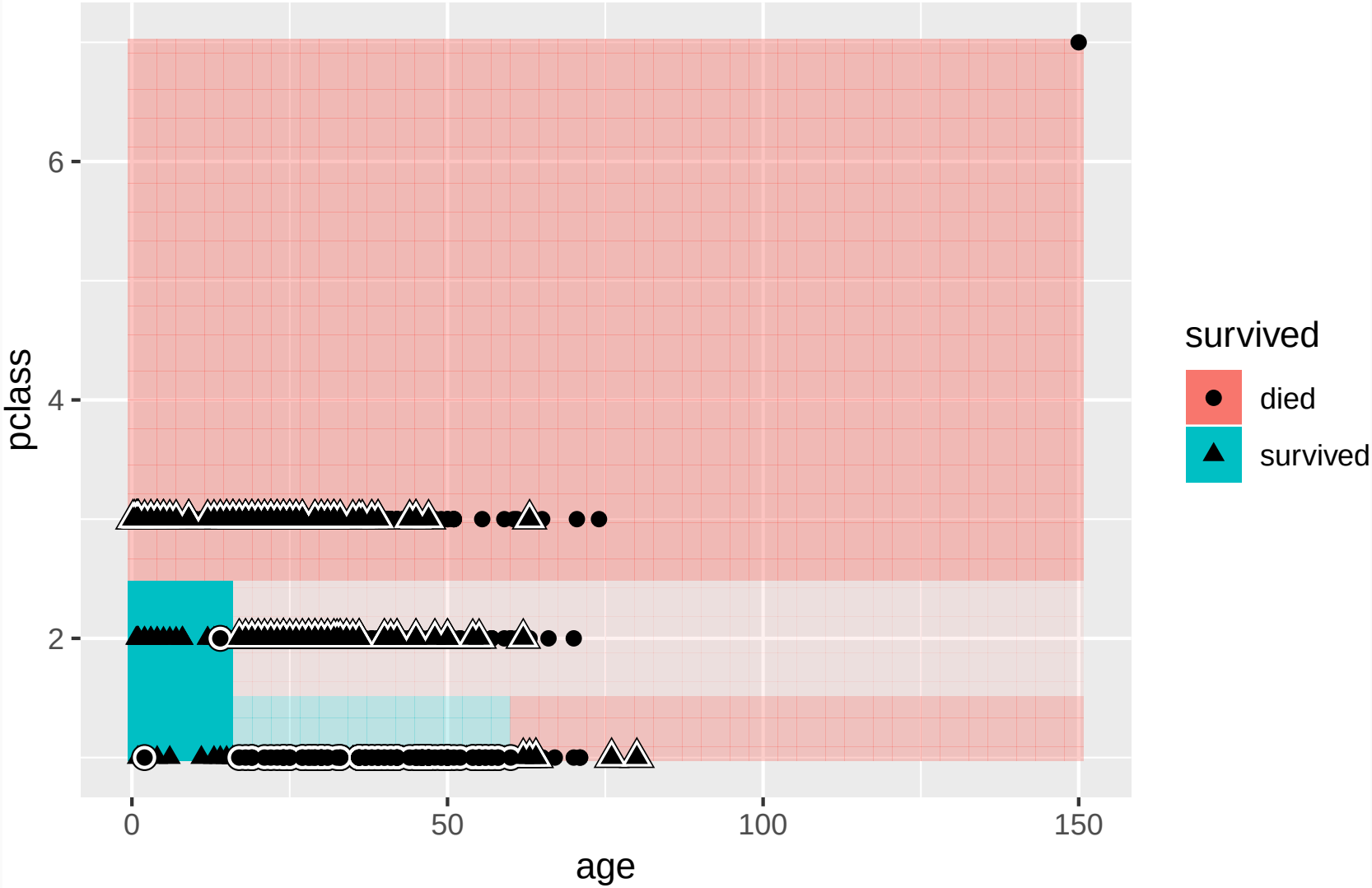
Instability of Trees: Different Structure on Bootstrap Sample



Trees Can Deal with Discrete Predictors (Titanic)



Trees Are Robust with Outliers in Predictors (Titanic)



Pros & Cons – Decision Trees

- Pros:
  - intuitive illustration of interactions
  - robust against outliers (in the predictors)
  - can model non-linear relationships
  - automatic variable selection
  - effective with a large number of predictors
- Cons:
  - truly linear relationships can only be approximated
  - tree-structure is unstable (interpret carefully!)
  - stopping-criteria need to be selected wisely (tuning!)

    -> poor predictive performance

Tree Bagging - Bootstrap Aggregation

Goal: improve predictive performance by variance reduction

-> aggregate predictions of many trees

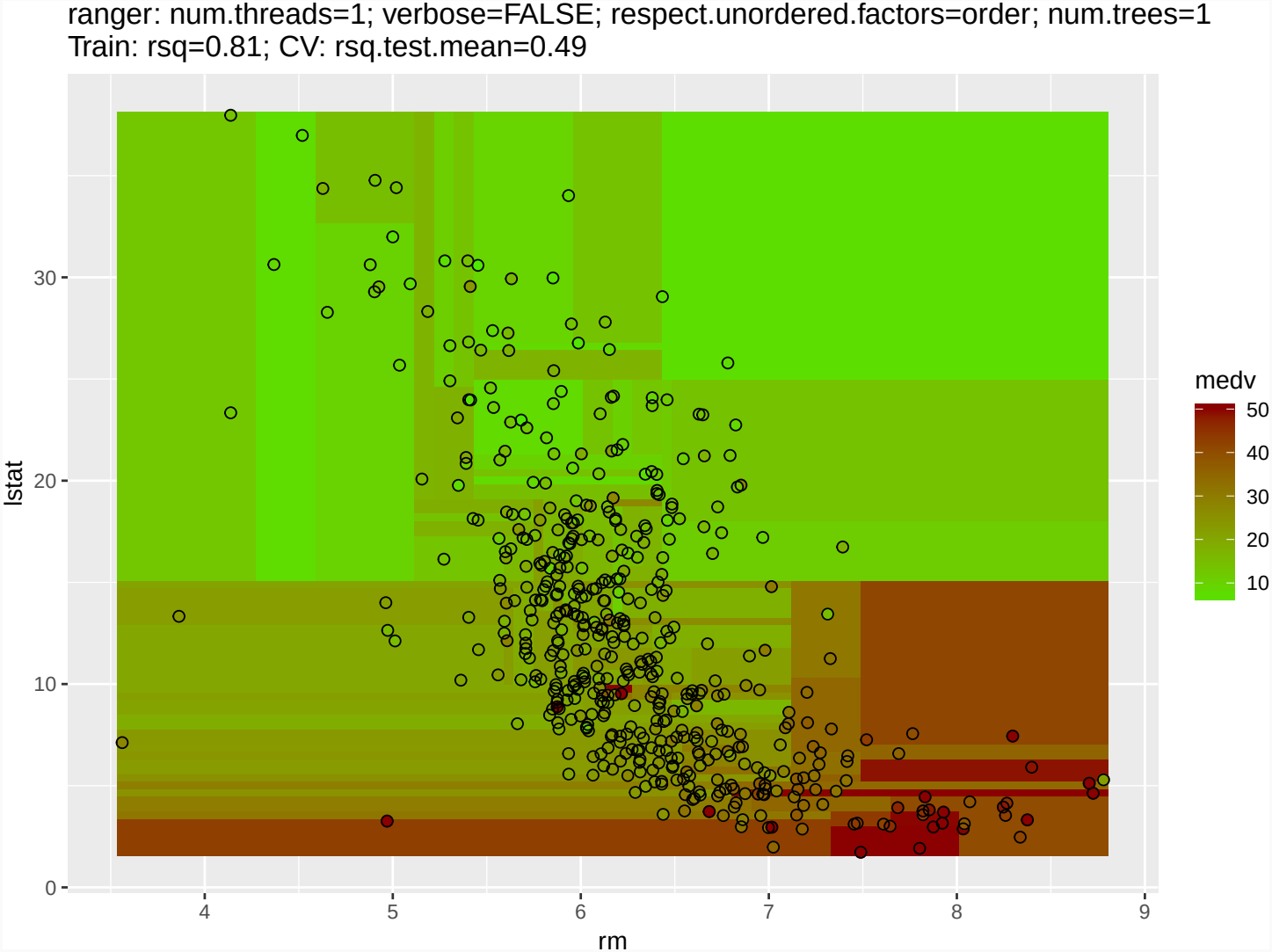
- Algorithm:
  - draw B bootstrap samples (with replacement)
  - fit deep decision tree on each bootstrap sample (liberal stopping criteria)
  - aggregate predictions across all B trees: mean (regression) or majority vote (classification)
- Limitation: highly correlated trees
  - good predictors are often selected early
  - variance reduction works best for uncorrelated trees

Random Forest (Breiman 2001)

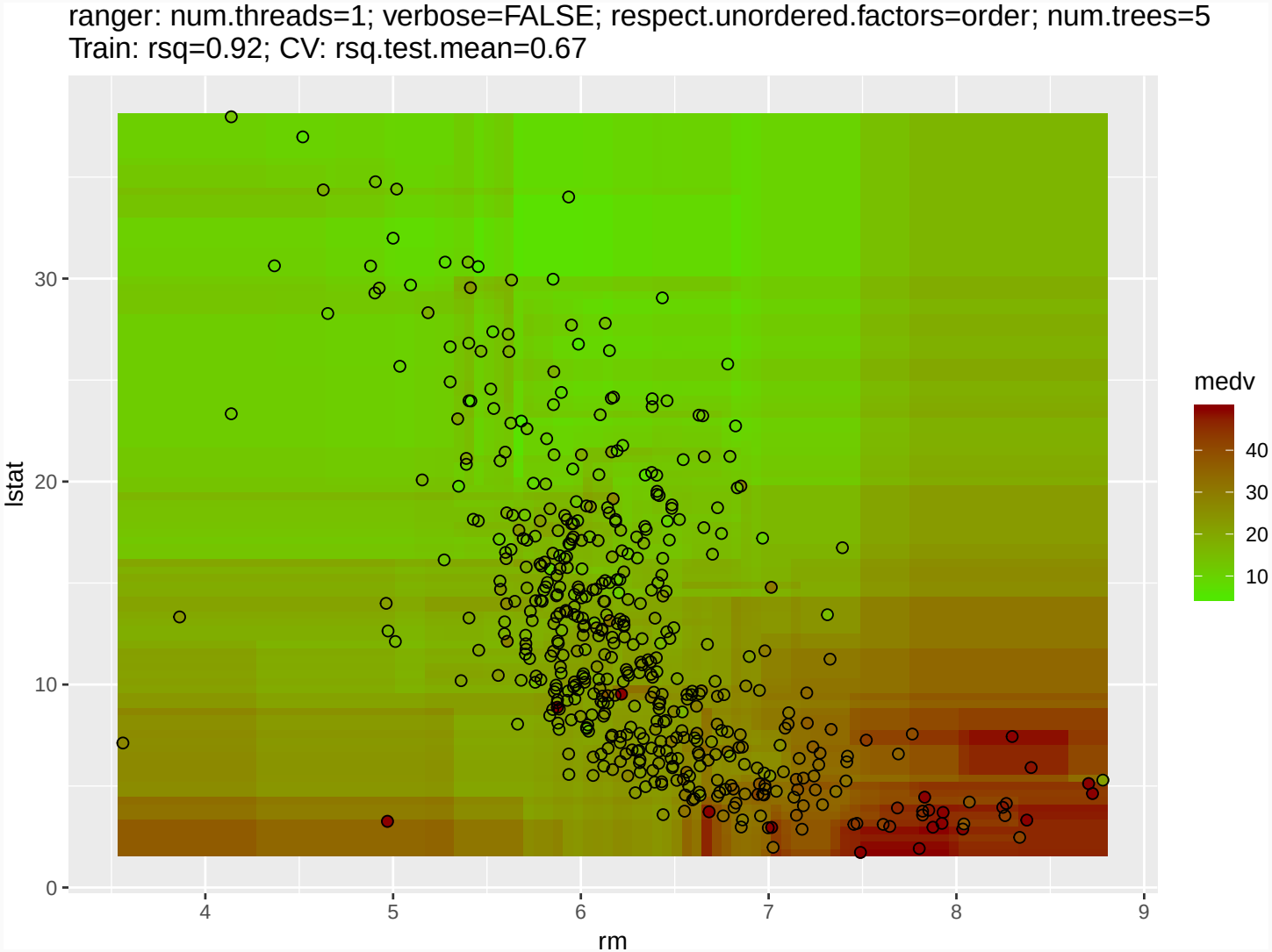
Goal: improve variance reduction by reducing the correlation between trees

- Random subset of predictors at each split
- Hyperparameters:
  - $mtry$ = number of predictors to consider at each split
  - $min.node.size$ = minimal node size to continue splitting
- Rules of thumb:
  - classification ($mtry = \sqrt{p}, min.node.size = 1$)
  - regression ($mtry = \frac{p}{3}, min.node.size = 5$)
- Alternative: determine optimal values by **tuning**

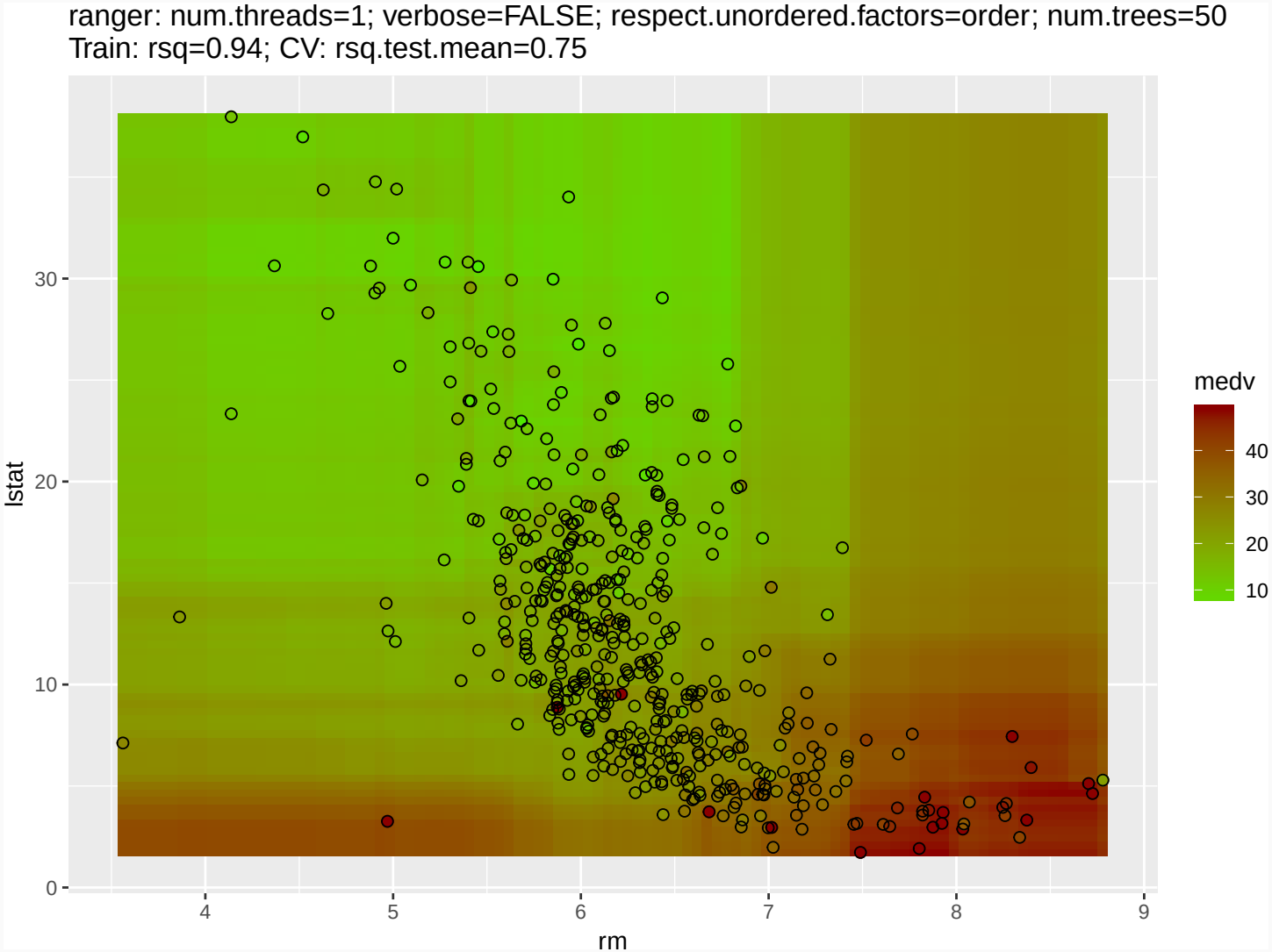
Random Forest (1 tree)



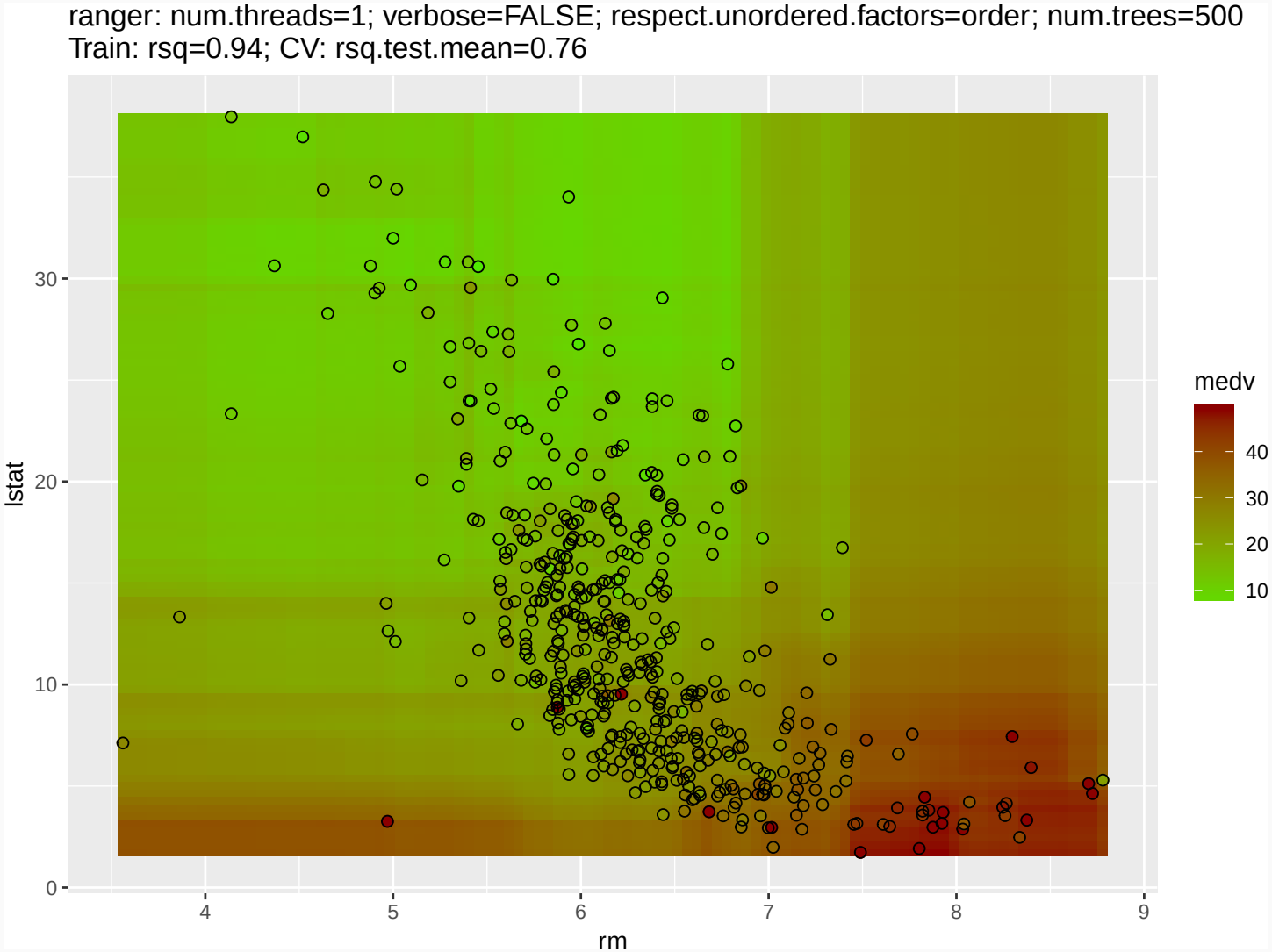
Random Forest (5 trees)



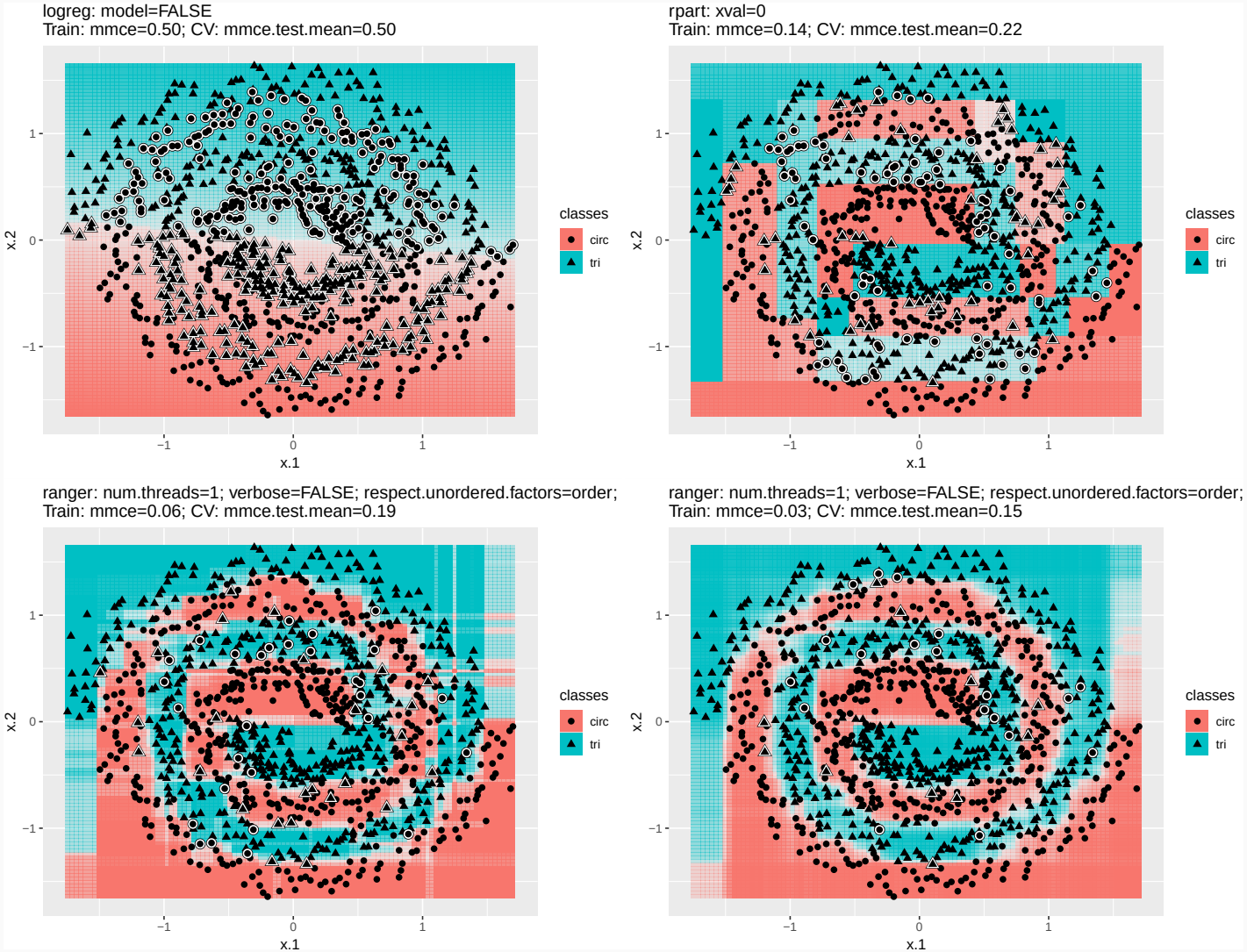
Random Forest (50 trees)



Random Forest (500 trees)



Nonlinear Decision Boundaries



Pros & Cons - Random Forest

- Pros:
  - one of the best "off-the-shelf" algorithms
  - inherits all advantages of decision trees
  - better performance than single trees
  - better handles nonlinearity and interactions
  - low bias, low variance
  - no overfitting (you can't fit too many trees)
  - tuning often not necessary
- Cons:
  - still not optimal for truly linear regression tasks
  - harder to interpret (additional tools necessary)

Hands-on: Random Forest Training

Prepare the data

We will start with a classification exercise. As our *target* is originally continuous, we categorize into high and low sociability. For the sake of convenience, we split at the median.

```
phonedata$Soci_bin <-
  ifelse(phonedata$Soci >= median(phonedata$Soci), "high", "low")
phonedata$Soci_bin <- as.factor(phonedata$Soci_bin)
```

Let's create a *task* object for classification.

```
task_Soci_bin <-
  makeClassifTask(id = "Sociability_Classif",
                  data = phonedata[, -which(names(phonedata)=="Soci")],
                  target = "Soci_bin",
                  positive = "high")
```

```
task_Soci_bin


Supervised task: Sociability_Classif
Type: classif
Target: Soci_bin
Observations: 609
Features:
   numerics     factors     ordered functionals
         95           0           0           0
Missings: FALSE
Has weights: FALSE
Has blocking: FALSE
Has coordinates: FALSE
Classes: 2
high  low
 336  273
Positive class: high
```

Train the model

Create a *learner* for the random forest classifier.

```
lrn <- makeLearner("classif.ranger", num.trees = 500)
```

Train the *learner* on the *task*.

```
set.seed(1)
rf <- train(lrn, task_Soci_bin)
rf
```

```
Model for learner.id=classif.ranger; learner.class=classif.ranger
Trained on: task.id = Sociability_Classif; obs = 609; features = 95
Hyperparameters: num.threads=1,verbose=FALSE,respect.unordered.factors=
```

Inspect the trained model

Extract the random forest model which was computed by the **ranger** package (Wright and Ziegler 2017) as we *trained* the learner.

```
getLearnerModel(rf)
```

```
Ranger result

Call:
 ranger::ranger(formula = NULL, dependent.variable = tn, data = getTask

Type:                             Classification
Number of trees:                  500
Sample size:                      609
Number of independent variables:  95
Mtry:                             9
Target node size:                 1
Variable importance mode:         none
OOB prediction error:             35.80 %
```

Compute predictions for the same data used in training.

```
pred <- predict(rf, task_Soci_bin)
```

Create a **confusion matrix** for the in-sample predictions.

```
conf.matrix <- calculateConfusionMatrix(pred, sums = TRUE)
conf.matrix
```

```
       high low -err.- -n-
high    336   0      0 336
low       0 273      0 273
-err.-    0   0      0  NA
-n-     336 273     NA 609
```

All observations have been classified without error. Obviously, this **IS NOT** a realistic estimate for the performance on new data!

-> In-sample performance is useless in machine learning!

-> To evaluate our model, we have to use resampling!

Hands-on: Benchmark Analysis

Preparations in mlr

Create *learners* for regression and classification (featureless learner, linear model, LASSO, and random forest).

```
lrns_regr <- list(
  makeLearner("regr.featureless"),
  makeLearner("regr.lm"),
  makeLearner("regr.cvglmnet"), # LASSO
  makeLearner("regr.ranger")
)

lrns_classif <- list(
  makeLearner("classif.featureless"),
  makeLearner("classif.logreg"),
  makeLearner("classif.cvglmnet"), # LASSO
  makeLearner("classif.ranger")
)
```

Comments on the LASSO:

- *Regularized linear regression model (Tibshirani 1996)*
- *Deals with a large number of predictor variables by shrinking the coefficients of non important predictors to zero*
- *Results can be interpreted quite similar to ordinary regression*
- *Often performs comparably to the random forest on survey data (e.g. Pargent and Albert-von der Gönna 2018)*
- *For the basics, checkout James et al. (2013)*

Create *resampling* descriptions for regression and classification.

```
rdesc_regr <- makeResampleDesc("CV", iters = 10)
rdesc_classif <- makeResampleDesc("CV", iters = 10, stratify = TRUE)
# stratify = TRUE; retains the same class proportions in all folds
```

Choose performance measures for regression and classification.

```
mes_regr <- list(rsq, mse, spearmanrho)
mes_classif <- list(mmce, tpr, tnr)
# tpr: sensitivity, tnr: specificity
```

Load the **parallelMap** package to parallelize the computation.

```
library(parallelMap)
```

Set the **L'Ecuyer** seed for reproducible parallelization.

```
set.seed(2, kind = "L'Ecuyer")
parallelStartSocket(parallel::detectCores(), level = "mlr.resample")
# DISCLAIMERS:
# parallelization often not fully reproducible (for different reasons)!
# with enough time, better use RepCV to make the results more stable!
# on Linux and Mac, better use parallelStartMulticore()
```

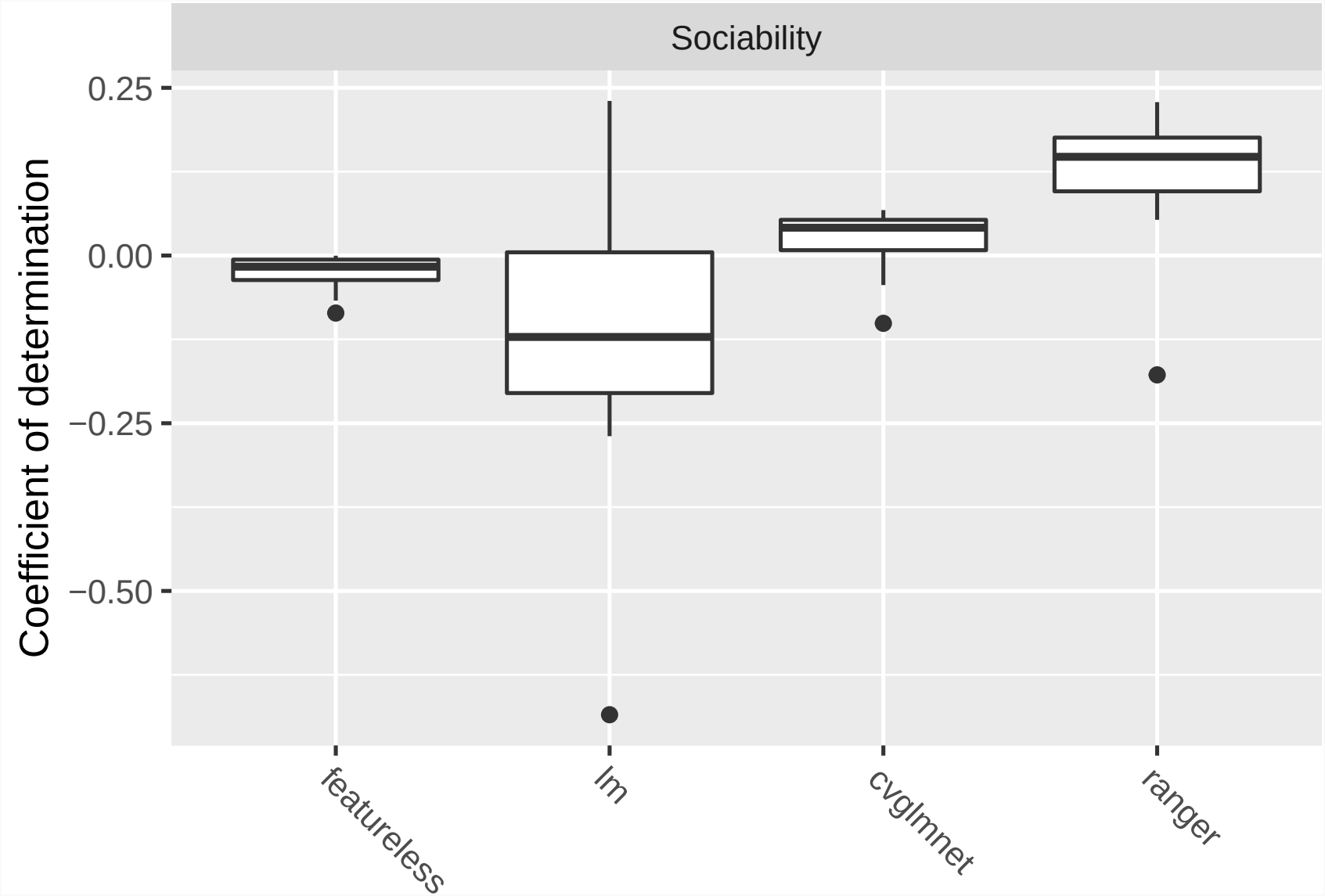
Run the *benchmarks* for regression and classifications.

```
bm_regr <- benchmark(learners = lrns_regr,
                     tasks = task_Soci,
                     resamplings = rdesc_regr,
                     measures = mes_regr)
bm_classif <- benchmark(learners = lrns_classif,
                        tasks = task_Soci_bin,
                        resamplings = rdesc_classif,
                        measures = mes_classif)
```

Terminate parallelization.

```
parallelStop()
```

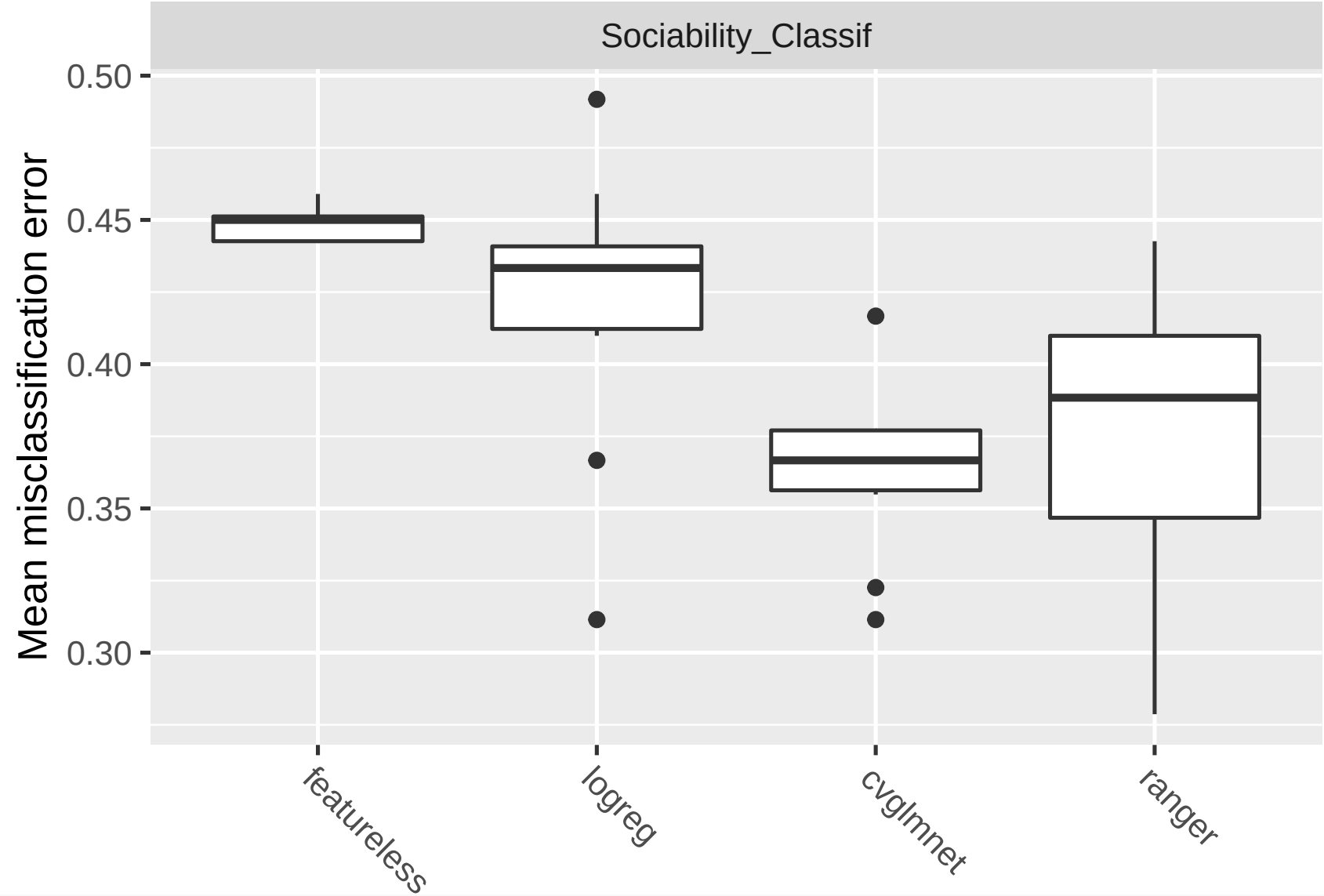
```
plotBMRBoxplots(bm_regr)
```



```
getBMRAggrPerformances(bm_regr, as.df = TRUE)[, -3]
```

task.id	learner.id	mse.test.mean	spearmanrho.test.mean
Sociability	regr.featureless	36	NA
Sociability	regr.lm	40	0.23
Sociability	regr.cvglmnet	35	0.32
Sociability	regr.ranger	31	0.38

```
plotBMRBoxplots(bm_classif)
```



```
getBMRAggrPerformances(bm_classif, as.df = TRUE)[, -3]
```

task.id	learner.id	tpr.test.mean	tnr.test.mean
Sociability_Classif	classif.featureless	1.00	0.00
Sociability_Classif	classif.logreg	0.63	0.51
Sociability_Classif	classif.cvglmnet	0.84	0.38
Sociability_Classif	classif.ranger	0.72	0.50

SECOND BENCHMARK EXAMPLE: Titanic

Load data and remove missing values.

```
library(rpart.plot)
data(ptitanic)
ptitanic <- ptitanic[complete.cases(ptitanic),]
```

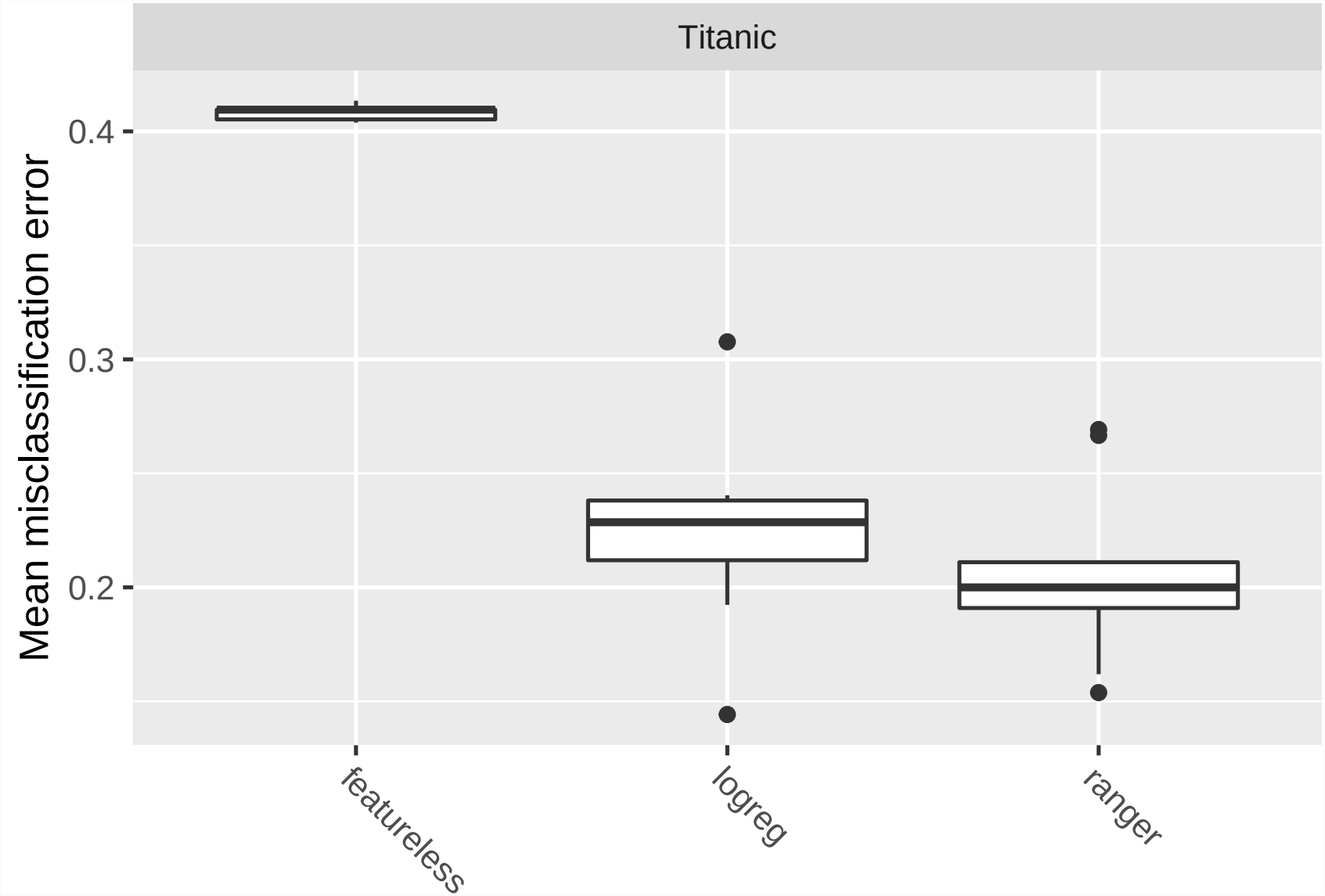
Create *task* object.

```
task_titanic <- makeClassifTask(id = "Titanic",
                                data = ptitanic,
                                target = "survived",
                                positive = "survived")
```

Compute *benchmark* with *CV*.

```
set.seed(1, kind = "L'Ecuyer")
parallelStartSocket(parallel::detectCores(), level = "mlr.resample")
bm <- benchmark(learners = list("classif.featureless",
                                "classif.logreg",
                                "classif.ranger"),
                tasks = task_titanic,
                resamplings = makeResampleDesc("CV", iters = 10,
                                               stratify = TRUE),
                measures = list(mmce, tpr, tnr))
parallelStop()
```

```
plotBMRBoxplots(bm)
```



```
getBMRAggrPerformances(bm, as.df = TRUE)[, -3]
```

task.id	learner.id	tpr.test.mean	tnr.test.mean
Titanic	classif.featureless	0.00	1.00
Titanic	classif.logreg	0.70	0.83
Titanic	classif.ranger	0.65	0.90

Interpretation of Machine-Learning Models

Goal: Describe model predictions (of any machine learning algorithm) in a humanly understandable way

- *Which* predictors most influence model predictions?
  -> **Variable Importance Measures:**
  - ranking of important predictors (no absolute interpretation)
  - mostly inspired by measures from random forests
- *How* do individual predictors influence model predictions?
  -> **Partial Dependence/ICE Plots**:
  - plot possible values of a variable against the (mean) predictions of the model
  - visualize interactions of predictor variables
- **iml** package (Molnar, Casalicchio, and Bischl 2018)

Interpretable Machine Learning with the iml Package

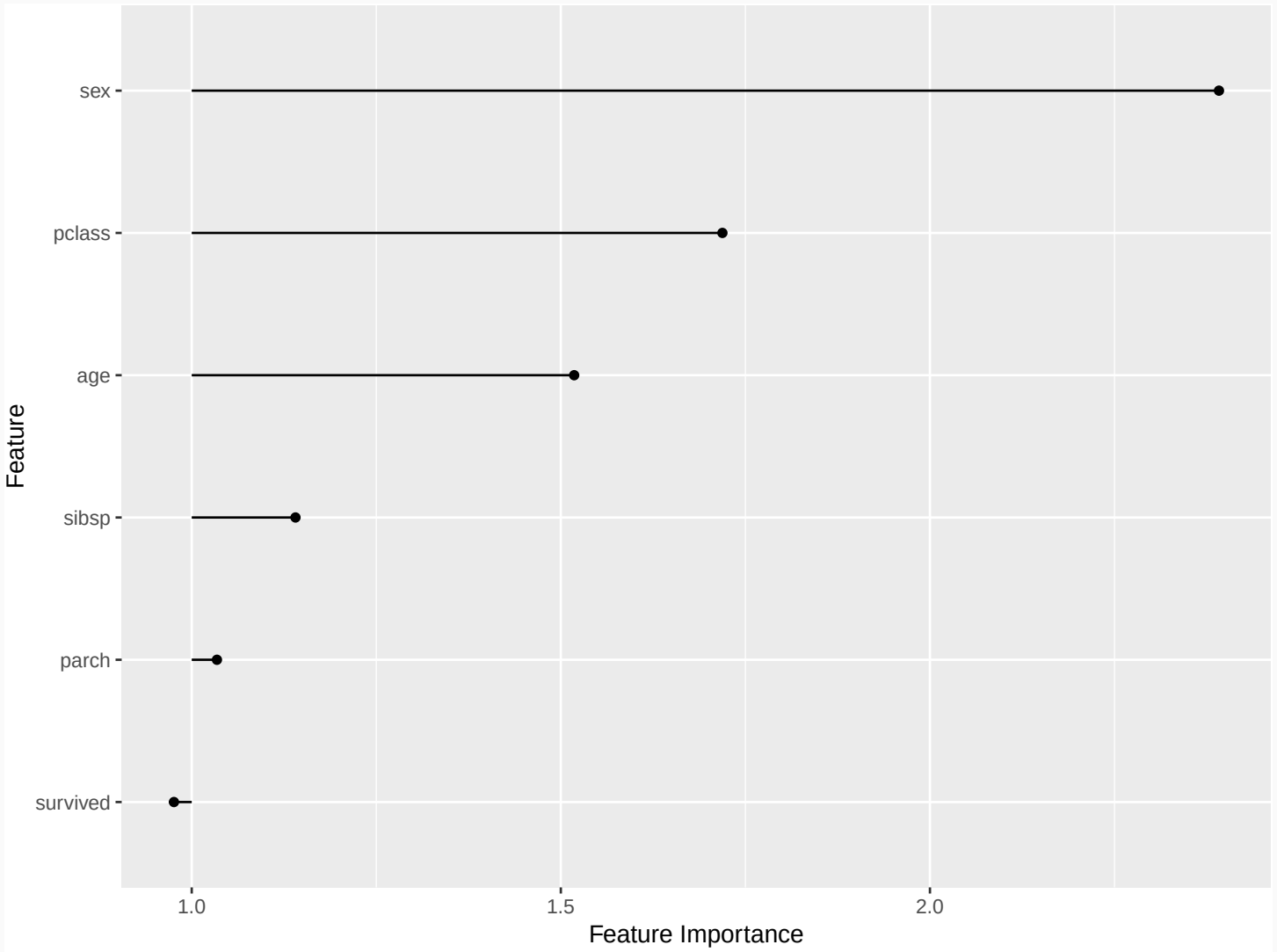
```
rf_titanic <- train(makeLearner("classif.ranger",
                                importance = "permutation",
                                predict.type = "prob"),
                    task = task_titanic)

library(iml)
# uses R6 classes: some unusual syntax

predictor <-
  Predictor$new(rf_titanic, data = getTaskData(task_titanic),
                y = getTaskData(task_titanic)$survived)
```

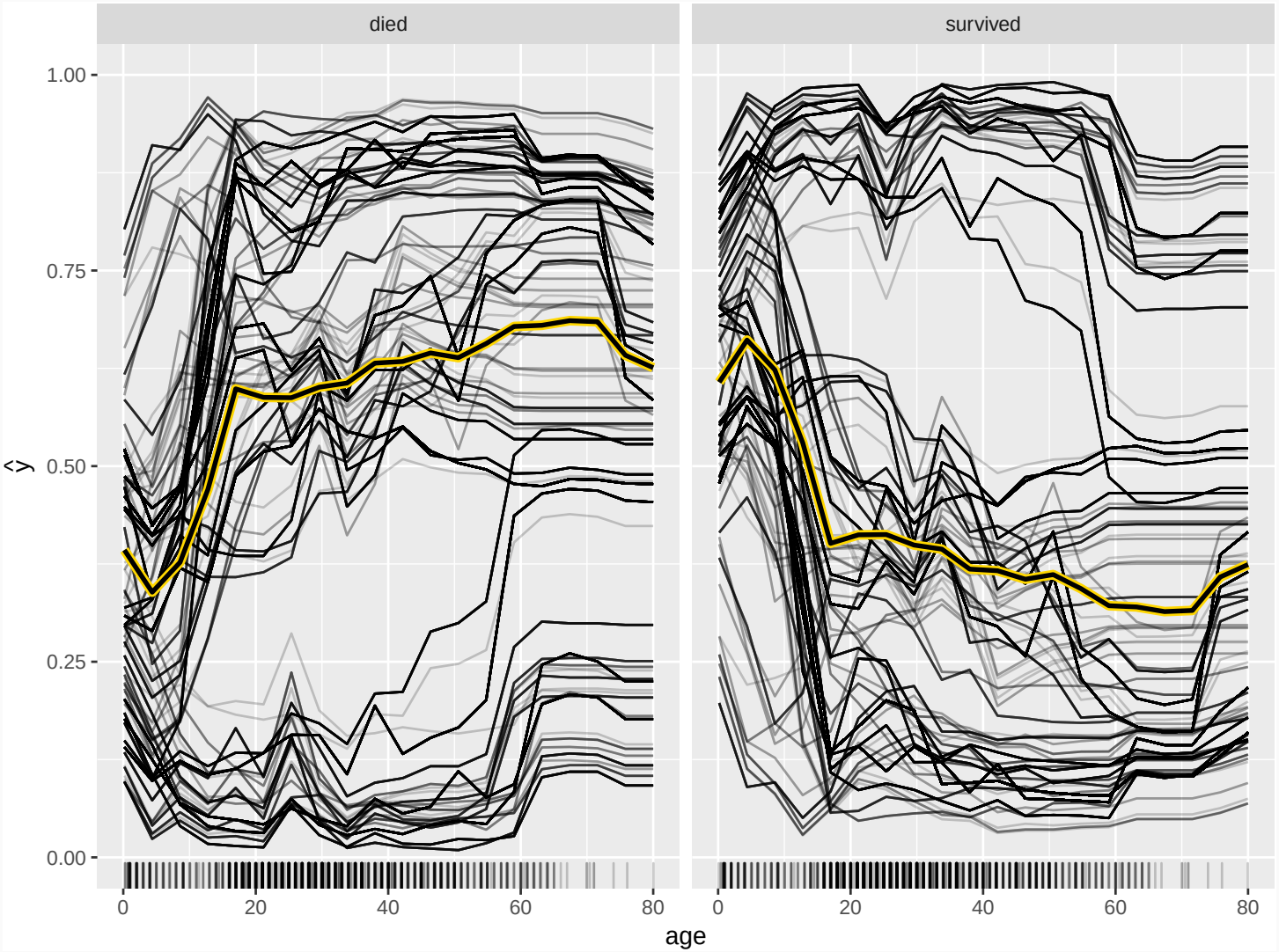
Permutation Variable Importance

```
plot(FeatureImp$new(predictor, loss = "ce"))
```



Partial Dependence/ICE Plot

```
plot(Partial$new(predictor, feature = "age"))
```



Random Forest Specific Variable Importance Measures

- Random Forests provide specific variable importance measures
- Some overestimate the importance of variables with many unique values (Strobl et al. 2007) or high correlations with truly important predictors (Strobl et al. 2008)

Important topics when using variable importance in practice:

- Different measures lead to different results
- Different measures have different interpretations
- Some measures use alternative random forest algorithms
- Some measures are expensive to compute

-> Think hard about what is most appropriate for your application!

Illustrate Differences Between Measures (PhoneStudy Data)

Spearman correlations between different variable importance measures for the **Sociability** prediction task (regression setting).

	cforest_uncond	cforest_cond	ranger_perm	ranger_impur
cforest_uncond	1.00	0.109	0.607	0.49
cforest_cond	0.11	1.000	0.019	0.20
ranger_perm	0.61	0.019	1.000	0.58
ranger_impur	0.49	0.203	0.581	1.00

For more details about variable importance and tree methods visit:

Tuesday, 14:00 – 18.09.2018, Room HZ 3
Trees, Model-Based Trees and Random Forests – An Introduction to Machine Learning Methods for Psychological Research
Chair: Prof. Dr. Carolin Strobl

Take Home Message

- All models (also linear models) should be evaluated adequately:
  - strictly separate training and test sets
  - use cross-validation instead/in addition to in-sample fit
- Benchmark different models when prediction is the main goal
- Use interpretable machine learning to reflect on your models

Thrilled to see more psychological studies using machine learning :-)

Additional Materials for Self Study

- Books:
  - An Introduction to Statistical Learning; James et al. (2013); http://www-bcf.usc.edu/~gareth/ISL/ISLR%20Seventh%20Printing.pdf
  - **ADVANCED** - The Elements of Statistical Learning; Hastie, Tibshirani, and Friedman (2009); https://web.stanford.edu/~hastie/Papers/ESLII.pdf
  - Applied Predictive Modeling; Kuhn and Johnson (2013)
- MOOCs:
  - https://online.stanford.edu/courses/sohs-ystatslearning-statistical-learning-self-paced
  - https://www.coursera.org/learn/machine-learning
  - https://www.coursera.org/learn/practical-machine-learning

Thank you!!!

Florian Pargent

florian.pargent@psy.lmu.de

@FPargent

Clemens Stachl

clemens.stachl@psy.lmu.de

@ClemensStachl

Lehrstuhl Psychologische Methodenlehre und Diagnostik

Ludwig-Maximilians-Universität München

http://www.psy.lmu.de/pm/index.html

Quellen I

Arendasy, M., M. Sommer, and M. Feldhammer. 2011. "Manual Big-Five Structure Inventory BFSI." *Schuhfried Gmbh, Mödling*.

Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. "Mlr: Machine Learning in R." *Journal of Machine Learning Research* 17 (170): 1–5.

Bischl, Bernd, Olaf Mersmann, Heike Trautmann, and Claus Weihs. 2012. "Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation." *Evolutionary*

Quellen II

Computation 20 (2): 249–75.

Breiman, Leo. 2001. “Random Forests.” *Machine Learning* 45 (1): 5–32.

Breiman, Leo, and others. 2001. “Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author).”

Quellen III

Statistical Science 16 (3): 199–231.

Breiman, Leo, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. 1984. *Classification and Regression Trees*. CRC press.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning 2nd Edition*. New York: Springer.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Springer.

Kuhn, Max, and Kjell Johnson. 2013. *Applied Predictive Modeling*. Vol. 26. Springer.

Kuhn, Max, and Hadley Wickham. 2018. *Tidymodels: Easily Install and Load the 'Tidymodels' Packages*.

Quellen IV

https://CRAN.R-project.org/package=tidymodels.

Molnar, Christoph, Guiseppe Casalicchio, and Bernd Bischl. 2018. "Iml: An R Package for Interpretable Machine Learning." *Journal of Open Source Software* 3 (26): 786. doi:10.21105/joss.00786.

Pargent, Florian, and Johannes Albert-von der Gönna. 2018. "Predictive Modeling with Psychological Panel Data." http://dx.doi.org/10.23668/psycharchives.838.

Schoedel, Ramona, Quay Au, Sarah Theres Völkel, Florian Lehmann, Daniela Becker, Markus Bühner, Bernd Bischl, Heinrich Hussmann, and Clemens Stachl. 2018. "Digital Footprints of Sensation Seeking: A Traditional Concept in the Big Data Era."

Quellen V

http://dx.doi.org/10.23668/psycharchives.846.

Shmueli, Galit. 2010. "To Explain or to Predict?" *Statistical Science* 25 (3): 289–310. doi:10.1214/10-STS330.

Stachl, Clemens, and Markus Bühner. 2015. "Show Me How You Drive and I'll Tell You Who You Are Recognizing Gender Using Automotive Driving Parameters." *Procedia Manufacturing*, 6th international conference on applied human factors and ergonomics (ahfe 2015) and the affiliated conferences, ahfe 2015, 3: 5587–94.

Stachl, Clemens, Sven Hilbert, Jiew-Quay Au, Daniel Buschek, Alexander De Luca, Bernd Bischl, Heinrich Hussmann, and Markus Bühner. 2017. "Personality Traits Predict Smartphone Usage." *European Journal of Personality* 31 (6): 701–22.

Quellen VI

doi:10.1002/per.2113.

Strobl, Carolin, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. 2008. "Conditional Variable Importance for Random Forests." *BMC Bioinformatics* 9 (1): 307. doi:10.1186/1471-2105-9-307.

Strobl, Carolin, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. 2007. "Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution." *BMC Bioinformatics* 8 (1): 25. doi:10.1186/1471-2105-8-25.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* 58 (1). [Royal Statistical Society, Wiley]: 267–88.

Quellen VII

http://www.jstor.org/stable/2346178.

Wright, Marvin N, and Andreas Ziegler. 2017. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software* 77 (1): 1–17. doi:10.18637/jss.v077.i01.

Yarkoni, Tal, and Jacob Westfall. 2017. "Choosing Prediction over Explanation in Psychology: Lessons from Machine Learning." *Perspectives on Psychological Science* 12 (6): 1100–1122. doi:10.1177/1745691617693393.

Youyou, Wu, Michal Kosinski, and David Stillwell. 2015. "Computer-Based Personality Judgments Are More Accurate Than Those Made by Humans." *Proceedings of the National Academy of*

Quellen VIII

Sciences 112 (4): 1036–40.