Machine learning methods for soil mapping: Random Forest Modelling

Bas Kempen

Machine learning

- No clear definition among experts.
- Machine learning is a “field of computer science that gives computers the ability to learn without being explicitly programmed” (Samuel, 1959).
- Ability to “learn” (supervised, unsupervised); self-driving cars, email filtering, web search, image recognition (incl. faces).
- Uncovering patterns and structures in (large) data sets; deriving predictive relationships, which makes machine learning algorithms interesting for digital soil mapping.

Algorithms: Random Forest

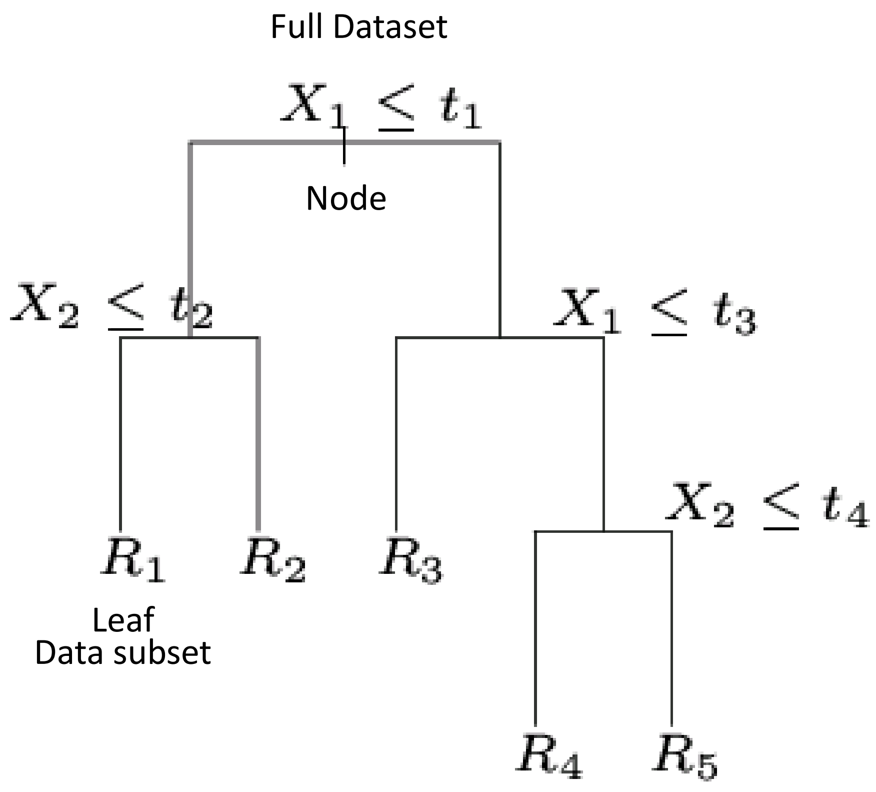
- Many algorithms: **random forest**, neural nets, gradient boosting and many more.
- Machine learning algorithms have become very popular for digital soil mapping since the last 5 years or so, especially random forest modelling.
- Random Forest developed by Tin Kam Ho (1995) and Leo Breiman (2001).
- Based on decision tree models.
- Let’s take a look at these first.

Classification and regression trees (CART)

- Classification tree for categorical data
- Regression tree for continuous data
- Overcome limitations of classical (linear) model:
  - non-linear relationships;
  - n of covariates > n observations;
  - interactions of categorical covariates that result in sparse cell counts;
  - non-parametric.
- Tree modelling: recursive partitioning of the data based on binary splitting using covariates

Growing a tree model

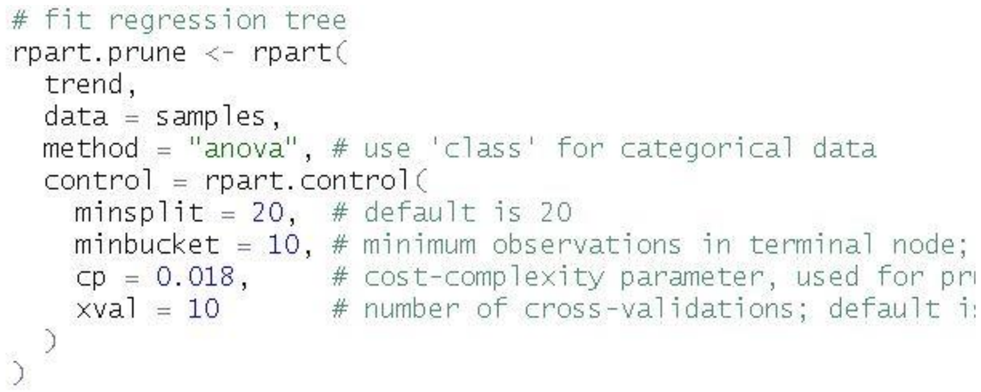
- Evaluate all covariates for each split.
- Split is chosen so that a maximum reduction of the error is achieved. Each node is more 'pure' than its parent node.
- Splitting process is repeated for next two nodes, etc.
- Greedy algorithm.
- The prediction at a leaf is the **mean** value of the data points (RT) or the **modal** class (CT).



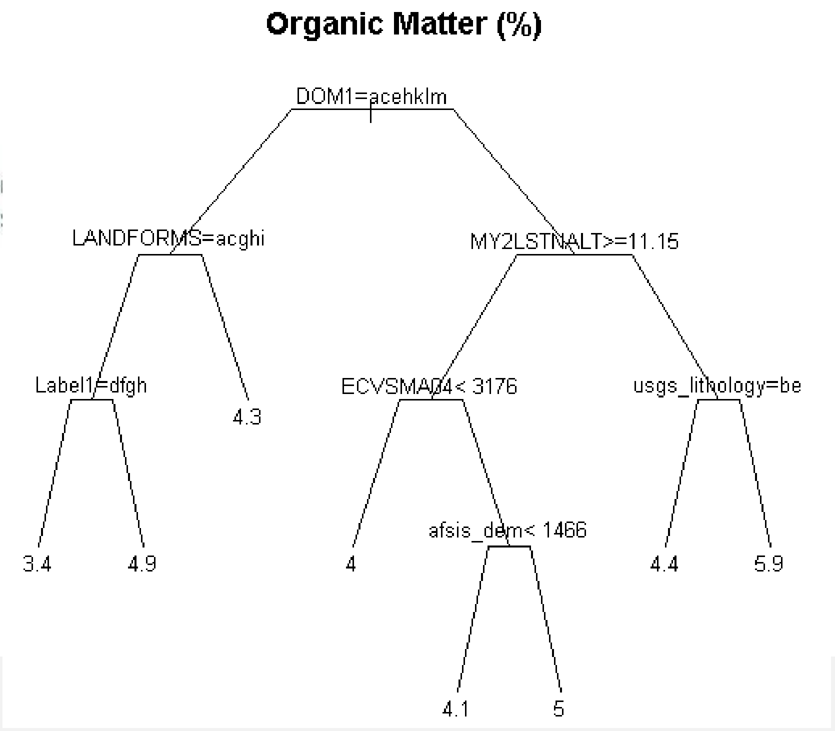
Source: Hastie et al., 2009. The Elements of Statistical Learning

Growing a tree in R

- rpart, tree, party packages
- Fitting a tree model to soil organic matter content



```
# fit regression tree
rpart.prune <- rpart(
  trend,
  data = samples,
  method = "anova", # use 'class' for categorical data
  control = rpart.control(
    minsplit = 20,  # default is 20
    minbucket = 10, # minimum observations in terminal node;
    cp = 0.018,     # cost-complexity parameter, used for pr
    xval = 10       # number of cross-validations; default i:
  )
)
```



Random Forests

- Limitations of CART:
  - Trees are known to be instable: sensitive to small changes in learning data -> tree structure can be completely altered.
  - Predictions of single trees show high variability: trees built on different bootstrap samples can have different structures.
  - Danger of over-fitting.
- Can be avoided using **ensemble methods**: base prediction on a whole set of trees rather than a single tree
- Ensemble methods use the fact that individual trees are unstable but on average produce the right result.
- Random forests is such ensemble method: a forest of trees is grown; the prediction is an aggregation of the individual tree predictions.

How does it work?

- Random forests combine:
  - bootstrap aggregation (‘bagging’)
  - random selection of predictors
- Algorithm:
  - Draw a bootstrap sample:
    - Random selection of 2/3 of the training data; repeat *n* times.
    - Grow an unpruned tree to each bootstrap sample
  - Random predictor selection: for each split in each tree a random subset is selected from the predictor variables. The best split is chosen from among the selected predictors.
  - Predict new data by aggregating the predictions of the n trees.
    - Average for continuous variables
    - Majority vote for categorical variables.

Fitting a random forest model

- Training data (soil property of interest)
- Explanatory variables (environmental covariates)
- Model tuning parameters:
  - **mtry**: number of randomly selected splitting variables.
  - **ntree**: number of trees grown
- Choice of mtry and ntree parameters affect the stability of the tree:
  - Should be large enough so that each variable has a chance to occur in enough trees
  - Adjust if prediction results and variable importance differ for different random seeds
  - Optimize tuning parameters (with caret package)

Out-Of-Bag accuracy assessment

- Random Forest comes with an internal accuracy assessment (based on cross-validation; no need to do a separate assessment).

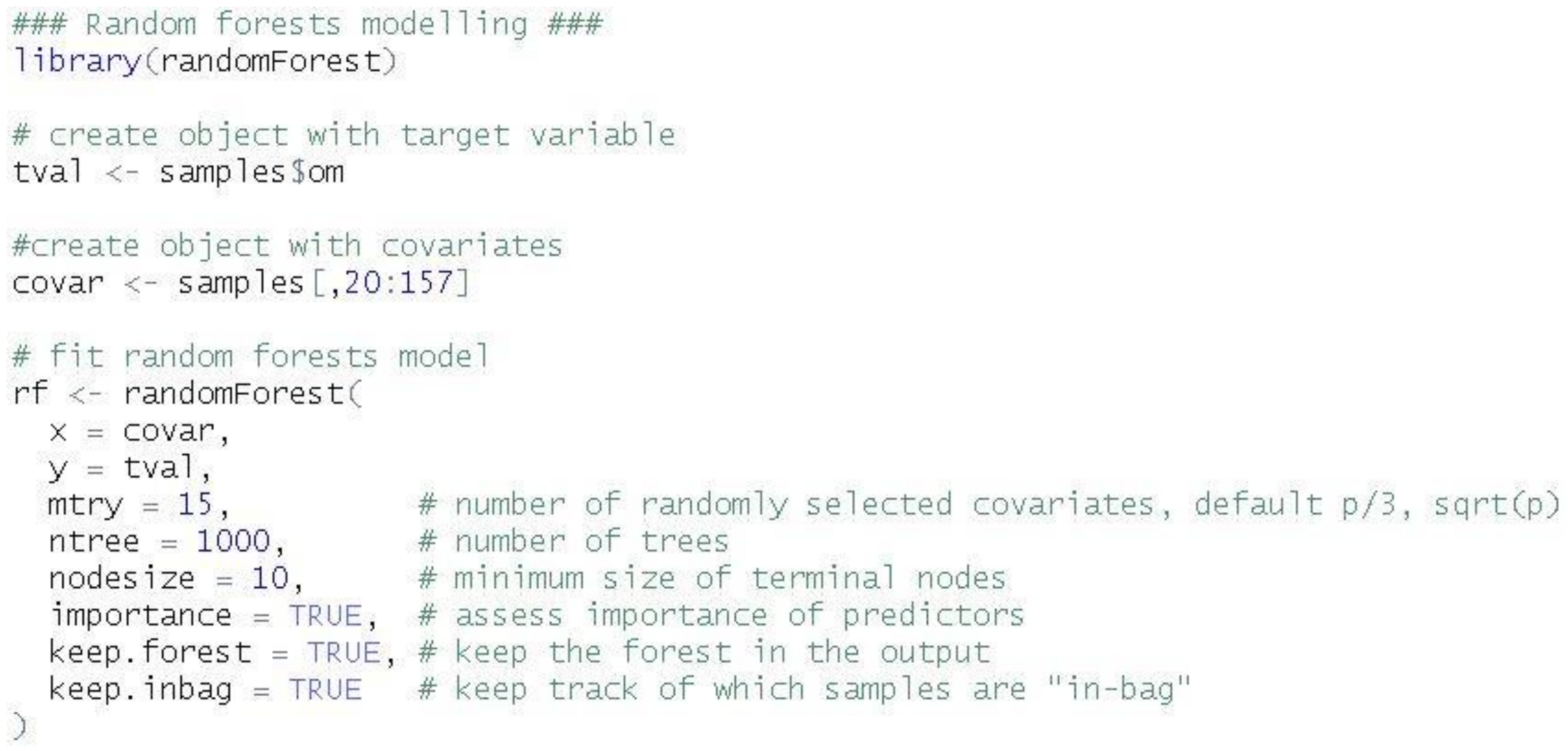
- Bootstrap sampling: the algorithm sets aside ~36% of the training data for each tree grown: out-of-bag data (OOB).

- Each data point will be OOB for approximately ~36% of the number of fitted trees.

- OOB data can be used to asses prediction accuracy:
  - Predict the data not in the bootstrap sample for each tree
  - Aggregate the OOB predictions: **mean** (continuous data), **majority vote** (categorical data).

Application in R

- Packages: **randomForest**, ranger, cforest, party



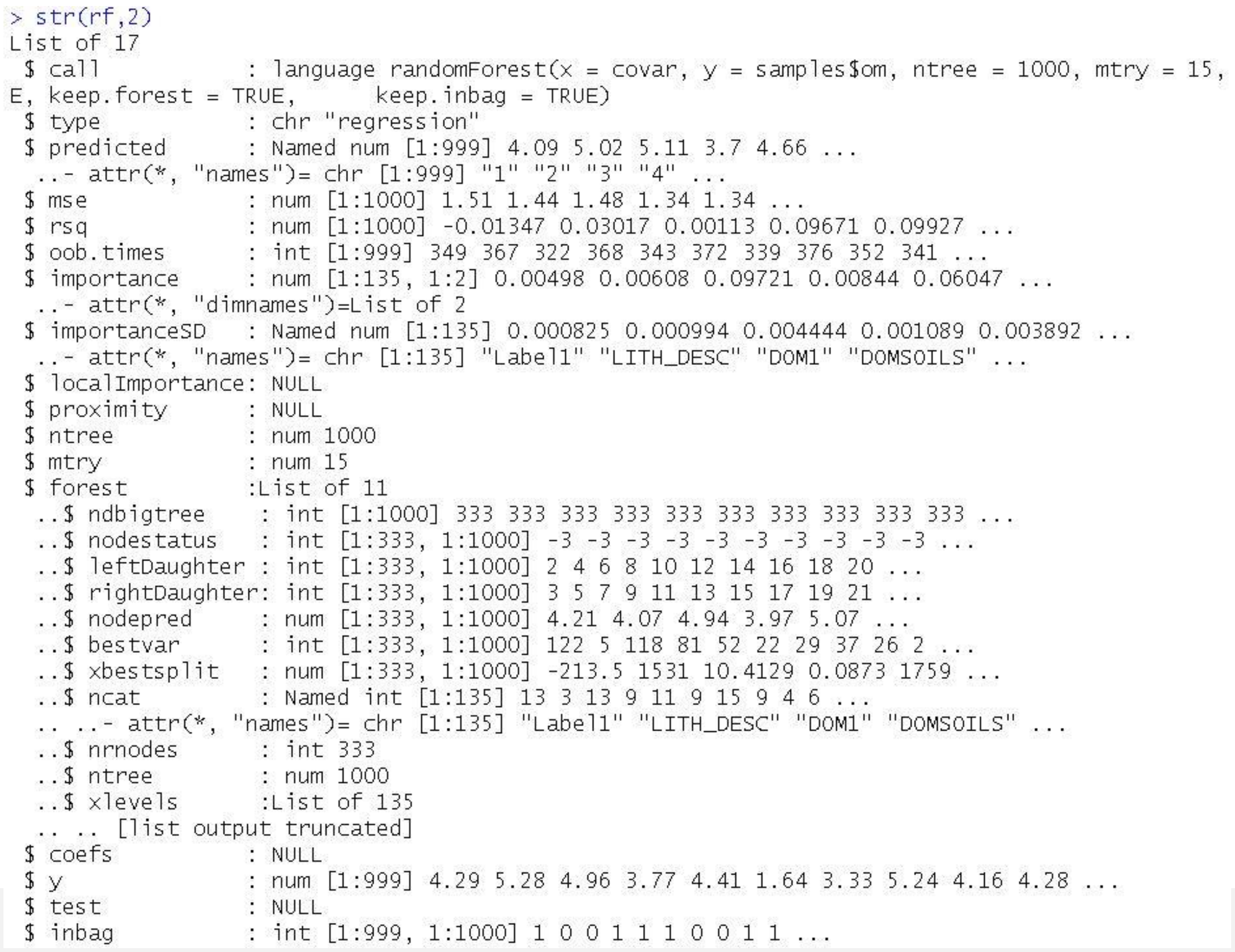
```r
### Random forests modelling ###
library(randomForest)

# create object with target variable
tval <- samples$om

#create object with covariates
covar <- samples[,20:157]

# fit random forests model
rf <- randomForest(
  x = covar,
  y = tval,
  mtry = 15,            # number of randomly selected covariates, default p/3, sqrt(p)
  ntree = 1000,         # number of trees
  nodesize = 10,        # minimum size of terminal nodes
  importance = TRUE,    # assess importance of predictors
  keep.forest = TRUE,   # keep the forest in the output
  keep.inbag = TRUE     # keep track of which samples are "in-bag"
)
```

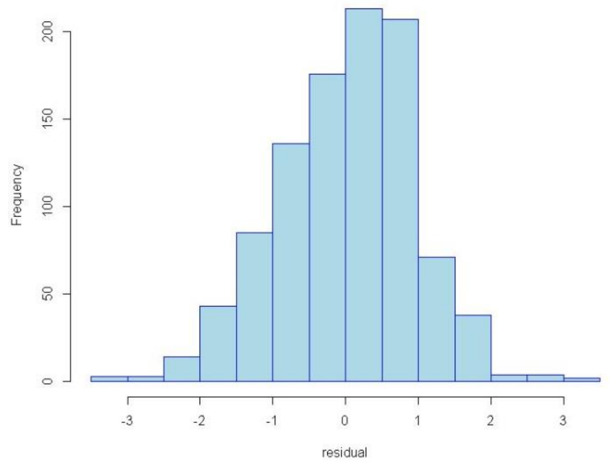
The random forest model in R

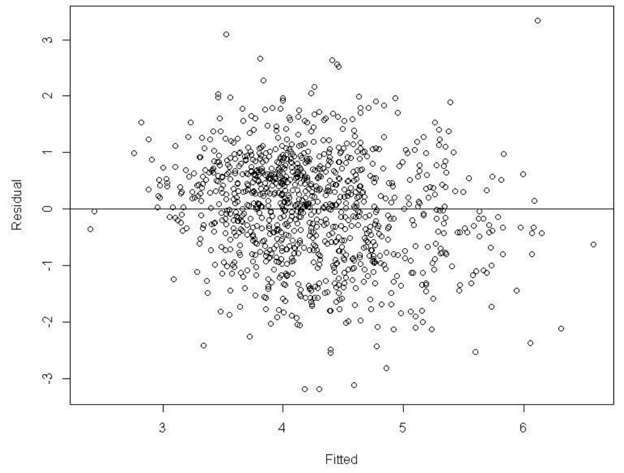


```
> str(rf,2)
List of 17
 $ call           : language randomForest(x = covar, y = samples$om, ntree = 1000, mtry = 15,
E, keep.forest = TRUE,      keep.inbag = TRUE)
 $ type           : chr "regression"
 $ predicted      : Named num [1:999] 4.09 5.02 5.11 3.7 4.66 ...
  ..- attr(*, "names")= chr [1:999] "1" "2" "3" "4" ...
 $ mse            : num [1:1000] 1.51 1.44 1.48 1.34 1.34 ...
 $ rsq            : num [1:1000] -0.01347 0.03017 0.00113 0.09671 0.09927 ...
 $ oob.times      : int [1:999] 349 367 322 368 343 372 339 376 352 341 ...
 $ importance     : num [1:135, 1:2] 0.00498 0.00608 0.09721 0.00844 0.06047 ...
  ..- attr(*, "dimnames")=List of 2
 $ importanceSD   : Named num [1:135] 0.000825 0.000994 0.004444 0.001089 0.003892 ...
  ..- attr(*, "names")= chr [1:135] "Label1" "LITH_DESC" "DOM1" "DOMSOILS" ...
 $ localImportance: NULL
 $ proximity      : NULL
 $ ntree          : num 1000
 $ mtry           : num 15
 $ forest         :List of 11
  ..$ ndbigtree    : int [1:1000] 333 333 333 333 333 333 333 333 333 333 ...
  ..$ nodestatus   : int [1:333, 1:1000] -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 ...
  ..$ leftDaughter : int [1:333, 1:1000] 2 4 6 8 10 12 14 16 18 20 ...
  ..$ rightDaughter: int [1:333, 1:1000] 3 5 7 9 11 13 15 17 19 21 ...
  ..$ nodepred     : num [1:333, 1:1000] 4.21 4.07 4.94 3.97 5.07 ...
  ..$ bestvar      : int [1:333, 1:1000] 122 5 118 81 52 22 29 37 26 2 ...
  ..$ xbestsplit   : num [1:333, 1:1000] -213.5 1531 10.4129 0.0873 1759 ...
  ..$ ncat         : Named int [1:135] 13 3 13 9 11 9 15 9 4 6 ...
  .. ..- attr(*, "names")= chr [1:135] "Label1" "LITH_DESC" "DOM1" "DOMSOILS" ...
  ..$ nrnodes      : int 333
  ..$ ntree        : num 1000
  ..$ xlevels      :List of 135
  .. .. [list output truncated]
 $ coefs          : NULL
 $ y              : num [1:999] 4.29 5.28 4.96 3.77 4.41 1.64 3.33 5.24 4.16 4.28 ...
 $ test           : NULL
 $ inbag          : int [1:999, 1:1000] 1 0 0 1 1 1 0 0 1 1 ...
```

Random forest residuals

- The randomForest function returns predicted values of the input data based on out-of-bag samples.
- Model residuals can be computed: predicted – observed (rf$predicted-rf$y).
- Check assumptions on residuals (prior to geostatistical analysis): normality, constant variance





- Fit variogram, krige residuals, add kriged residuals to RF predictions

Variable importance

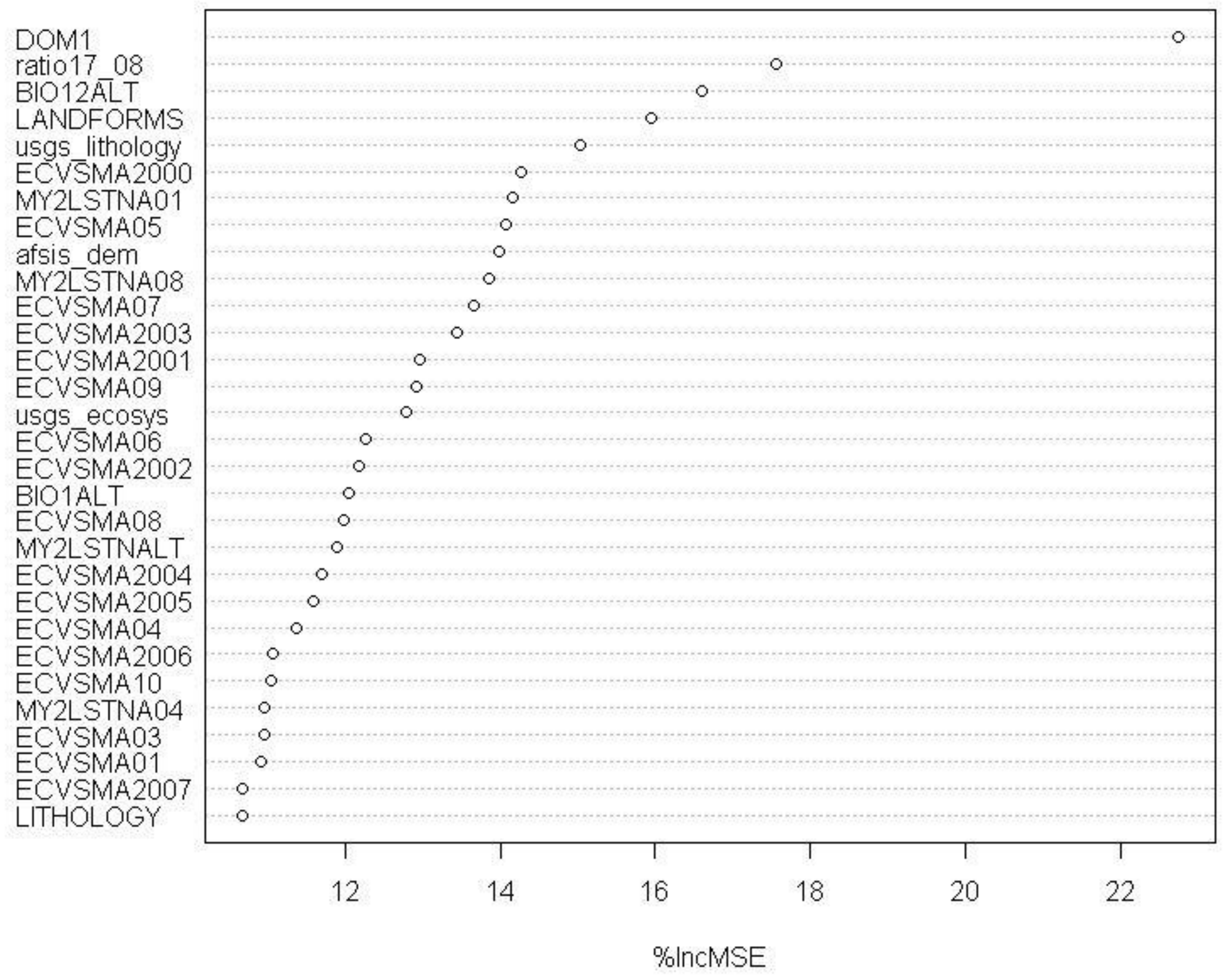
- Ensembles of trees are not easy to interpret: no such thing as an average tree; an individual tree does not tell much.

- Ensemble can reflect the potentially (complex) effect of a variable on the response $\longrightarrow$ assess the relevance of each variable over all trees of the ensemble.

- Variable importance plot: shows how much prediction error increases when the **values** of one predictor **are permuted** (break association with response) while all others are left unchanged.

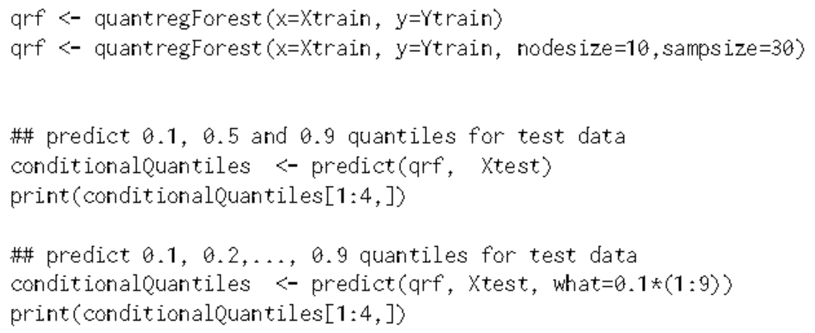
- Permuted variable is used together with other variables to predict the response $\longrightarrow$ prediction accuracy will decrease.

Variable importance plot



Uncertainty assessment

- Uncertainty associated to random forest predictions cannot be easily qualified and is an field of ongoing research and online discussions.
- Methods are computationally intensive (requires much memory)!
- R package **quantregForest**: compute quantiles for predictions.



```
qrf <- quantregForest(x=Xtrain, y=Ytrain)
qrf <- quantregForest(x=Xtrain, y=Ytrain, nodesize=10,sampsize=30)


## predict 0.1, 0.5 and 0.9 quantiles for test data
conditionalQuantiles  <- predict(qrf,  Xtest)
print(conditionalQuantiles[1:4,])

## predict 0.1, 0.2,..., 0.9 quantiles for test data
conditionalQuantiles  <- predict(qrf, Xtest, what=0.1*(1:9))
print(conditionalQuantiles[1:4,])
```

- Pragmatic approach: predict *n* values for each location, compute variance or standard deviation, and from these confidence intervals:

```
pred.rf <- predict(rf, newdata = newdata, predict.all = TRUE)

pred.rfvar <- apply(pred.rf$individual, MARGIN = 1, FUN =var)
```

Optimizing tuning parameters

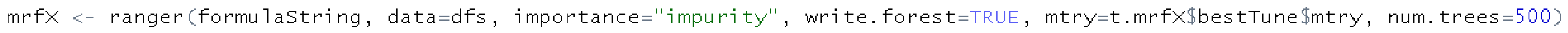
- Instead of using the default one can try to find an optimal value for the **mtry** parameter.
- **train** function of the **caret** package:

```
ctrl <- trainControl(method="repeatedcv", number=3, repeats=1, savePredictions = "final")
rf.tuneGrid <- expand.grid(mtry = seq(4,20,by=2))
t.mrfX <- train(formulaString.lst[[j]], data=dfs, method="rf", trControl=ctrl, tuneGrid=rf.tuneGrid)
t.mrfX$bestTune$mtry
```

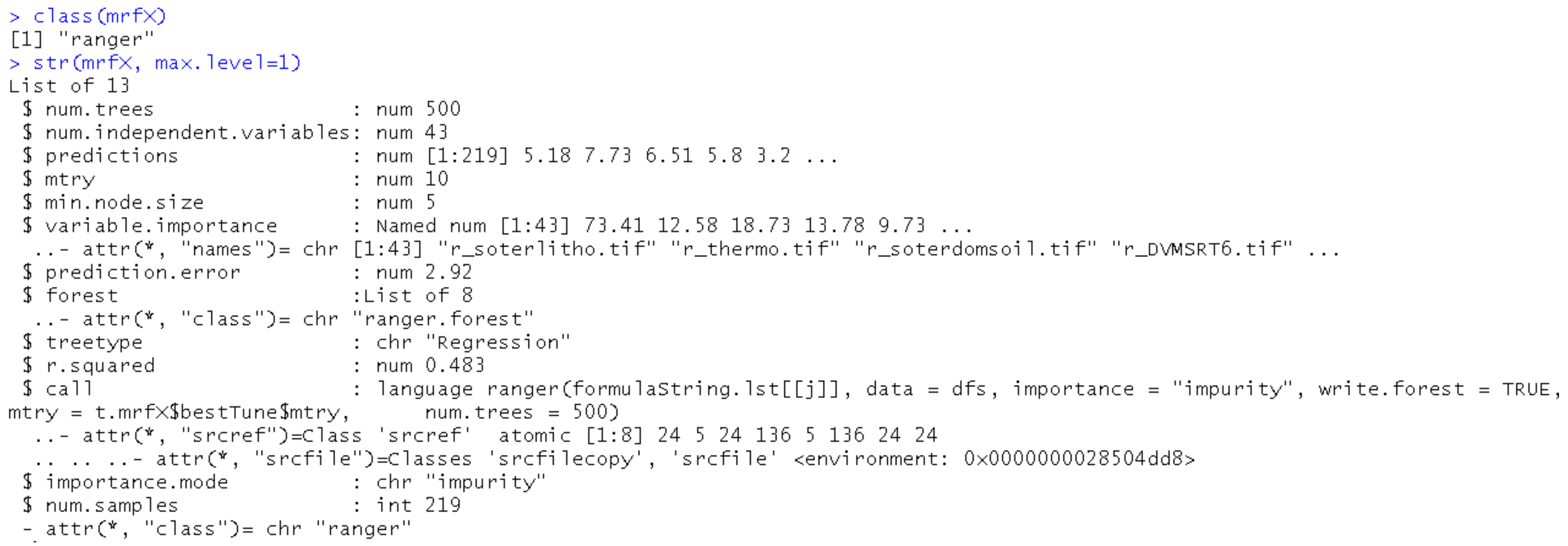
- Offer the optimal **mtry** value to the **randomForest** function
- Try various **ntree** values (e.g. 500, 750, 1000, 1500)

ranger package

- Random forest models can also be fitted with the **ranger** package.
- Faster implementation; allows parallel computing.



```
mrfX <- ranger(formulaString, data=dfs, importance="impurity", write.forest=TRUE, mtry=t.mrfX$bestTune$mtry, num.trees=500)
```



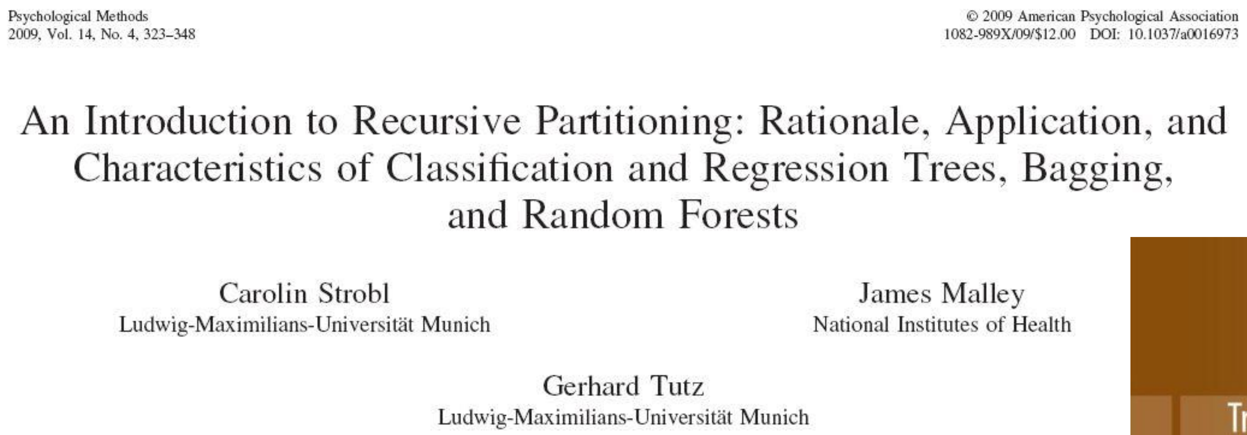
```
> class(mrfX)
[1] "ranger"
> str(mrfX, max.level=1)
List of 13
 $ num.trees                : num 500
 $ num.independent.variables: num 43
 $ predictions              : num [1:219] 5.18 7.73 6.51 5.8 3.2 ...
 $ mtry                     : num 10
 $ min.node.size            : num 5
 $ variable.importance      : Named num [1:43] 73.41 12.58 18.73 13.78 9.73 ...
  ..- attr(*, "names")= chr [1:43] "r_soterlitho.tif" "r_thermo.tif" "r_soterdomsoil.tif" "r_DVMSRT6.tif" ...
 $ prediction.error         : num 2.92
 $ forest                   :List of 8
  ..- attr(*, "class")= chr "ranger.forest"
 $ treetype                 : chr "Regression"
 $ r.squared                : num 0.483
 $ call                     : language ranger(formulaString.lst[[j]], data = dfs, importance = "impurity", write.forest = TRUE,
mtry = t.mrfX$bestTune$mtry,      num.trees = 500)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 24 5 24 136 5 136 24 24
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000028504dd8>
 $ importance.mode          : chr "impurity"
 $ num.samples              : int 219
 - attr(*, "class")= chr "ranger"
```

Be aware

- No clear interpretation.
- Prediction uncertainty not easy to quantify (computationally intensive).
- Spatial correlation cannot be accounted for when fitting the model.
- Bias in variable selection (Strobl et al. 2009).
- Stability of the forest depends on ntree, mtry settings.

Recources

- https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm


Psychological Methods
2009, Vol. 14, No. 4, 323–348

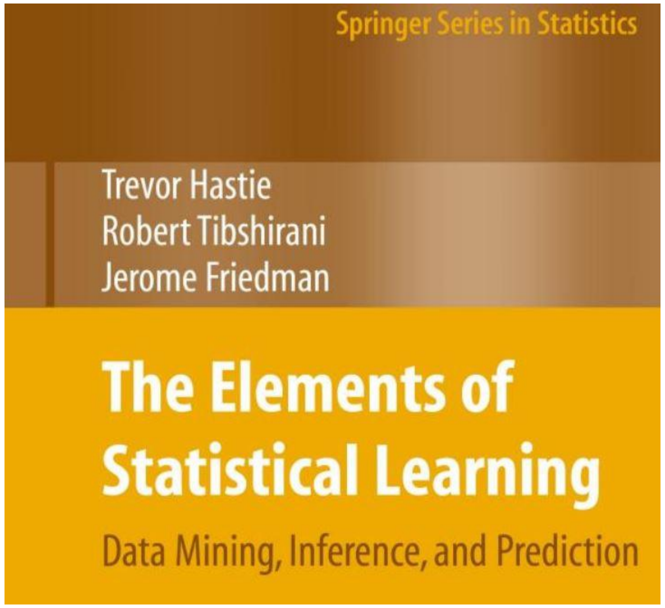
© 2009 American Psychological Association
1082-989X/09/$12.00 DOI: 10.1037/a0016973

An Introduction to Recursive Partitioning: Rationale, Application, and Characteristics of Classification and Regression Trees, Bagging, and Random Forests

Carolin Strobl
Ludwig-Maximilians-Universität Munich

James Malley
National Institutes of Health

Gerhard Tutz
Ludwig-Maximilians-Universität Munich


Springer Series in Statistics

Trevor Hastie
Robert Tibshirani
Jerome Friedman

The Elements of Statistical Learning

Data Mining, Inference, and Prediction

ISRIC
World Soil Information

Thank you for listening

www.isric.org