Supplement

This supplement provides the R code and documentation for the application examples presented in the paper "An Introduction to Recursive Partitioning: Rationale, Application and Characteristics of Classification and Regression Trees, Bagging and Random Forests". It was created by means of the `Sweave` function for mixing R and LaTeX code (Leisch 2002).

Classification and Regression Trees

- Select a working directory, where all created objects and figures will be stored.

  ```
  > setwd("~/myfolder")
  ```

- Read in the data set.

  ```
  > dat_smoking <- read.table("dat_smoking.txt")
  ```

  The variable `intention_to_smoke` is the binary response variable. The other variables are two binary and two numeric predictor variables.

  (If SPSS data frames are supposed to be read, attach the package `foreign` and use the functions `read.spss` and `as.data.frame` to create an appropriate R data frame.)

- Attach the add-on package `party`.

  ```
  > library("party")
  ```

  (If packages have not been installed previously, they can be installed with the `install.packages` command. Use the option `dependencies = TRUE` to ensure all necessary functions from other packages are also available.)

- Fit and plot a classification tree.

  ```
  > myctree <- ctree(intention_to_smoke ~ ., data = dat_smoking)
  ```

  The association between the response variable `intention_to_smoke` and all other variables in the data set, as indicated by the . symbol in the function call, is modeled.

  The default parameter settings in the function `ctree` guarantee that variable selection is unbiased (Hothorn, Hornik, and Zeileis 2006).

  A classification tree is fitted automatically, because the response variable is a factor. (The "c" in `ctree` does not stand for "classification", but refers to the conditional inference tests employed in split selection.) For a numeric response, a regression tree would be fitted.

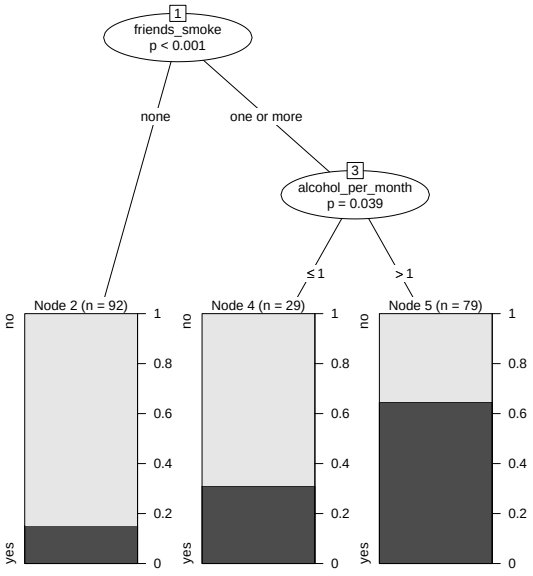
  Make sure your response variable is correctly encoded!

  This can be checked, e.g., by means of:

  ```
  > class(dat_smoking$intention_to_smoke)
  ```

```
[1] "factor"
```

```
> plot(myctree)
```



Model-Based Recursive Partitioning

- Make available the data set from the add-on package `lme4`.

  ```
  > data("sleepstudy", package="lme4")
  ```

- Select some subjects. (Otherwise fitting will take a while, because all combinations of subjects need to be compared for parameter instabilities in their regression models.)

  ```
  > dat_sleep <- subset(sleepstudy, Subject %in% c(308,309,335,350))
  > dat_sleep$Subject <- factor(dat_sleep$Subject)
  ```

  (The latter command only eliminates the remaining factor levels.)

- Fit and plot a model-based tree.

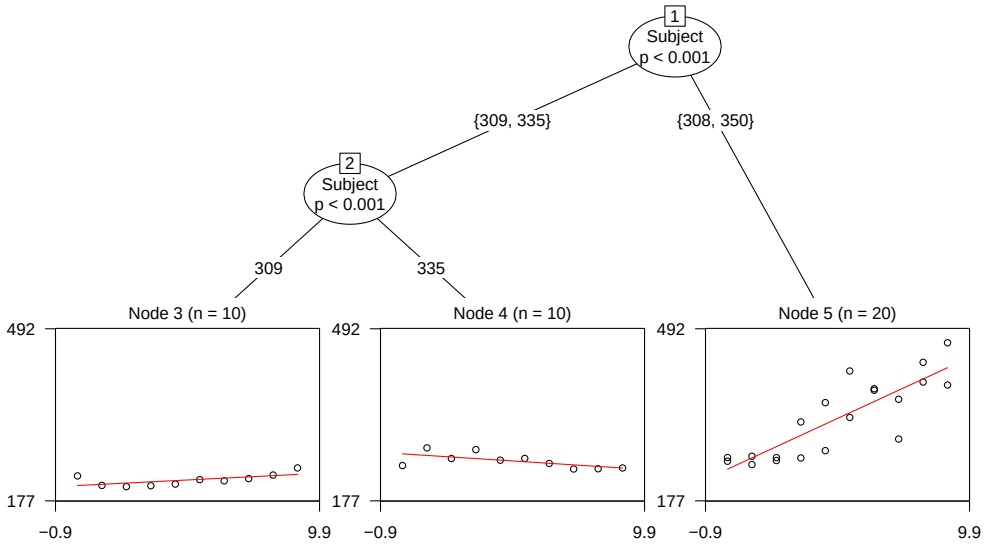
  ```
  > mymob <- mob(Reaction ~ Days | Subject, data = dat_sleep,
  + control = mob_control(minsplit = 10))
  ```

  The minimum number of observations per node necessary for splitting `minsplit` is set to 10 here, because 10 observations are available for each subject and we want to be able to identify even single subjects with deviating model parameters.

  If each observation corresponded to one subject, and subjects were partitioned w.r.t. covariates such as age and gender, the default value of `minsplit` would guarantee, as a stop

criterion, that in each terminal node a sufficient number of observations is available for model fitting.

```
> plot(mymob)
```



Random Forests

- Read in the data set.

  ```
  > dat_genes <- read.table("dat_genes.txt")
  ```

  The variable `status` is the binary response variable. The other variables are clinical and gene predictor variables, of which two were modified to be relevant.

- Set control parameters for random forest construction.

  ```
  > mycontrols <- cforest_unbiased(ntree=1000, mtry=20, minsplit=5)
  ```

  The parameter settings in the default option `cforest_unbiased` guarantee that variable selection and variable importance are unbiased (Strobl, Boulesteix, Zeileis, and Hothorn 2007).

  The `ntree` argument controls the overall number of trees in the forest, and the `mtry` argument controls the number of randomly preselected predictor variables for each split.

  If a data set with more genes was analyzed, the number of trees (and potentially the number of randomly preselected predictor variables) should be increased to guarantee stable results.

  The square-root of the number of variables is often suggested as a default value for `mtry`. Note, however, that in the `cforest` function the default value for `mtry` is fixed to 5 for technical reasons, and needs to be adjusted if desired.

If `mtry` was set to the number of predictor variables in the data set, `ncol(dat)-1` (= number of columns, but not counting the column for the response variable), the procedure would be equal to bagging.

The minimum number of observations per node necessary for splitting, `minsplit`, is set to a low value here, because the sample is rather small and in random forests usually large trees are desired. The other potential stopping criterion for the `cforest` function, the minimum criterion value necessary for splitting, `mincriterion`, is already set to 0 per default.

The control parameters can either be stored in advance and then used in the function call, as displayed here, or specified directly in the function call, as in the previous example.

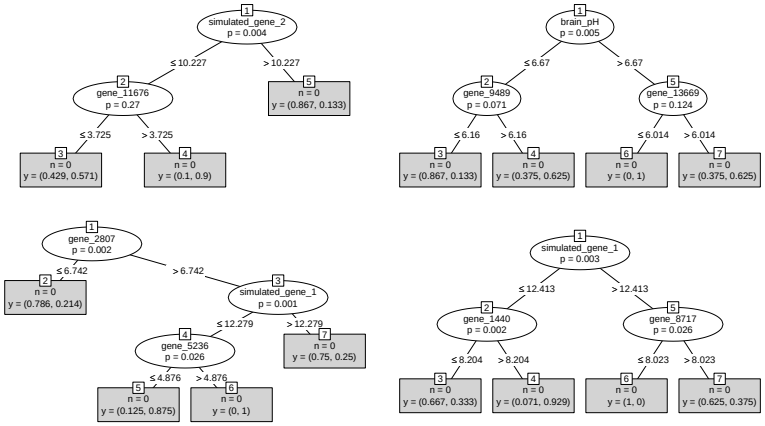
- Set an (arbitrary) random seed and fit a random forest with the control parameters defined above.

  Note that, as a hint to the reader, random seeds are set every time random sampling or random permutations are involved in the following.

```
> set.seed(2908)
> mycforest <- cforest(status ~ ., data=dat_genes, controls=mycontrols)
```

- Look at some trees in the forest (the same method was used to illustrate the variability of single trees in bagging and random forests in the paper).

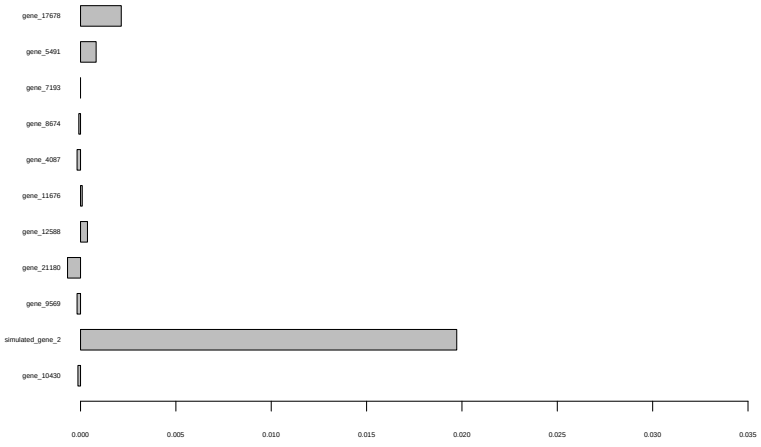
```
> xgr <- 2
> grid.newpage()
> cgrid <- viewport(layout = grid.layout(xgr, xgr), name = "cgrid")
> pushViewport(cgrid)
> for (i in 1:xgr) {
+  for (j in 1:xgr) {
+    pushViewport(viewport(layout.pos.col = i, layout.pos.row = j))
+    tr <- party:::prettytree(mycforest@ensemble[[i + j * xgr]],
+                             names(mycforest@data@get("input")))
+    plot(new("BinaryTree", tree = tr, data = mycforest@data,
+             responses = mycforest@responses),
+         newpage = FALSE, pop = FALSE, type="simple")
+    upViewport()
+  }
+ }
```



- Compute and plot the permutation importance of each predictor variable.

```
> set.seed(2908)
> myvarimp <- varimp(mycforest)

> barplot(myvarimp[90:100], space=0.75, xlim=c(0,0.035),
+   names.arg=rownames(myvarimp)[90:100], horiz=TRUE, cex.names=0.45,
+   cex=0.45, las=1)
```



(Only a few genes are displayed here to save space. All but the first plot options are only for aesthetics.)

- Prediction in terms of the predicted response class or the predicted class probabilities for some selected subjects.

```
> subjects <- 28:32
> y <- dat_genes$status[subjects]
> y_hat <- predict(mycforest, newdata=dat_genes[subjects,])
> p_hat <- sapply(treeresponse(mycforest, newdata=dat_genes[subjects,]),
+   FUN=function(x)x[,1])
> tab <- cbind(y, y_hat, p_hat)
> rownames(tab) <- paste("subject",subjects)
```

The results are displayed here as a LaTeX table by means of the `xtable` function from the package of the same name. (Only one class probability needs to be displayed for a binary classification problem.)

```
> library("xtable")
> colnames(tab)<-c("$y$", "$\\hat{y}$", "$\\hat{p}\\left(y=1\\right)$")
> print(xtable(tab, align="cccc", digits=c(0,0,0,2)),
+ type = "latex", sanitize.text.function = function(x){x})
```

	y	$\hat{y}$	$\hat{p}\,(y=1)$
subject 28	1	1	0.80
subject 29	1	2	0.46
subject 30	1	1	0.64
subject 31	2	2	0.48
subject 32	2	2	0.43

- Compute the percentage of correct predictions and the confusion matrix from the entire learning sample or from the out-of bag (`OOB`) sample only.

```
> y_hat<-predict(mycforest)
> y_hat_oob<-predict(mycforest, OOB=TRUE)
> sum(dat_genes$status==y_hat)/nrow(dat_genes)

[1] 0.9016393

> sum(dat_genes$status==y_hat_oob)/nrow(dat_genes)

[1] 0.6721311

> table(dat_genes$status, y_hat)

                y_hat
                 Bipolar disorder Healthy control
  Bipolar disorder               28               2
  Healthy control                 4              27
```

```
> table(dat_genes$status, y_hat_oob)

                  y_hat_oob
                   Bipolar disorder Healthy control
  Bipolar disorder               20              10
  Healthy control                10              21
```

References

Hothorn, T., K. Hornik, and A. Zeileis (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics 15*(3), 651–674.

Leisch, F. (2002). Sweave: Dynamic generation of statistical reports. In W. Härdle and B. Rönz (Eds.), *Proceedings in Computational Statistics*, Heidelberg, pp. 575–580. Physika Verlag.

Strobl, C., A.-L. Boulesteix, A. Zeileis, and T. Hothorn (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics 8:25*.