Biostatistics: Exercise 11

Beate Sick, Lisa Herzog

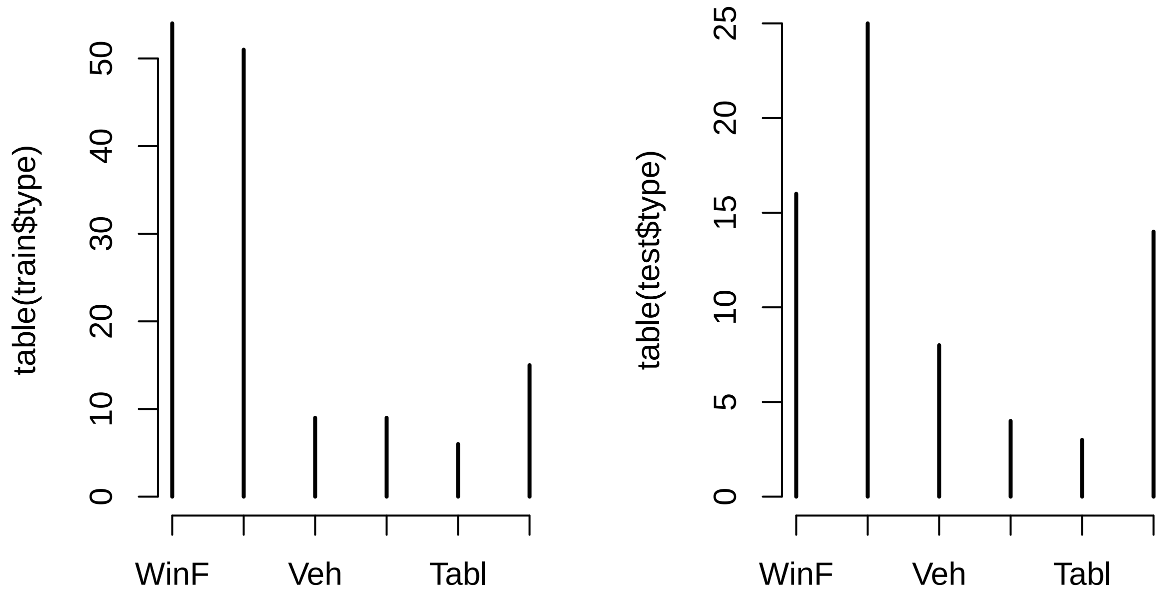
24.11.2020

Exercise 1: Random Forest for classification

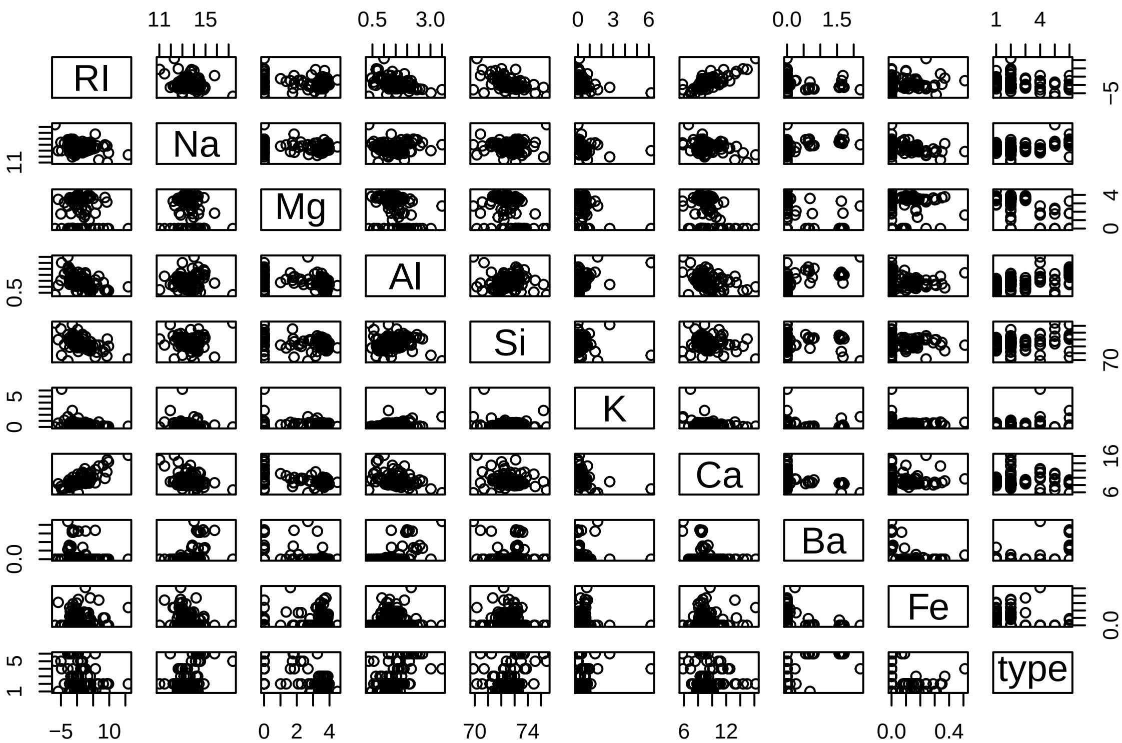
The goal in this exercise is to use a Random Forest for classification. The data set summarizes the chemical concentration of 9 different elements (e.g. Na and Mg) and each observation corresponds to one out of 6 classes corresponding to different types of glass fragments. You can download the training `train.fgl.RData` and the test data set `test.fgl.RData` from the Website.

- Load the training and the test data into R using the function `load()` and become acquainted with the data. Perform a descriptive analysis. Assess how the target variable `type` is distributed in the training and the test data and evaluate the pair-wise relationships between the explanatory variables and the target `type`. Comment on your analysis results.

```
load(paste0(dir, "data/train.fgl.RData"))
load(paste0(dir, "data/test.fgl.RData"))

str(train)
## 'data.frame':    144 obs. of  10 variables:
##  $ RI  : num  3.01 -1.82 -0.58 -0.57 -0.44 ...
##  $ Na  : num  13.6 13.5 13.3 13.3 13.2 ...
##  $ Mg  : num  4.49 3.55 3.62 3.6 3.61 3.46 3.66 3.56 3.59 3.54 ...
##  $ Al  : num  1.1 1.54 1.24 1.14 1.05 1.56 1.27 1.27 1.31 1.23 ...
##  $ Si  : num  71.8 73 73.1 73.1 73.2 ...
##  $ K   : num  0.06 0.39 0.55 0.58 0.57 0.67 0.6 0.54 0.58 0.58 ...
##  $ Ca  : num  8.75 7.78 8.07 8.17 8.24 8.09 8.56 8.38 8.5 8.39 ...
##  $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Fe  : num  0 0 0 0 0 0.24 0 0.17 0 0 ...
##  $ type: Factor w/ 6 levels "WinF","WinNF",..: 1 1 1 1 1 1 1 1 1 1 ...

# visualize distribution of target variable
par(mfrow = c(1,2))
plot(table(train$type))
plot(table(test$type))
```




```r
# from the barplots we see that the levels of the target variable are unbalanced.
# The distribution of the target variable is similar in the train and test.

# Get a visual impression of pair-wise relationship between variables and target
pairs(train)
```




```
# from the last column in the pairs plot we see that none of the explanatory
# variables alone can separate the classes reasonably well.
```

- Use the training data to fit a classification RF. Set the arguments to `importance=TRUE` for a later assessment of the importance of the different explanatory variables on the target and `ntree=1000`.
  - How large is the out-of-bag error over all classes?
  - Which class(es) are especially hard to classify correctly?
  - Which class(es) are most easy to classify correctly?

```
library(randomForest)
## Warning: package 'randomForest' was built under R version 4.0.3
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
set.seed(0815)
rf1 = randomForest(type ~ ., data = train, ntree = 1000, importance = TRUE )
print(rf1)
##
## Call:
##  randomForest(formula = type ~ ., data = train, ntree = 1000,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 24.31%
```

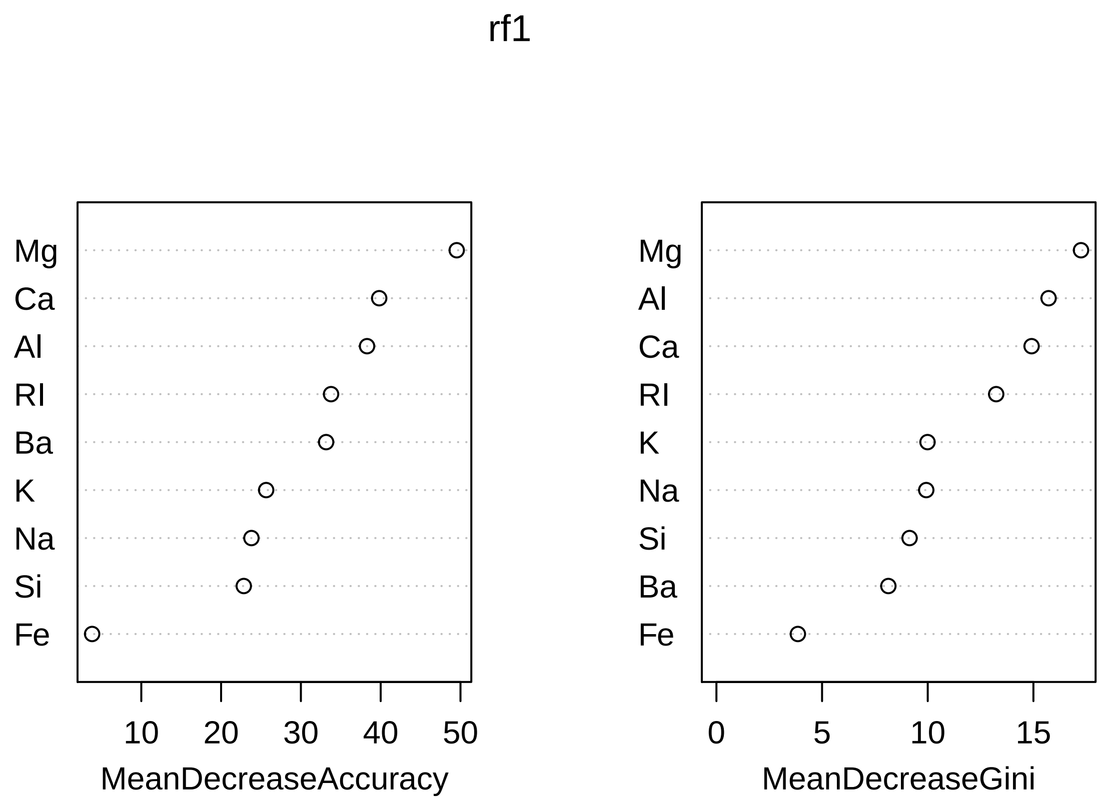
```
## Confusion matrix:
##       WinF WinNF Veh Con Tabl Head class.error
## WinF    48     5   1   0    0    0   0.1111111
## WinNF   10    39   0   1    1    0   0.2352941
## Veh      8     1   0   0    0    0   1.0000000
## Con      0     3   0   5    0    1   0.4444444
## Tabl     1     1   0   0    4    0   0.3333333
## Head     0     2   0   0    0   13   0.1333333

# OOB estimate of the error rate is ~24%
# VEH is the hardest class to predict with an error of 100%
# WinF and Head appear to be easy to predict with low error of about 10%
```

- Use the trained RF to predict the classes in the test data. Determine the test confusion matrix, the accuracy and the misclassification rate. Comment on your results.

```
# predict the test data
rf1.pred = predict(rf1, newdata = test, type = "class")

# Confusion matrix
(table2 = table(rf1.pred, test$type))
##
## rf1.pred WinF WinNF Veh Con Tabl Head
##    WinF    12     3   5   0    0    1
##    WinNF    4    22   2   1    0    2
##    Veh      0     0   1   0    0    0
##    Con      0     0   0   3    0    0
##    Tabl     0     0   0   0    3    0
##    Head     0     0   0   0    0   11

# accuracy
accuracy2 = sum(diag(table2))/sum(table2)
accuracy2
## [1] 0.7428571

# The accuracy is ~74%.
# This corresponds to a misclassification rate of 26%: 1 - 0.74 = 0.26

# This is similar to the OOB misclassification rate in the training data.
# The OOB error in the training data is achieved based on trees in the RF that
# did not see the observations in the bootstrap-train-sample.
# Since the training and the test data come from the same distribution, we
# have expected similar misclassification rates in the training and test data set.
```

- Which explanatory variables are most important for the classification?

```
varImpPlot(rf1)
```




```
# the concentration of Mg is most useful for the classification
```

Exercise 2: Random Forest versus lm for a regression model with continuous outcome

- The data set Boston is available in the package `MASS`. Load it and explore the help page to grab a minimal understanding of the data.

```
library(MASS)

data(Boston)
#help("Boston")

dim(Boston)
## [1] 506  14
# 506 obs., 14 variables
str(Boston)
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
```

```
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...

# We see chas is not correctly coded:
Boston$chas <- as.factor(Boston$chas)
```

- Randomly split the data into two subsets, a training and a test data set, using the proportion of 70% - 30%.

```
# We randomly sample the row indexes of the data set:
set.seed(1)
idx.tr <- sample(x = 1:nrow(Boston), size = as.integer(0.7 * nrow(Boston)))

train <- Boston[idx.tr, ]
test <- Boston[-idx.tr, ]
```

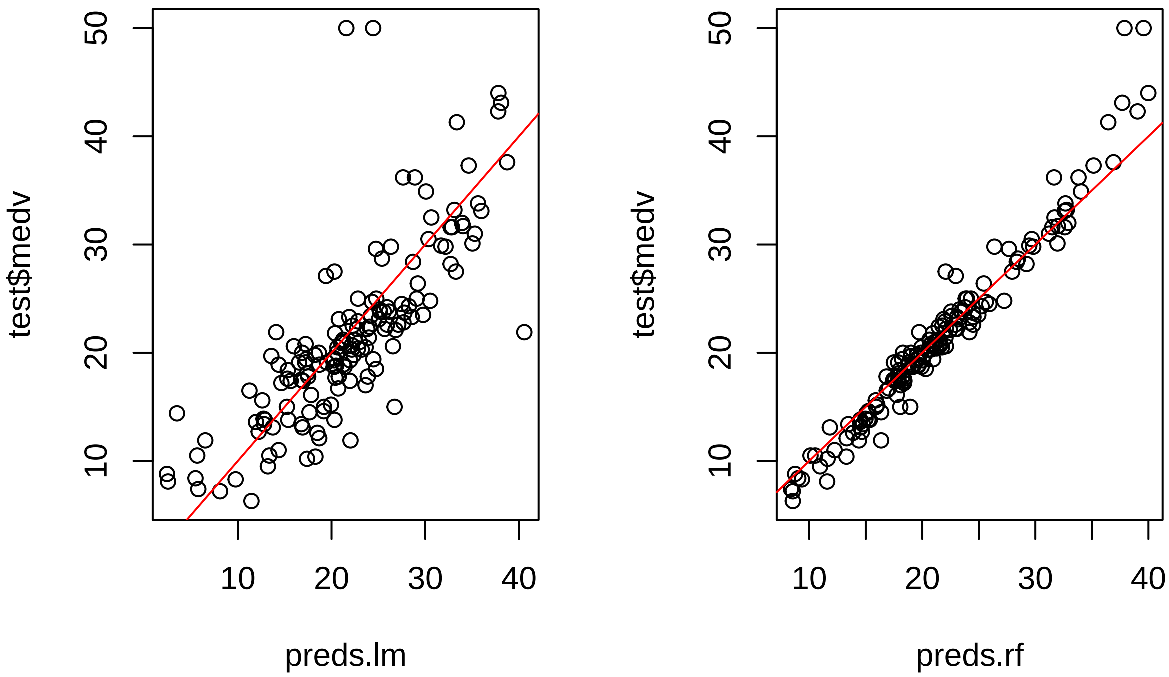
- Fit a regression model (once with lm and once with randomForest) with `medv` as target variable and all other variables as predictors. Fit the models using the training set.

```
# Linear Regression Model with "medv" as target and 13 covariates:
fit.lm <- lm(medv ~ ., data = train)
summary(fit.lm)
##
## Call:
## lm(formula = medv ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1732  -2.7272  -0.5408   1.6594  23.9450
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  28.209249   6.094508   4.629 5.24e-06 ***
## crim         -0.070644   0.042379  -1.667 0.096439 .
## zn            0.036046   0.016006   2.252 0.024959 *
## indus         0.030055   0.069986   0.429 0.667874
## chas1         3.419418   0.933343   3.664 0.000288 ***
## nox         -14.582615   4.374454  -3.334 0.000952 ***
## rm            4.856324   0.493995   9.831  < 2e-16 ***
## age          -0.014469   0.015251  -0.949 0.343431
## dis          -1.411125   0.231606  -6.093 3.01e-09 ***
## rad           0.303706   0.075162   4.041 6.59e-05 ***
## tax          -0.013742   0.004219  -3.257 0.001238 **
## ptratio      -0.941851   0.154394  -6.100 2.88e-09 ***
## black         0.010698   0.003084   3.469 0.000590 ***
## lstat        -0.480384   0.061173  -7.853 5.37e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.593 on 340 degrees of freedom
## Multiple R-squared:  0.7778, Adjusted R-squared:  0.7693
```

```
## F-statistic: 91.52 on 13 and 340 DF,  p-value: < 2.2e-16

# we assume that the model assumptions for lm are fulfilled

# Now the RF for the continuous outcome medv:
library(randomForest)
fit.rf <- randomForest(medv ~ ., data = test, importance = TRUE)
fit.rf
##
## Call:
##  randomForest(formula = medv ~ ., data = test, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 21.27425
##                     % Var explained: 68.43

# for the RF there are no model assumptions that need to be checked
```

- Get the predictions for the test data using the fitted models (lm and rf) and plot the observed `medv` values in the test set versus the predicted values. Based on the plot – how do both models compare?

```
par(mfrow = c(1, 2))

# For the lm Model:
preds.lm <- predict(fit.lm, test)
plot(preds.lm, test$medv)
abline(0, 1, col = "red")

#For the rf:
preds.rf <- predict(fit.rf, test)
plot(preds.rf, test$medv)
abline(0, 1, col = "red")
```




```
# All models yield quite unbiased predictions
# but the variance of the RF predictions is much lower
```

- Calculate the mean squared error (MSE) of these predictions on the test set. Is the MSE better for lm or for the random forest? (Hint: MSE $= \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$)

```
(mse.lm <- mean((preds.lm - test$medv)^2))
## [1] 27.31196

(mse.rf <- mean((preds.rf - test$medv)^2))
## [1] 4.143477

# The random forest model yields a much lower MSE compared to the linear
# regression model.
```

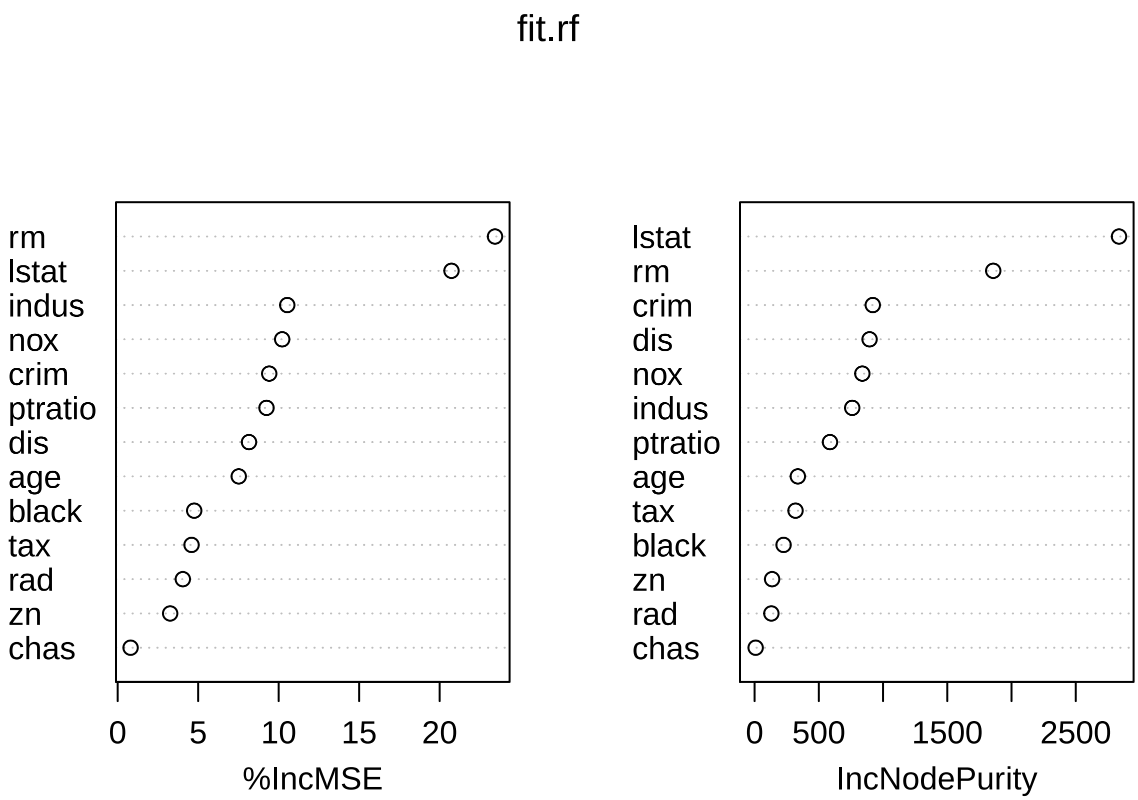
f) Assess the influence of the predictors `rm` and `lstat` in the linear model and the random forest. What do you oberseve. (R-Hint: `varImpPlot()`, `partialPlot()`)

```
summary(fit.lm)
```

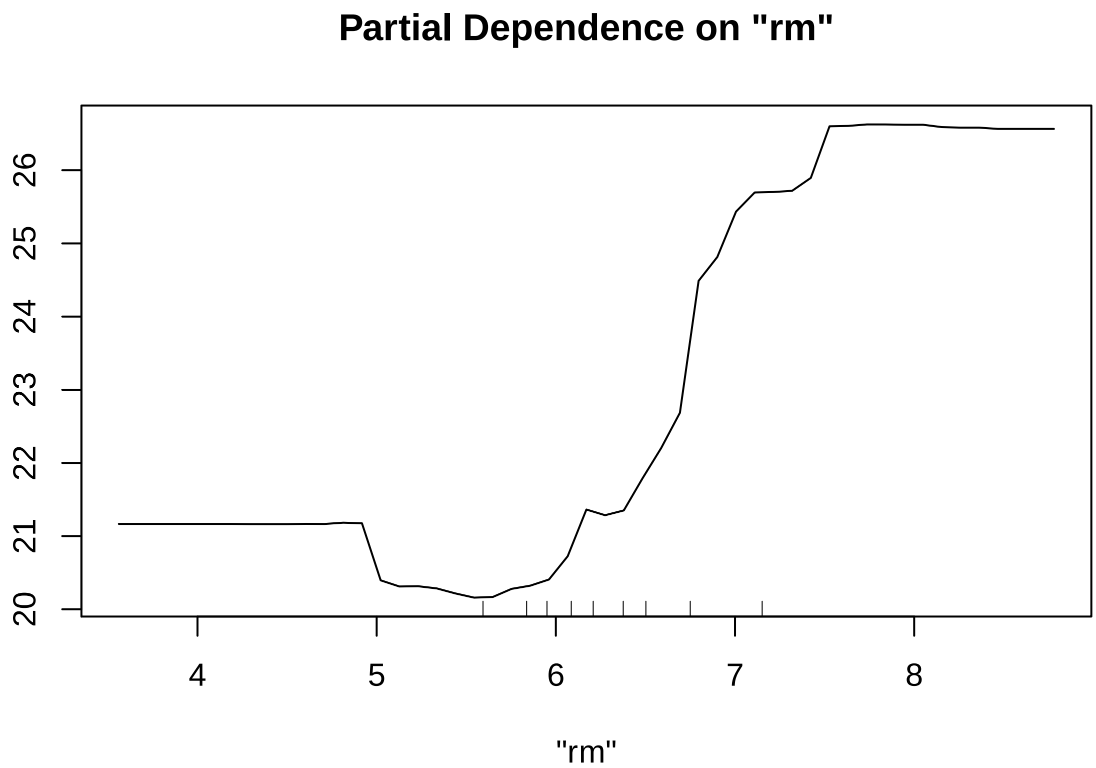
```
##
## Call:
## lm(formula = medv ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1732  -2.7272  -0.5408   1.6594  23.9450
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  28.209249   6.094508   4.629 5.24e-06 ***
## crim         -0.070644   0.042379  -1.667 0.096439 .
## zn            0.036046   0.016006   2.252 0.024959 *
## indus         0.030055   0.069986   0.429 0.667874
## chas1         3.419418   0.933343   3.664 0.000288 ***
## nox         -14.582615   4.374454  -3.334 0.000952 ***
## rm            4.856324   0.493995   9.831  < 2e-16 ***
## age          -0.014469   0.015251  -0.949 0.343431
## dis          -1.411125   0.231606  -6.093 3.01e-09 ***
## rad           0.303706   0.075162   4.041 6.59e-05 ***
## tax          -0.013742   0.004219  -3.257 0.001238 **
## ptratio      -0.941851   0.154394  -6.100 2.88e-09 ***
## black         0.010698   0.003084   3.469 0.000590 ***
## lstat        -0.480384   0.061173  -7.853 5.37e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.593 on 340 degrees of freedom
## Multiple R-squared:  0.7778, Adjusted R-squared:  0.7693
## F-statistic: 91.52 on 13 and 340 DF,  p-value: < 2.2e-16
```

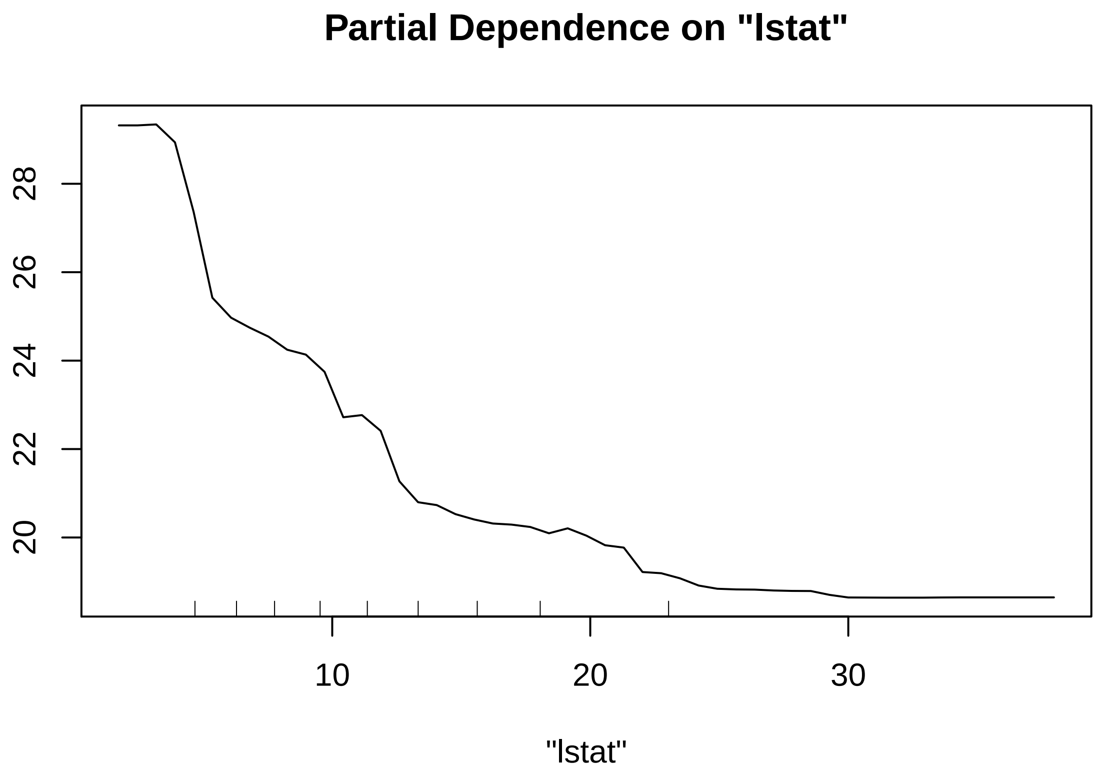
```r
# both variables are highly significant. If rm increases by 1 unit
# (and everything else remains the same), medv increases by 3.84.
# If lstat increases by 1 unit (and everything else remains the same),
# medv decreases by -0.56.

varImpPlot(fit.rf)
```




```r
# rm and lstat are the most important variables

# check the marginal dependency of medv on rm and lstat
partialPlot(fit.rf, Boston, x.var = "rm")
```




```r
# If rm increases, medv increases (in a non-linear manner)

partialPlot(fit.rf, Boston, x.var = "lstat")
```




```
# If lstat increases, medv decreases (in a non-linear manner)

# This is in concordance with the effect of the variables in the
# linear regression model.
```