Machine Learning I Group Work

Adriana Ricklin, Yvonne Schaerli, Christina Sudermann, Carole Mattmann

11 June 2020

Contents

1 Packages

```
library(gridExtra)
library(tree)
```

```r
library(dplyr)
library(readr)
library(randomForest)
library(e1071)     # for SVM
library(ISLR)
library(tibble)
library(MASS)
library(gbm)
library(ipred)
library(neuralnet) # for NN
library(h2o)       # for NN
library(bit64)     # for NN
library(h2o)       # for NN
library(ggplot2)
library(tidyverse) # inclued ggplot2
library(mgcv)
```

2 Import and data cleaning

```r
#getwd()
#setwd("~/GitHub/machinelearning/machinelearning/03_rmarkdown") #adrianas extrawurst
insurance <- read.csv("../01_data/insurance.csv", header=TRUE)
str(insurance)
```

```
## 'data.frame':    1338 obs. of  7 variables:
##  $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 1 1 2 1 ...
##  $ bmi     : num  27.9 33.8 33 22.7 28.9 ...
##  $ children: int  0 1 3 0 0 0 1 3 2 0 ...
##  $ smoker  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...
##  $ region  : Factor w/ 4 levels "northeast","northwest",..: 4 3 3 2 2 3 3 2 1 2 ...
##  $ charges : num  16885 1726 4449 21984 3867 ...
```

```r
# smoker = 1 / nonsmoker = 0
insurance$smoker <- as.character(insurance$smoker)
insurance$smoker[insurance$smoker == "yes"] <- "1"
insurance$smoker[insurance$smoker == "no"] <- "0"
insurance$smoker <- as.factor(insurance$smoker)
# female = 1 / male = 0
insurance$sex <- as.character(insurance$sex)
insurance$sex[insurance$sex == "female"] <- "1"
insurance$sex[insurance$sex == "male"] <- "0"
insurance$sex <- as.factor(insurance$sex)
# region / SE = 1 / SW = 0 / NE = 2 / NW = 3
insurance$region <- as.character(insurance$region)
insurance$region[insurance$region == "southwest"] <- "1"
insurance$region[insurance$region == "southeast"] <- "0"
insurance$region[insurance$region == "northeast"] <- "2"
insurance$region[insurance$region == "northwest"] <- "3"
insurance$region <- as.factor(insurance$region)
head(insurance)
```

```
##   age sex    bmi children smoker region   charges
```

```
## 1  19    1 27.900        0       1       1 16884.924
## 2  18    0 33.770        1       0       0  1725.552
## 3  28    0 33.000        3       0       0  4449.462
## 4  33    0 22.705        0       0       3 21984.471
## 5  32    0 28.880        0       0       3  3866.855
## 6  31    1 25.740        0       0       0  3756.622
```

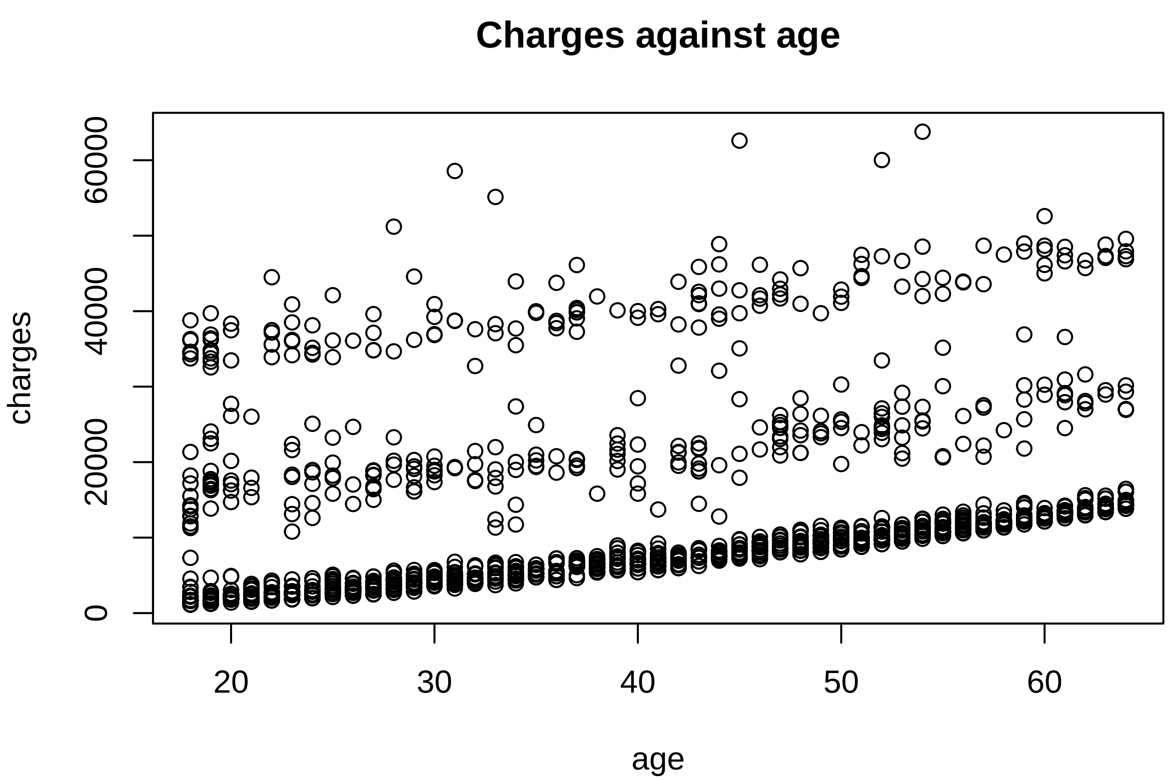
3 Linear models -> Christina

The data set used for this project contains 1338 observations and seven variables containing the age, sex, bmi, children, smoker, region and charges. Those variables are covering continuous and categorical variables.The region as an categorical variable is covering four different levels.

3.1 Basic analysis of continuous variables

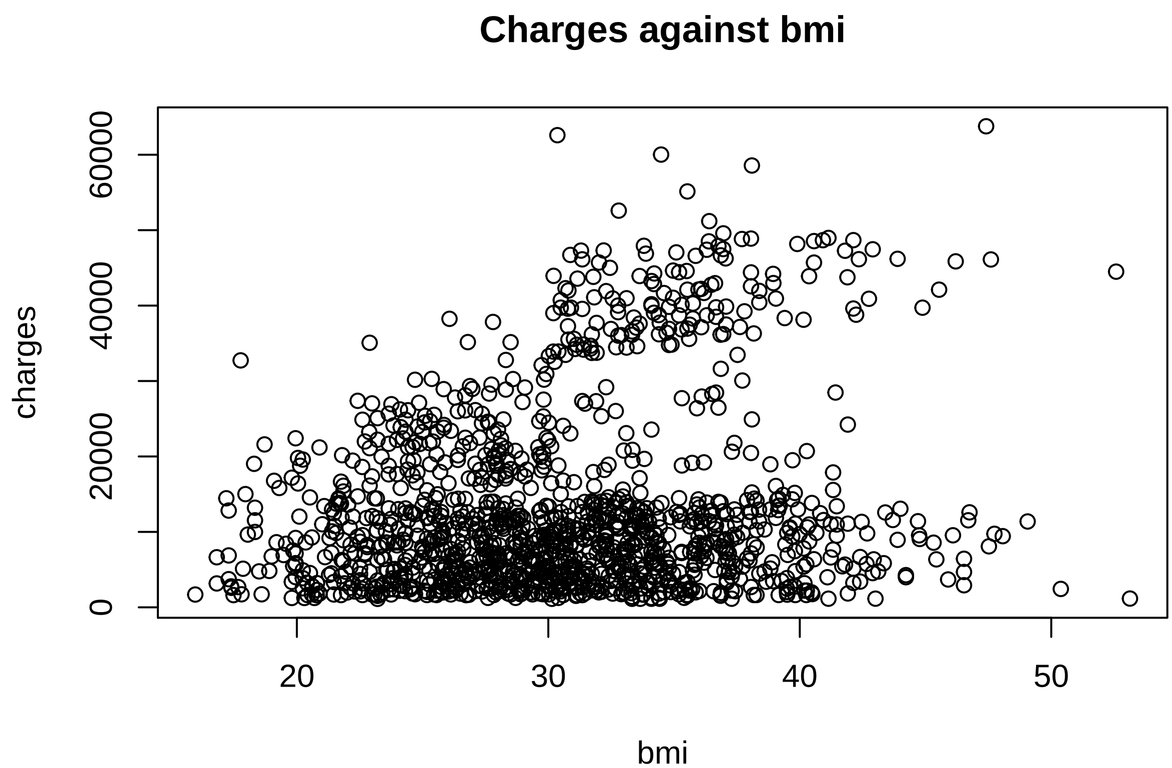
Here we will begin with a graphical analysis. Therefore we will plot the response variable against the given predictors to gather first relationships, which can be used later in the modelling process.

```
plot(charges ~ age, data =insurance, main = 'Charges against age')
```



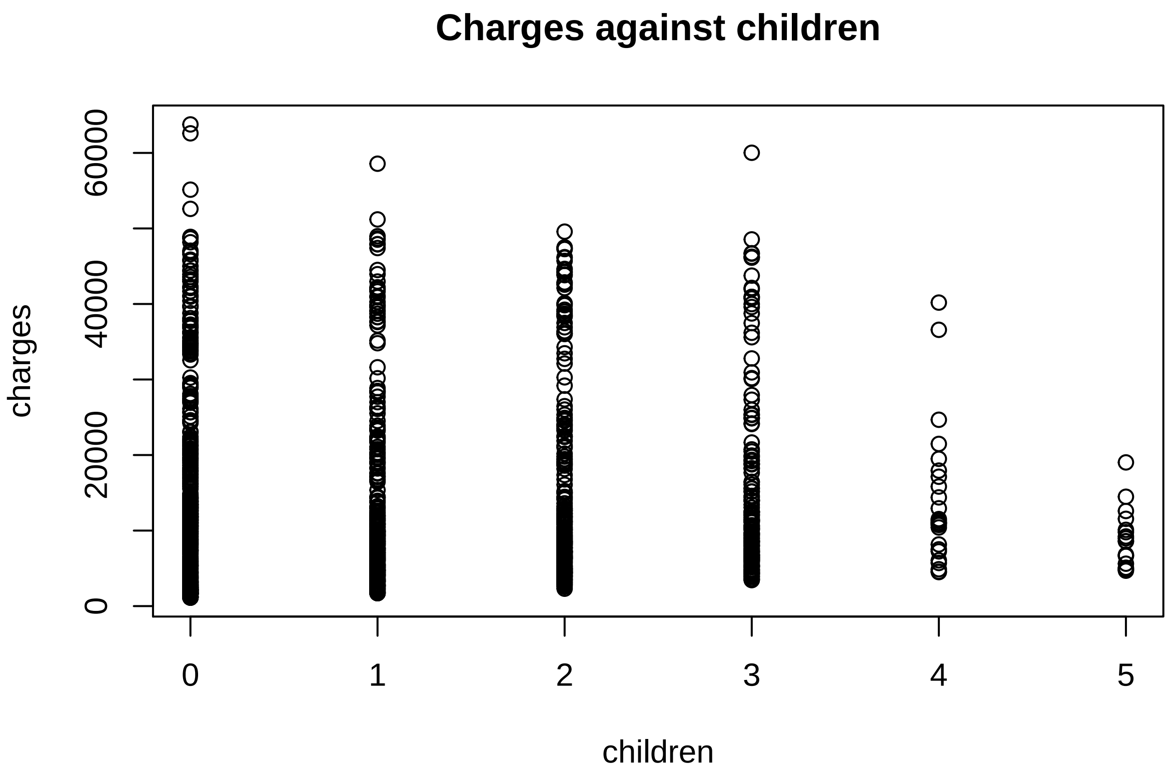
There seems to be a positive relationship between age and charges.

```r
plot(charges ~ bmi, data =insurance, main = 'Charges against bmi')
```



The above plot is not showing a clear relationship between bmi and the corresponding charges.

```r
plot(charges ~ children, data =insurance, main = 'Charges against children')
```



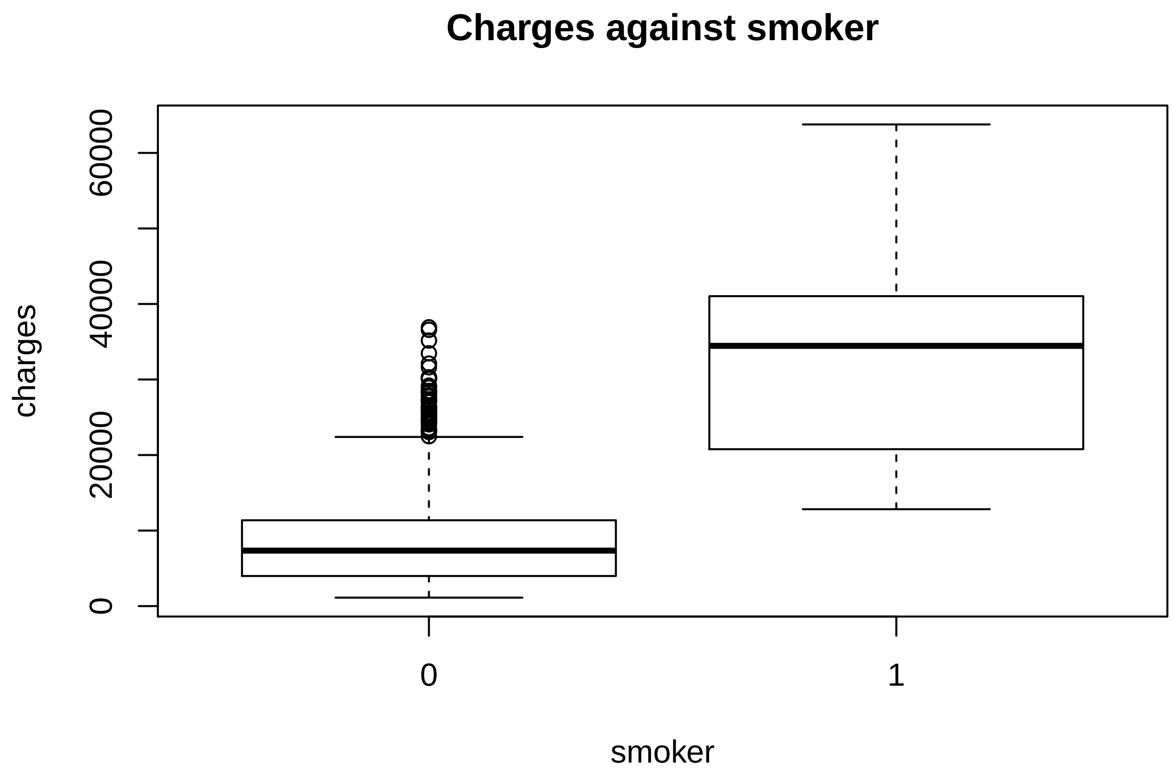
There might be an affect between the number of children and the charges. As shown above it might be possible to interpret that with a rising number of children the charges a decreasing. Still this is not a clear relationship, more a wide interpretation of the plot.

3.2 Basic analysis of categorical variables

```
##plot(charges ~ sex, data =insurance, main = 'Charges against sex')
```

As shown above there are some differences in the charges, comparing the boxplot for the two given genders. It seems that the range for 50% of the obersavation is bigger than the one for the female group. Also the 95% quantile is about 10'000 lower than the same quantile for the male group.

```
plot(charges ~ smoker, data =insurance, main = 'Charges against smoker')
```



This plot is showing a clear affect of smoking on the charges. As you can see persons who are not smoking having mean charges that are three times less as the one of people who are smoking. even the outliers of the smokers are still in the range where only 50% of the smokers are located.

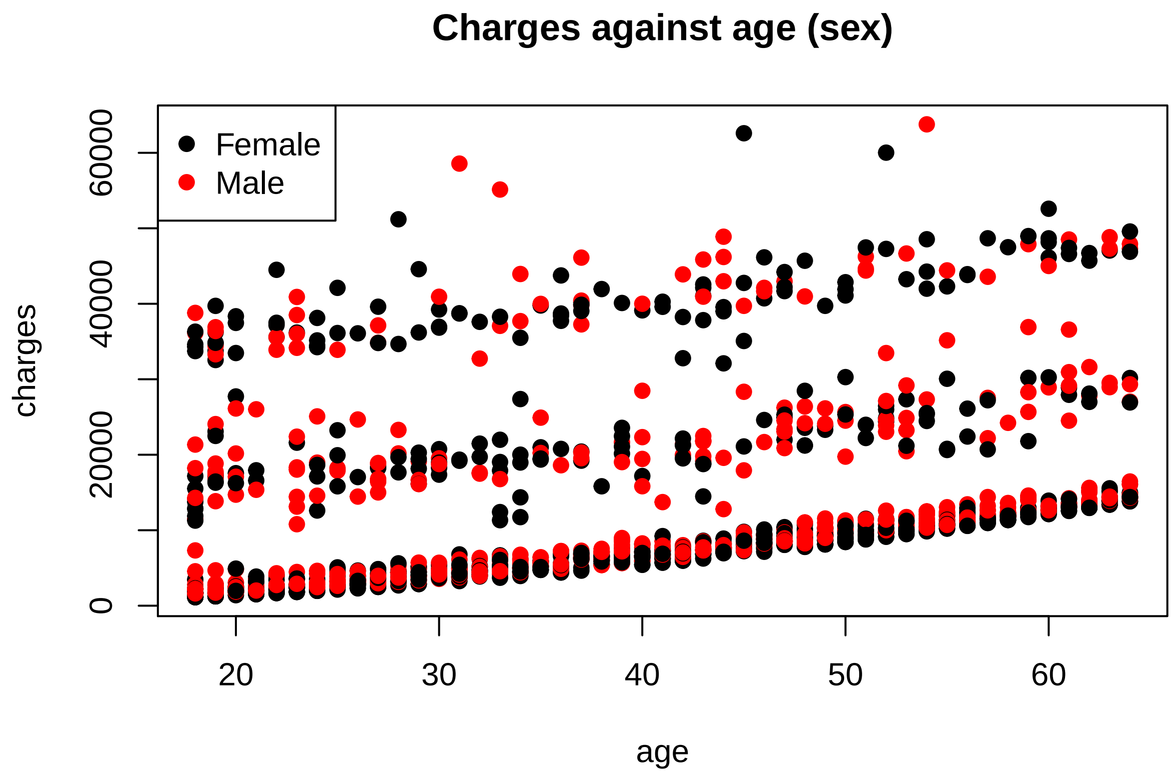
```
##plot(charges ~ region, data =insurance, main = 'Charges against region')
```

In this case there isn't any clear realtionship or trend visible between the region and the corresponding charges. Comparing the four regions together seems that there are slightly small diferences in the width of the box, distribution of 50% of the observation as well the setting of the 95% quantile. Those differences will be analysed later.

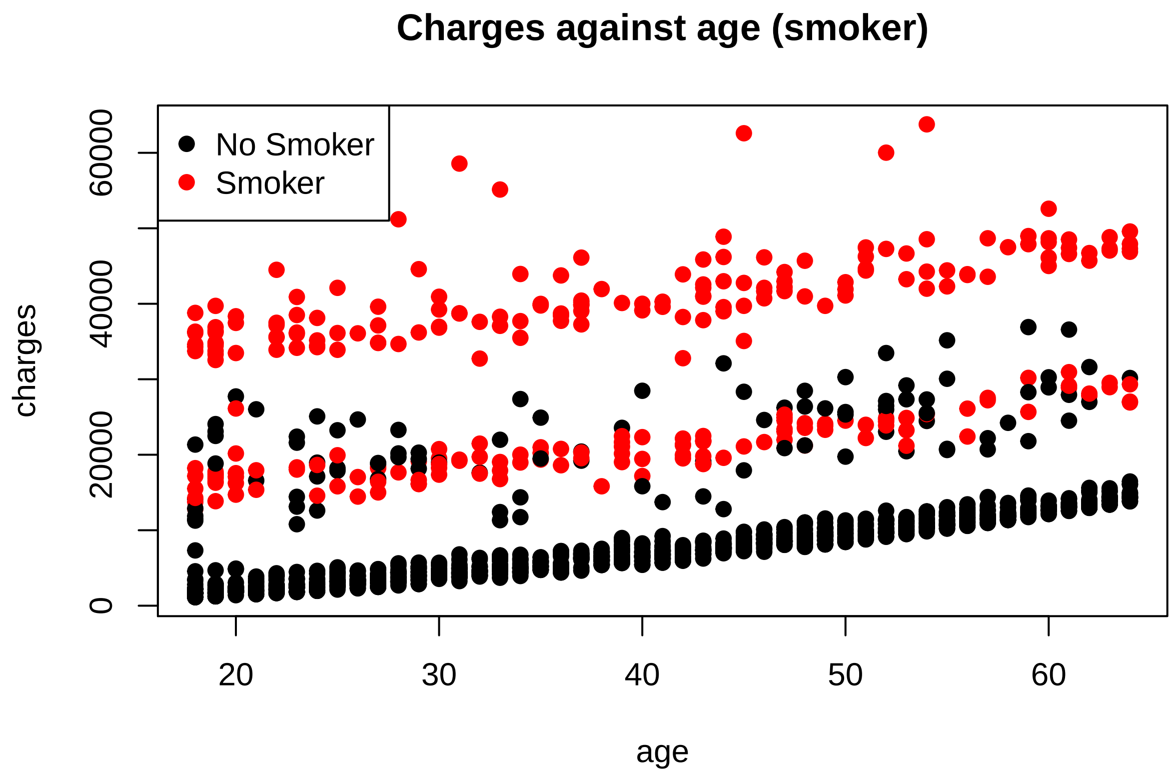
3.3 Basic analysis of categorical and continous variables

In this part we want to show exemplary how the relationship between charges and the age can be affected by adding additionally a categorical variable. This can be used for further data explorations and as well be adapted on all other variables. For this section we will just perform this enlargement for the case of the age to illustrate some possible relationships, which also can be helpful for the following modeling process.

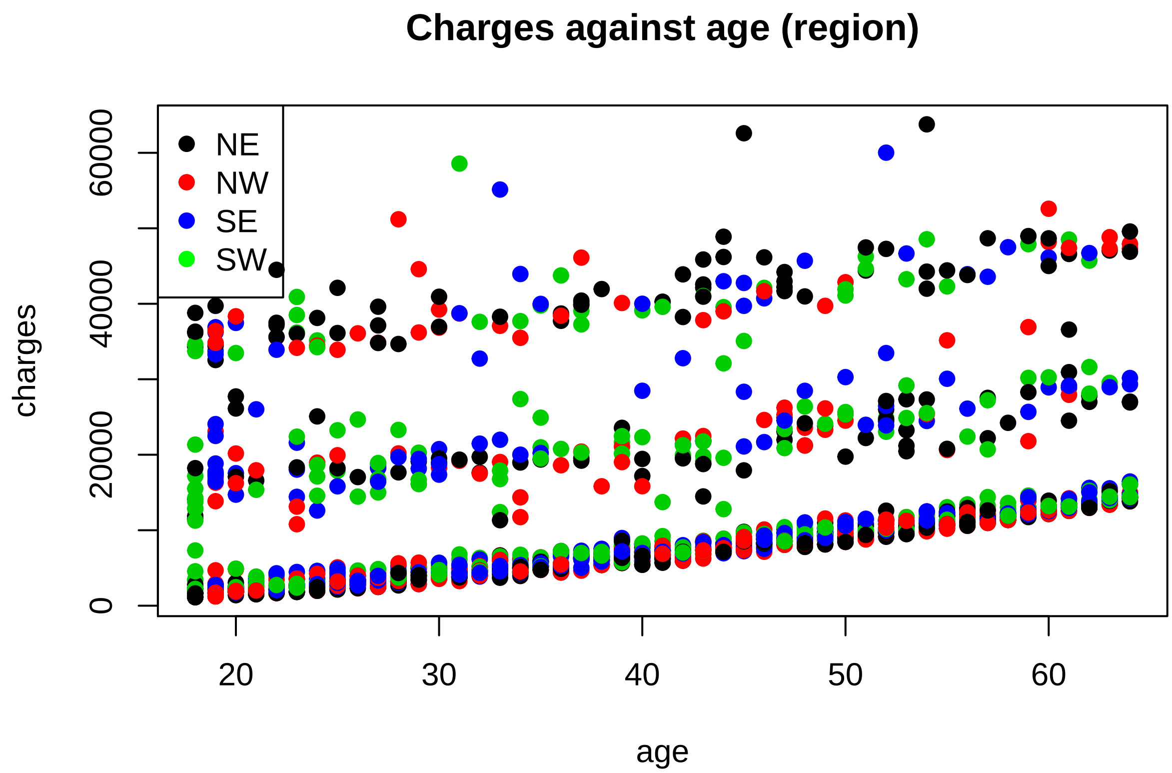
```
plot(charges ~ age, data = insurance,
     col = sex,
     pch = 19,
     main = "Charges against age (sex)")
legend( "topleft",
        pch = 19,
        legend = c("Female", "Male"),
        col = c("black", "red"))
```




```
plot(charges ~ age, data = insurance,
     col = smoker,
     pch = 19,
     main = "Charges against age (smoker)")
legend( "topleft",
        pch = 19,
        legend = c("No Smoker", "Smoker"),
        col = c("black", "red"))
```

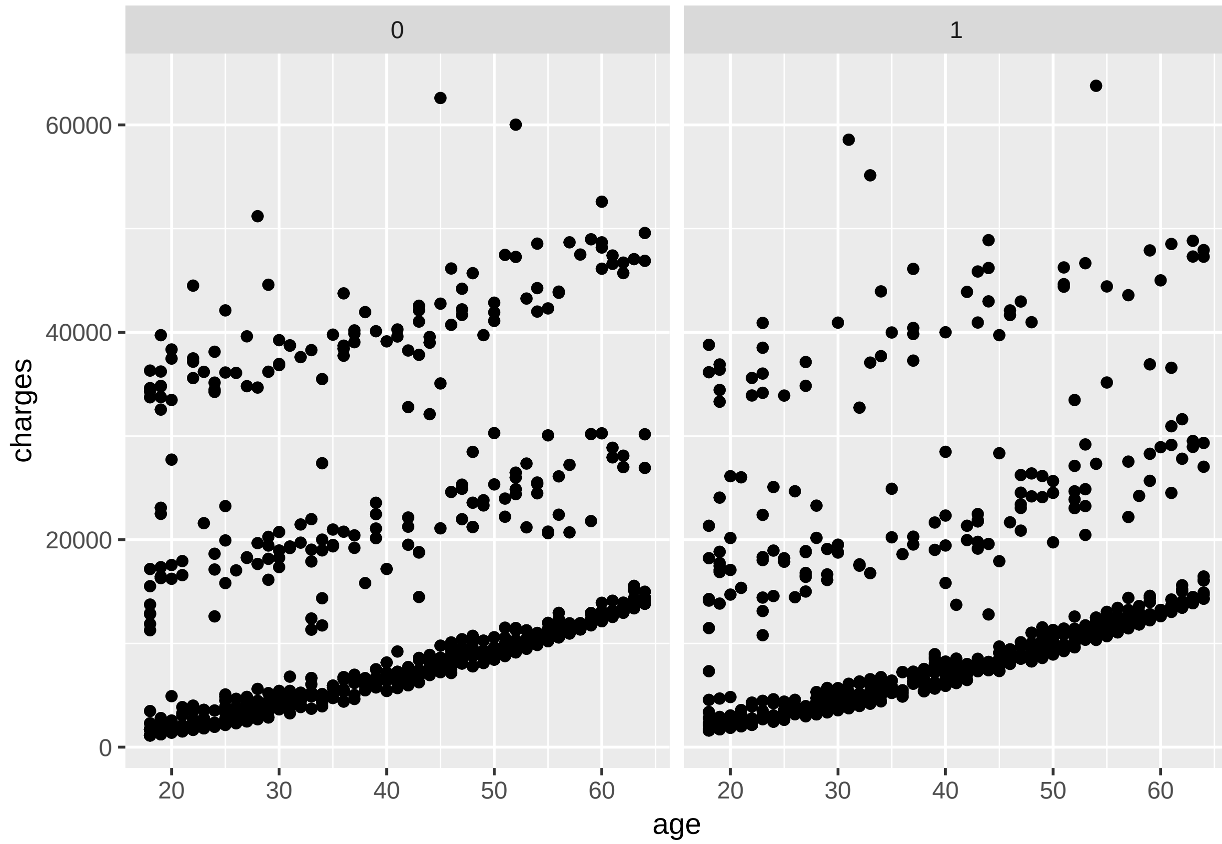



```
plot(charges ~ age, data = insurance,
     col = region,
     pch = 19,
     main = "Charges against age (region)")
legend( "topleft",
        pch = 19,
        legend = c("NE", "NW","SE","SW"),
        col = c("black", "red" ,"blue" , "green"))
```



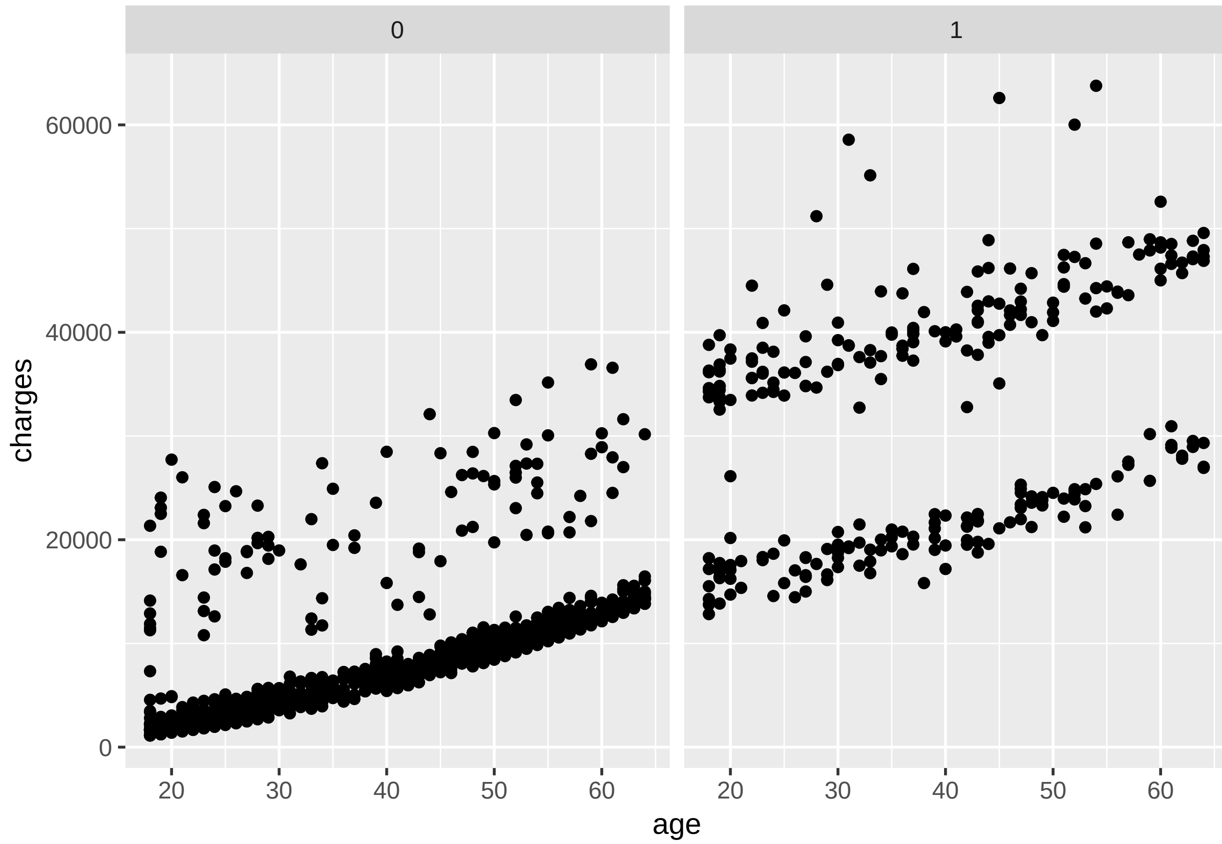
Since it is possible to have overlapping observations, we will plot the same set-up using the qplot. Having separate facets will avoid overlapping and therefore might serve different results as already seen above.

```
qplot(y=charges, x=age,
      data = insurance,
      facets = ~ sex)
```

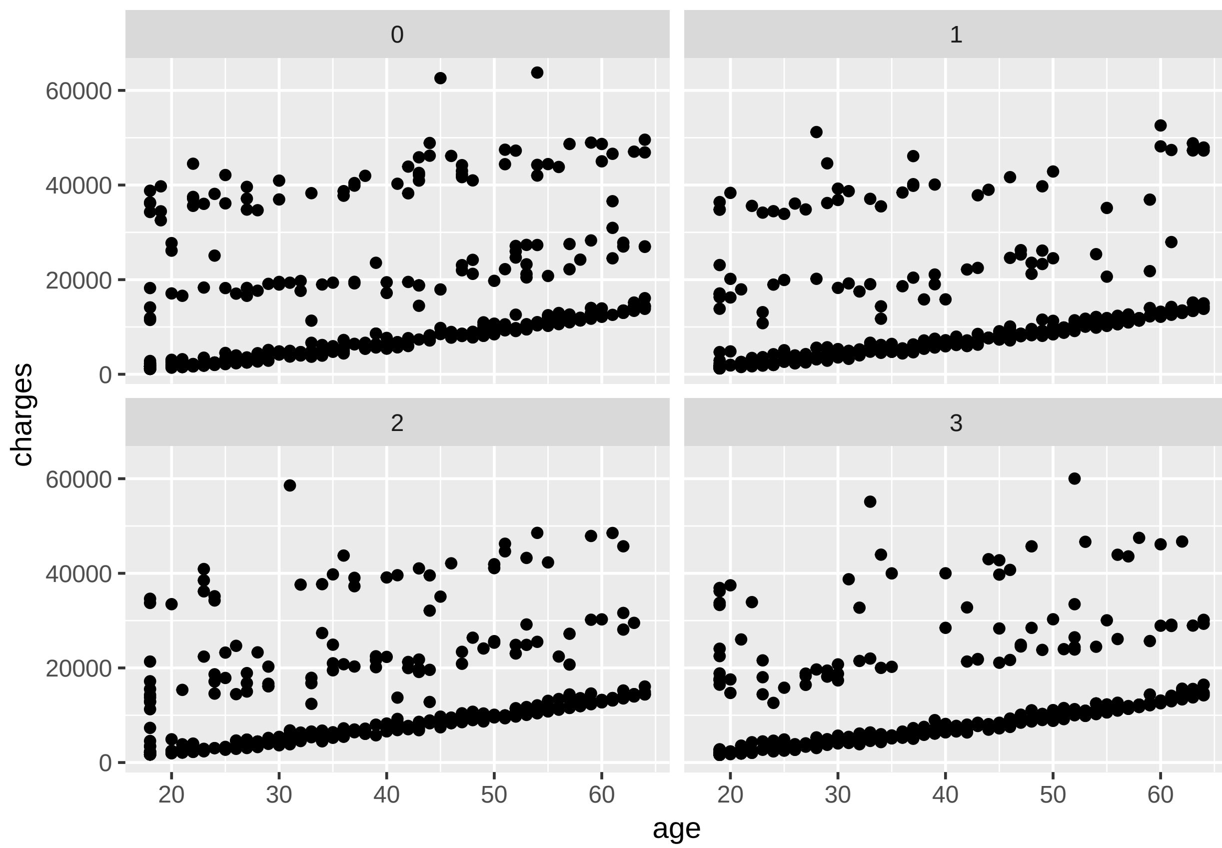



```r
lm.insurance <- lm(charges ~ age, data = insurance)
```

```r
qplot(y=charges, x=age,
      data = insurance,
      facets = ~ smoker)
```




```
qplot(y=charges, x=age,
      data = insurance,
      facets = ~ region)
```



3.4 Fitting a first Linear Model

As usually we will start by fitting a simple regression model to the insurance dataset. Therfore we will start with one variable and add additonal complexity in each following subset. This step-by-step approach helps to explore the data slightly better and will simplify the final modelling.

3.4.1 Linear Model with one variable

For the first simple regression model we will use the age.

```
lm.insurance <- lm(charges ~ age, data = insurance)
summary(lm.insurance)
```

```
##
## Call:
## lm(formula = charges ~ age, data = insurance)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -8059  -6671  -5939   5440  47829
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3165.9      937.1   3.378 0.000751 ***
## age            257.7       22.5  11.453  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 11560 on 1336 degrees of freedom
## Multiple R-squared:  0.08941,    Adjusted R-squared:  0.08872
## F-statistic: 131.2 on 1 and 1336 DF,  p-value: < 2.2e-16
```

3.4.1.1 Coefficent and Interpretation

As shown in the R Output a person with the age of 0 will have a base charge of 3165.885. This intercept and its interpreation seems to be nonsensical. With each year the charges are increasing by 257.7226, which is the slope for this regressionline

3.4.1.2 P-values

With a value of 2e-16 the age seesm to have a very strong effect on the charges. This means that the slope of the chages is not flat, not zero, so the hypothesisi has to be thrown away.

3.4.1.3 Including the gender as a second variable

In this subsest we will consider also the sex for modelling a simple regression model.

```r
lm.insurance.2 <- lm(charges ~ age + sex, data = insurance)
summary(lm.insurance.2)
```

```
##
## Call:
## lm(formula = charges ~ age + sex, data = insurance)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -8821  -6947  -5511   5443  48203
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3882.46     980.49   3.960  7.9e-05 ***
## age           258.87      22.47  11.523  < 2e-16 ***
## sex1        -1538.83     631.08  -2.438   0.0149 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11540 on 1335 degrees of freedom
## Multiple R-squared:  0.09344,    Adjusted R-squared:  0.09209
## F-statistic:  68.8 on 2 and 1335 DF,  p-value: < 2.2e-16
```

3.4.1.4 Including interaction

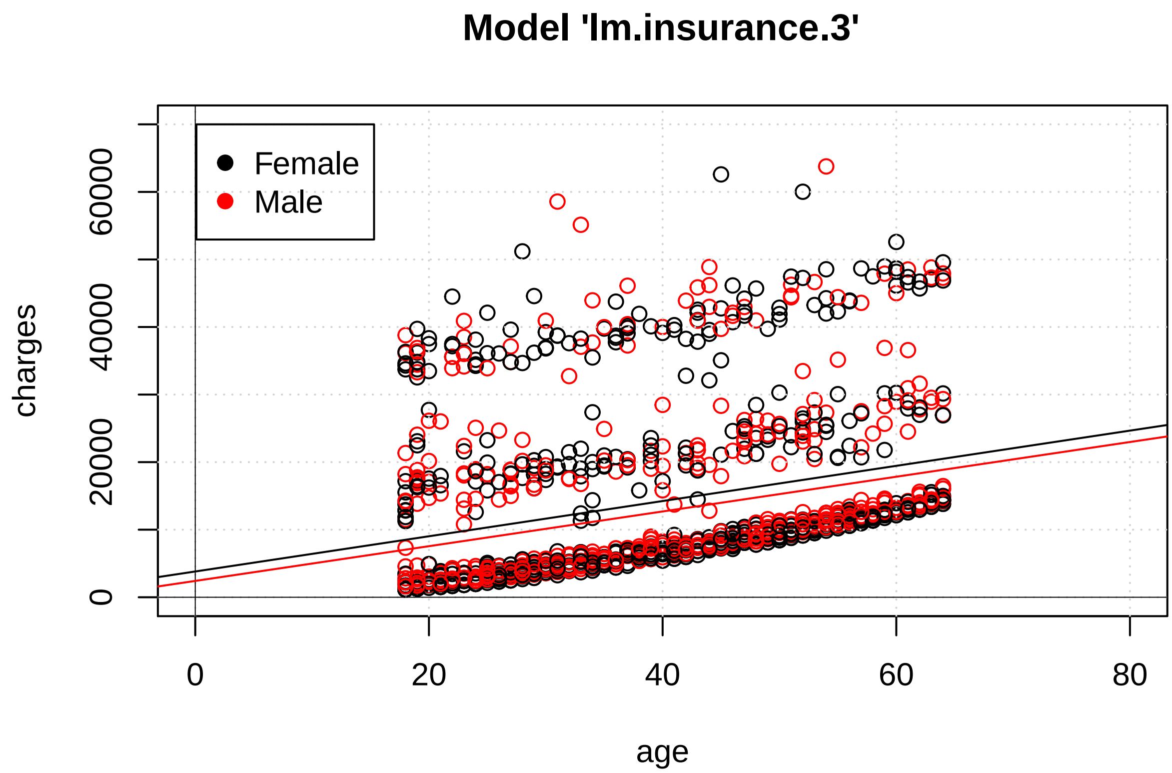
Since a second variable was added we want to explore if there might be significant interaction between age and sex, that might have to be considered for the modeling process.

```r
lm.insurance.3 <-lm(charges ~ age * sex, data =insurance)
summary(lm.insurance.3)
```

```
##
## Call:
## lm(formula = charges ~ age * sex, data = insurance)
##
## Residuals:
##    Min     1Q Median     3Q    Max
```

```
## -8823  -6936  -5500   5456  48187
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3811.77    1308.29   2.914  0.00363 **
## age           260.68      31.62   8.244 3.96e-16 ***
## sex1        -1394.92    1872.30  -0.745  0.45638
## age:sex1       -3.67      44.95  -0.082  0.93494
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11540 on 1334 degrees of freedom
## Multiple R-squared:  0.09345,    Adjusted R-squared:  0.09141
## F-statistic: 45.84 on 3 and 1334 DF,  p-value: < 2.2e-16
```

```r
plot(charges ~ age, data =insurance,
     main = "Model 'lm.insurance.3'",
     xlim = c(0, 80),
     ylim = c(0, 70000),
     col = sex)
##
grid()
##
abline(h = 0, lwd = 0.5)
abline(v = 0, lwd = 0.5)
##
abline(a = coef(lm.insurance.3)[1],
       b = coef(lm.insurance.3)["age"])
abline(a = coef(lm.insurance.3)[1] + coef(lm.insurance.3)["sex1"] ,
       b = coef(lm.insurance.3)["age"] + coef(lm.insurance.3)["age:sex1"],
       col = "red")
legend(x = 0.1, y = 70000,
       pch = 19,
       legend = c("Female","Male"),
       col = c("black", "red"))
```




```
coef(lm.insurance.3)
```

```
##   (Intercept)           age          sex1      age:sex1
##   3811.773852    260.681339 -1394.925331     -3.669849
```

```
summary(lm.insurance.3)$coefficients
```

```
##                 Estimate Std. Error      t value     Pr(>|t|)
## (Intercept)  3811.773852 1308.29248  2.91354869 3.633010e-03
## age           260.681339   31.62249  8.24354130 3.958052e-16
## sex1        -1394.925331 1872.29663 -0.74503437 4.563822e-01
## age:sex1       -3.669849   44.95054 -0.08164194 9.349437e-01
```

3.5 Final Linear Model Development

In this section we want to find an appropriate model, which accounts all relevant parameters and interactions. Afterwards we will then compare the fitted model with a base model and test its performance.

3.5.1 Linear Model with all variables

```
lm.insurance.all <- lm(charges ~ age + sex + bmi + children + smoker + region , data = insurance)
summary(lm.insurance.all)
```

```
##
## Call:
## lm(formula = charges ~ age + sex + bmi + children + smoker +
##     region, data = insurance)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11304.9  -2848.1   -982.1   1393.9  29992.8
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -13104.87    1090.51 -12.017  < 2e-16 ***
## age            256.86      11.90  21.587  < 2e-16 ***
## sex1           131.31     332.95   0.394 0.693348
## bmi            339.19      28.60  11.860  < 2e-16 ***
## children       475.50     137.80   3.451 0.000577 ***
## smoker1      23848.53     413.15  57.723  < 2e-16 ***
## region1         74.97     470.64   0.159 0.873460
## region2       1035.02     478.69   2.162 0.030782 *
## region3        682.06     478.96   1.424 0.154669
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6062 on 1329 degrees of freedom
## Multiple R-squared:  0.7509, Adjusted R-squared:  0.7494
## F-statistic: 500.8 on 8 and 1329 DF,  p-value: < 2.2e-16
```

As the above R Output shows not all varaibles seems to have a significant effect on the charges.

3.5.1.1 Testing sex before dropping

Before removing those two variables we first will make a deeper analysis.

```r
lm.sex <- lm(charges ~ sex, data=insurance)
summary(lm.sex)
```

```
##
## Call:
## lm(formula = charges ~ sex, data = insurance)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -12835  -8435  -3980   3476  51201
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  13956.8      465.2  30.003   <2e-16 ***
## sex1         -1387.2      661.3  -2.098   0.0361 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12090 on 1336 degrees of freedom
## Multiple R-squared:  0.003282,   Adjusted R-squared:  0.002536
## F-statistic:   4.4 on 1 and 1336 DF,  p-value: 0.03613
```

```r
coef(lm.sex)
```

```
## (Intercept)        sex1
##   13956.751   -1387.172
```

As seen in the output considering only the sex it is not a significant and standalone explaining variable for the charges. Comparing the p-value of the sex within the full model including all possible variables it is so high with a value of 0.693348, that it can be dropped from our final model.

3.5.1.2 Testing region before dropping

```
lm.region.1 <- lm(charges ~ region, data = insurance)
aggregate(charges ~ region, FUN = mean, data =insurance)
```

```
##   region  charges
## 1      0 14735.41
## 2      1 12346.94
## 3      2 13406.38
## 4      3 12417.58
```

```
summary(lm.region.1)
```

```
##
## Call:
## lm(formula = charges ~ region, data = insurance)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -13614  -8463  -3793   3385  49035
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  14735.4      633.3  23.266   <2e-16 ***
## region1      -2388.5      922.2  -2.590   0.0097 **
## region2      -1329.0      922.9  -1.440   0.1501
## region3      -2317.8      922.2  -2.513   0.0121 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12080 on 1334 degrees of freedom
## Multiple R-squared:  0.006634,   Adjusted R-squared:  0.0044
## F-statistic:  2.97 on 3 and 1334 DF,  p-value: 0.03089
```

There is strong evidence that the mean charges for northeast is not eqaual to zero. But there isn't any evidence that all other regions differ from the reference region northeast. To have a better understanding we will make an anova test between the above shown model and a base model lm.region.0 as schon below.

```
lm.region.0 <- lm(charges ~ 1 , data = insurance)
coef(lm.region.0)
```

```
## (Intercept)
##    13270.42
```

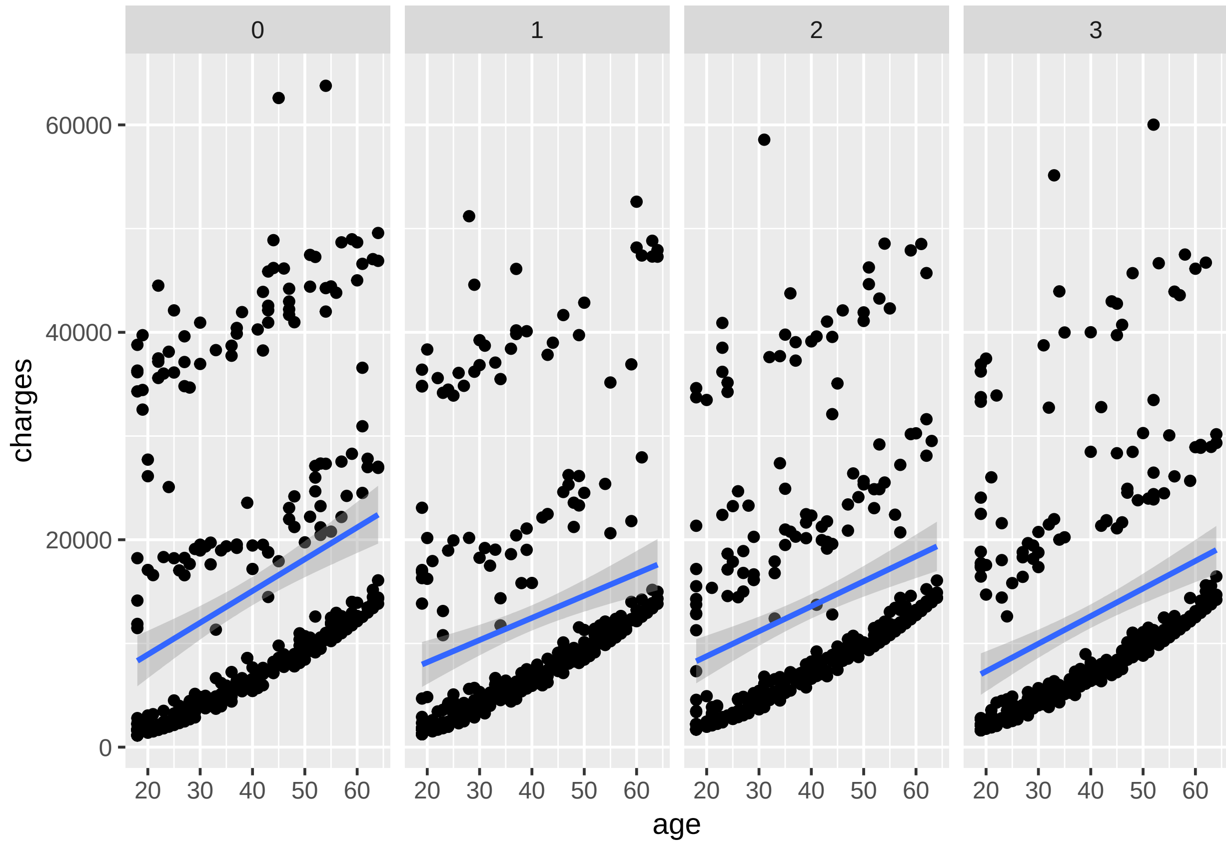
```
anova(lm.region.0, lm.region.1)
```

```
## Analysis of Variance Table
##
## Model 1: charges ~ 1
## Model 2: charges ~ region
##   Res.Df        RSS Df  Sum of Sq      F  Pr(>F)
## 1   1337 1.9607e+11
## 2   1334 1.9477e+11  3 1300759681 2.9696 0.03089 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The anova test shows that there is a low evidence that the model with more parameters(in this case only the region) better fits the data. The F-value seems to be very small and also the p-value is not really significant with 0.03089. Anyhow there is a drop in the RSS, by adding 3 additional parameters of the different regions. To have a better understanding we can additionally perform posthoc contrasts to decide afterwards if we will drop the region finally from our model. Before performing several posthoc test and repeating this exercise to all possible combinations we will use the ggplot to explore the data quickly upfront.

```
ggplot(data=insurance,
       mapping = aes(y = charges, x= age))+ geom_point()+geom_smooth(method = 'lm') + facet_grid(. ~reg
```



Using the visualization it seems that there seems to be no obvoius differnce and effect. Therfore we decide to drop also the variable region from our linear model.

3.5.2 Dropping variables from the model

```
drop1(lm.insurance.all, test="F")
```

```
## Single term deletions
##
## Model:
## charges ~ age + sex + bmi + children + smoker + region
##          Df  Sum of Sq        RSS   AIC   F value    Pr(>F)
## <none>                 4.8840e+10 23316
## age       1 1.7124e+10 6.5964e+10 23717  465.9837 < 2.2e-16 ***
## sex       1 5.7164e+06 4.8845e+10 23315    0.1556  0.693348
## bmi       1 5.1692e+09 5.4009e+10 23449  140.6627 < 2.2e-16 ***
```

```
## children  1 4.3755e+08 4.9277e+10 23326   11.9063  0.000577 ***
## smoker    1 1.2245e+11 1.7129e+11 24993 3331.9680 < 2.2e-16 ***
## region    3 2.3343e+08 4.9073e+10 23317    2.1173  0.096221 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
lm.insurance.1 <- update(lm.insurance.all, .~. -region -sex)
formula(lm.insurance.1)
```

```
## charges ~ age + bmi + children + smoker
```

```r
summary(lm.insurance.1)
```

```
##
## Call:
## lm(formula = charges ~ age + bmi + children + smoker, data = insurance)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11897.9  -2920.8   -986.6   1392.2  29509.6
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -12102.77     941.98 -12.848  < 2e-16 ***
## age            257.85      11.90  21.675  < 2e-16 ***
## bmi            321.85      27.38  11.756  < 2e-16 ***
## children       473.50     137.79   3.436 0.000608 ***
## smoker1      23811.40     411.22  57.904  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6068 on 1333 degrees of freedom
## Multiple R-squared:  0.7497, Adjusted R-squared:  0.7489
## F-statistic: 998.1 on 4 and 1333 DF,  p-value: < 2.2e-16
```

Comparing the summary output of lm.insurance.0 and lm.insurance.1 the latest model is only including variables having a p-value that indicates a significant factor.

3.5.3 Considering Interactions

```r
drop1(lm.insurance.1, test="F")
```

```
## Single term deletions
##
## Model:
## charges ~ age + bmi + children + smoker
##          Df  Sum of Sq        RSS   AIC  F value    Pr(>F)
## <none>                 4.9078e+10 23315
## age       1 1.7297e+10 6.6375e+10 23717  469.789 < 2.2e-16 ***
## bmi       1 5.0884e+09 5.4167e+10 23445  138.203 < 2.2e-16 ***
## children  1 4.3477e+08 4.9513e+10 23325   11.809 0.0006077 ***
## smoker    1 1.2345e+11 1.7253e+11 24995 3352.911 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
lm.insurance.2 <- update(lm.insurance.1, .~. +age:bmi + age:children + age:smoker + bmi:children + bmi:s
summary(lm.insurance.2)
```

```
##
## Call:
## lm(formula = charges ~ age + bmi + children + smoker + age:bmi +
##     age:children + age:smoker + bmi:children + bmi:smoker + children:smoker,
##     data = insurance)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13996.3  -1947.6  -1331.5   -406.4  29570.2
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -6.745e+02  2.106e+03  -0.320    0.749
## age               2.055e+02  4.963e+01   4.140  3.7e-05 ***
## bmi              -6.339e+01  6.756e+01  -0.938    0.348
## children          6.957e+02  6.341e+02   1.097    0.273
## smoker1          -1.983e+04  1.861e+03 -10.651  < 2e-16 ***
## age:bmi           1.912e+00  1.561e+00   1.225    0.221
## age:children      1.201e+00  8.527e+00   0.141    0.888
## age:smoker1      -9.141e-01  2.384e+01  -0.038    0.969
## bmi:children     -5.334e+00  1.863e+01  -0.286    0.775
## bmi:smoker1       1.437e+03  5.317e+01  27.029  < 2e-16 ***
## children:smoker1 -3.858e+02  2.841e+02  -1.358    0.175
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4874 on 1327 degrees of freedom
## Multiple R-squared:  0.8392, Adjusted R-squared:  0.838
## F-statistic: 692.8 on 10 and 1327 DF,  p-value: < 2.2e-16
```

As the above output shows not all of the added interactions have to be considered in the model. Upfront we also tried to generate a linear model including all possible interactions. Since the output was not satisfying we skiped this analysis at this point. Therfore the working assumption in this step was quite easy by editing only possible and simple interactions. Based on the results we will now drop the unnecessary interactions.

```r
drop1(lm.insurance.2, test="F")
```

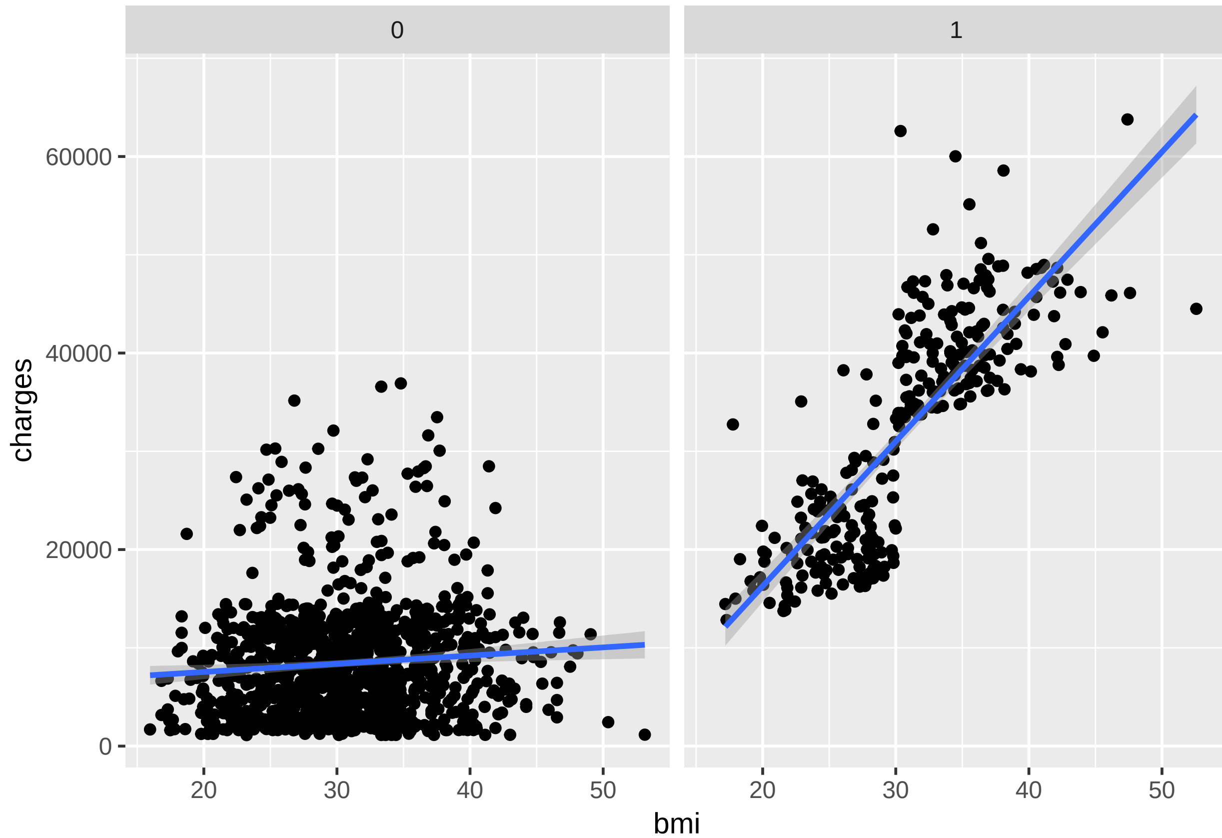
```
## Single term deletions
##
## Model:
## charges ~ age + bmi + children + smoker + age:bmi + age:children +
##     age:smoker + bmi:children + bmi:smoker + children:smoker
##                 Df  Sum of Sq        RSS   AIC  F value Pr(>F)
## <none>                        3.1519e+10 22735
## age:bmi          1 3.5624e+07 3.1555e+10 22734   1.4998 0.2209
## age:children     1 4.7151e+05 3.1520e+10 22733   0.0199 0.8880
## age:smoker       1 3.4909e+04 3.1519e+10 22733   0.0015 0.9694
## bmi:children     1 1.9467e+06 3.1521e+10 22733   0.0820 0.7747
## bmi:smoker       1 1.7353e+10 4.8872e+10 23319 730.5705 <2e-16 ***
## children:smoker  1 4.3796e+07 3.1563e+10 22734   1.8439 0.1747
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
lm.insurance.3 <- update(lm.insurance.2, .~. -age:bmi - age:children - age:smoker - bmi:children -child
summary(lm.insurance.3)
```

```
##
## Call:
## lm(formula = charges ~ age + bmi + children + smoker + bmi:smoker,
##     data = insurance)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14598.6  -1924.4  -1321.4   -465.6  29892.4
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2729.002    831.270  -3.283  0.00105 **
## age            264.948      9.553  27.735  < 2e-16 ***
## bmi              5.656     24.873   0.227  0.82014
## children       508.924    110.615   4.601 4.61e-06 ***
## smoker1     -20194.709   1654.505 -12.206  < 2e-16 ***
## bmi:smoker1   1433.788     52.823  27.143  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4871 on 1332 degrees of freedom
## Multiple R-squared:  0.8388, Adjusted R-squared:  0.8382
## F-statistic:  1387 on 5 and 1332 DF,  p-value: < 2.2e-16
```

After considering also the interaction between bmi and smoker, which seems to be significant with a small p-value of 2e-16. The variable bmi itself has now a p-value of 0.82014. Therfore lets have a look on the relationship between bmi and smoker to have a better understanding:

```r
ggplot(data=insurance,
       mapping = aes(y = charges, x= bmi))+ geom_point()+ geom_smooth(method = 'lm') +facet_grid(. ~ smo
```



As expected there seems to be a clear relationship between charges and bmi, considering if a person is smoking or not. Since this relationship makes even sense from the domain perspective, we will definitly keep this in our model.

3.5.4 Measure of Fit

In this section we were able to fit several models, having different levels of complexity. For the following analysis we have choosen our four models to measure the individual fit. Moreover we will compare them among each other to find the most appropriate model for the given insurance dataset.

```
formula(lm.insurance.all)
```

```
## charges ~ age + sex + bmi + children + smoker + region
```

```
summary(lm.insurance.all)$r.squared
```

```
## [1] 0.750913
```

```
summary(lm.insurance.all)$adj.r.squared
```

```
## [1] 0.7494136
```

```
formula(lm.insurance.1)
```

```
## charges ~ age + bmi + children + smoker
```

```
summary(lm.insurance.1)$r.squared
```

```
## [1] 0.7496945
```

```r
summary(lm.insurance.1)$adj.r.squared
```

```
## [1] 0.7489434
```

```r
formula(lm.insurance.2)
```

```
## charges ~ age + bmi + children + smoker + age:bmi + age:children +
##     age:smoker + bmi:children + bmi:smoker + children:smoker
```

```r
summary(lm.insurance.2)$r.squared
```

```
## [1] 0.8392491
```

```r
summary(lm.insurance.2)$adj.r.squared
```

```
## [1] 0.8380378
```

```r
formula(lm.insurance.3)
```

```
## charges ~ age + bmi + children + smoker + bmi:smoker
```

```r
summary(lm.insurance.3)$r.squared
```

```
## [1] 0.8388379
```

```r
summary(lm.insurance.3)$adj.r.squared
```

```
## [1] 0.8382329
```

For the comparison we used R-Squared and the adjusted R-Squared to measure the performance of our models. Since the adjusted R-squared can provide a more precise view of that correlation by also taking into account how many independent variables are added to our particular models against we will base our conclusion on this parameter.

Therefore we are happy to state that the latest model number three is able to explain the charges with a percentage of 83.82329 % based on the independent variables.

Comparing the latest model with the first one there is an increase of 8.88% in the adjusted R-squared. Even if the latest model is performing the best compared to the others, it is always a trade off between the gain in the fit and the corresponding effort.

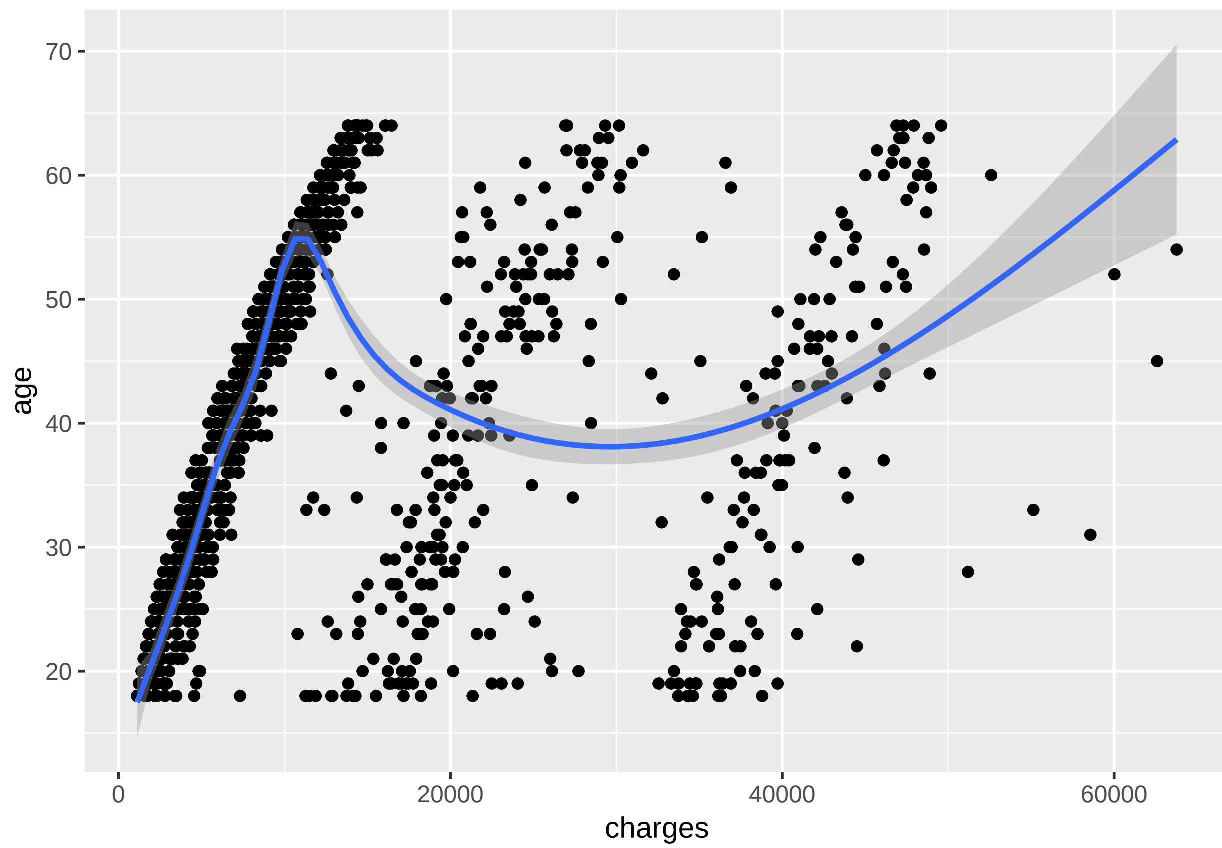
4 Linear models (GAM & Polynomial) -> Yvonne

5 Polynomial

As we can see from the following plot, the models are not always linear.

```r
gg.age.charges <- ggplot(data = insurance,
                  mapping = aes (y = age,
                                 x = charges)) +
  geom_point()
gg.age.charges + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Clearly visible is the increase in costs up to about 18'000 with increasing age. The dependence of amount and costs above 10,000 is more difficult to predict due to the non-linearity.

We add linearity to the model by adding a square term to charges.

```r
# Model with a linear effect for charges
lm.insurance.1 <- lm(age ~ sex + bmi + children + charges, data = insurance)

#Model with a quadratic effect for charges
lm.insurance.2 <- update(lm.insurance.1, . ~ . + I(charges^2))
```

The F-test is used to test the qudrate term.

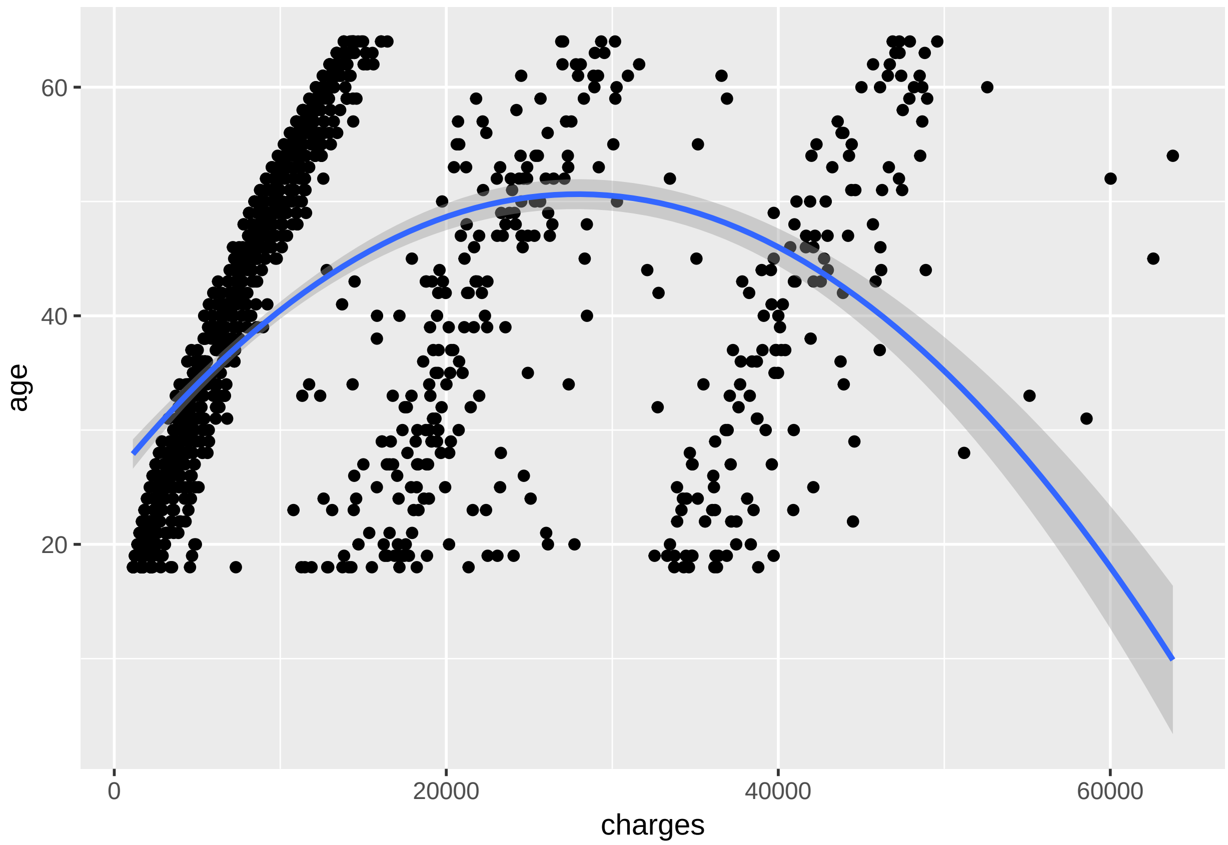
```r
anova(lm.insurance.1, lm.insurance.2)
```

```
## Analysis of Variance Table
##
## Model 1: age ~ sex + bmi + children + charges
## Model 2: age ~ sex + bmi + children + charges + I(charges^2)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1   1333 239086
## 2   1332 198860  1     40227 269.45 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-test shows a strong indication charges requires a quadratic term.

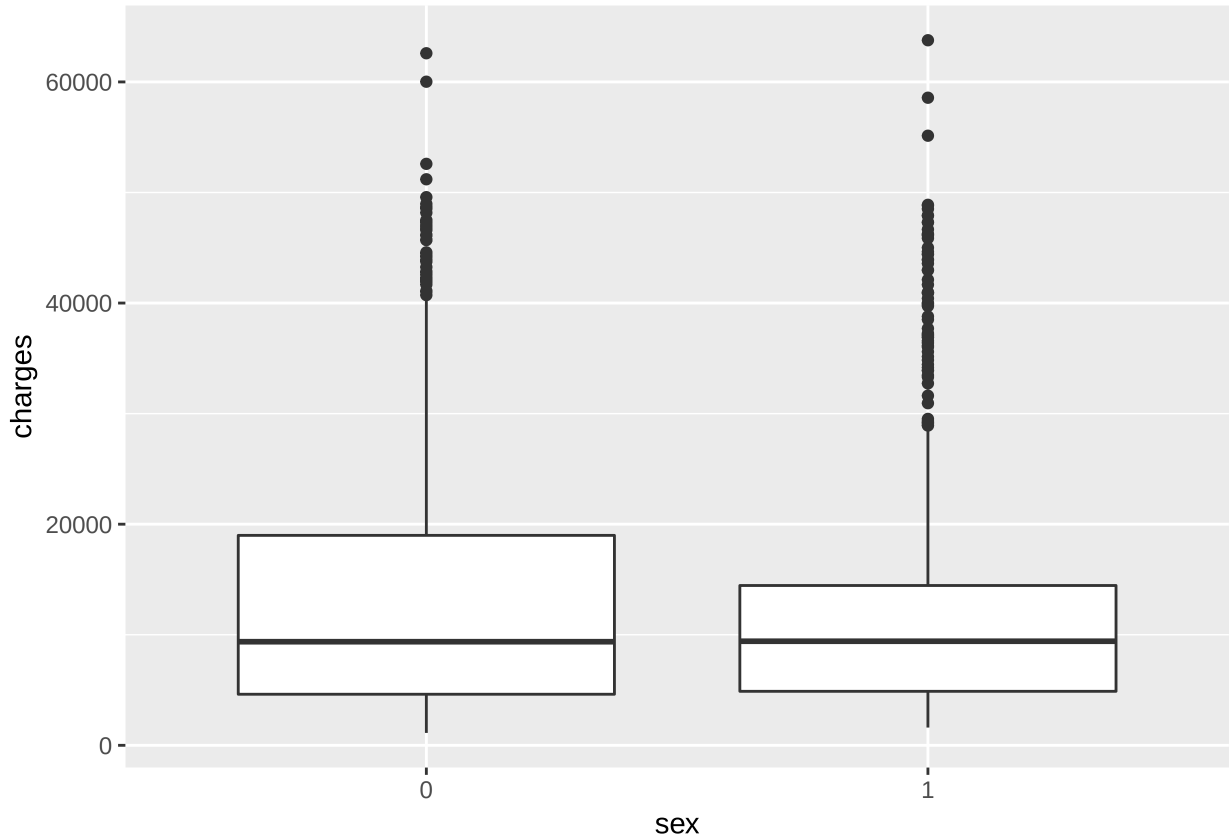
The graphical result is shown below.

```r
gg.age.charges +
  geom_smooth(method = "lm",
              formula = y ~ poly(x, degree = 2))
```



What is the result if you look at the men and women separately.

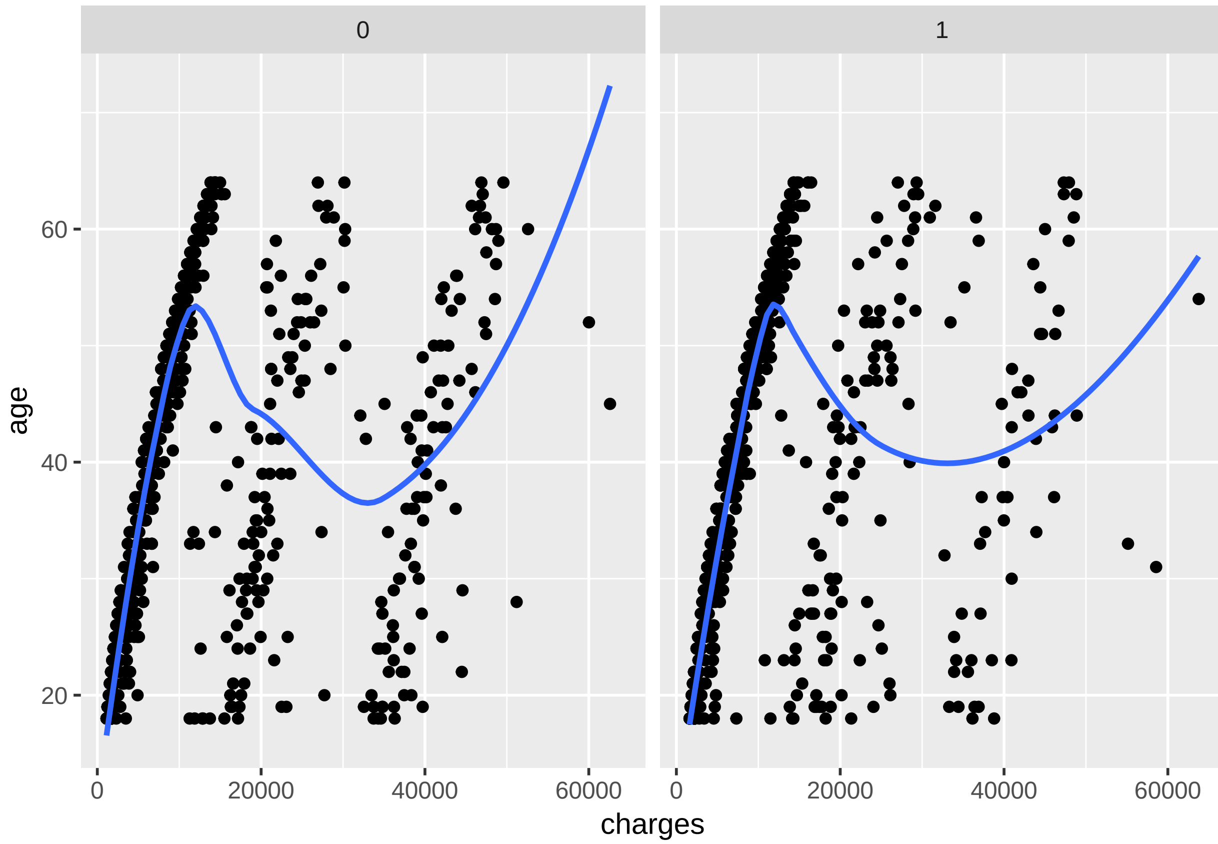
```r
ggplot(data = insurance,
         mapping = aes(y = charges,
                       x = sex)) +
    geom_boxplot()
```



This graph shows that there is not really a difference in the median between women and men. The only difference is in the middle 50% range where the men have a larger range.

```
ggplot(data = insurance,
       mapping = aes(y = age,
                     x = charges)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  facet_wrap(. ~ sex)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



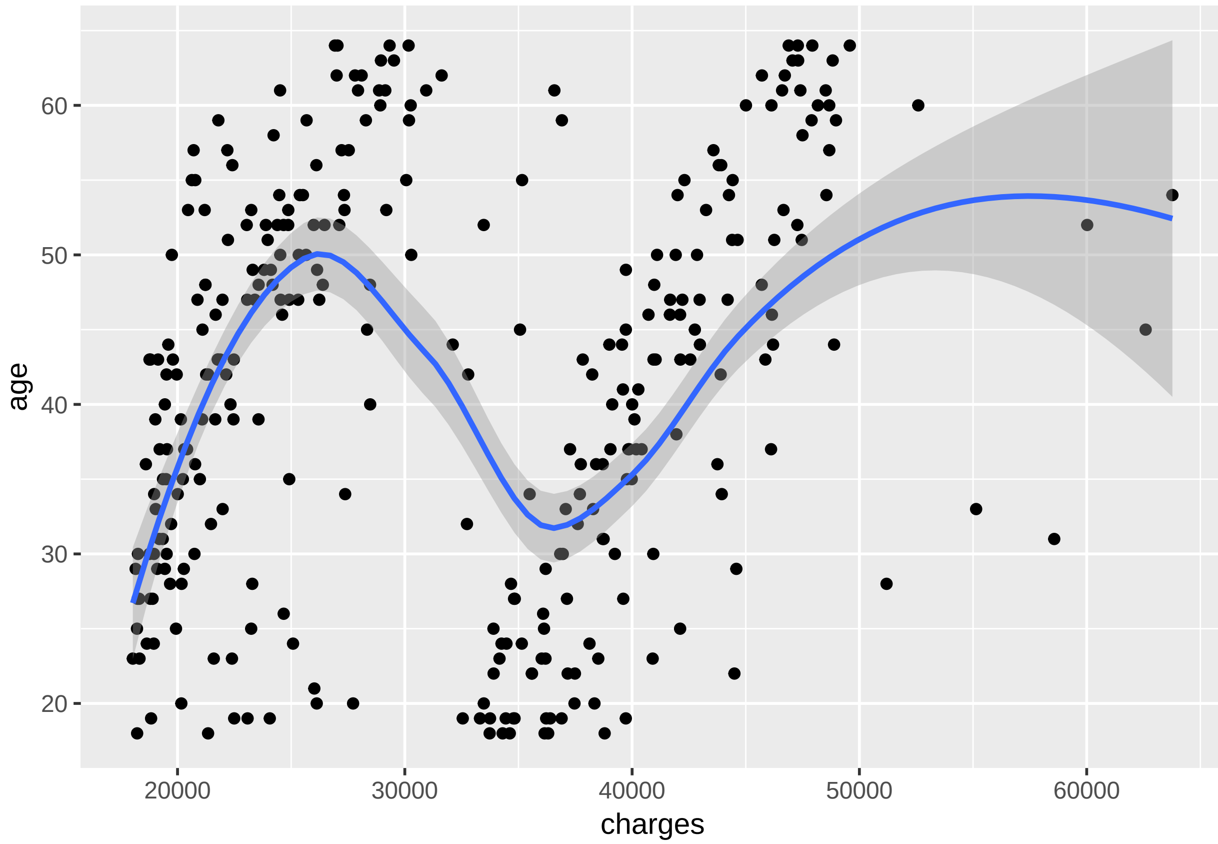
The rough course of costs as a function of age is comparable for men and women. A closer look at the result reveals slight differences. However, this is not a reason to differentiate the models according to gender.

The first cost increase looks more like a linear model. What does the result look like if the costs are brought above 18'000? This will be examined in the following considerations.

```
insurance.part2 <- filter(insurance, charges > 18000 )
```

```
gg.age.charges.part2 <- ggplot(data = insurance.part2,
                               mapping = aes (y = age,
                                              x = charges)) +
  geom_point()
gg.age.charges.part2 + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
lm.insurance.11 <- lm(age ~ sex + bmi + children +charges, data = insurance.part2)
lm.insurance.12 <- update(lm.insurance.11, . ~ . + I(charges^2))

anova(lm.insurance.11, lm.insurance.12)
```

```
## Analysis of Variance Table
##
## Model 1: age ~ sex + bmi + children + charges
## Model 2: age ~ sex + bmi + children + charges + I(charges^2)
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1    308 53277
## 2    307 52872  1    405.12 2.3523 0.1261
```

The result shows that there is no strong evidence that charges needs a quadratic term. The model appear to be linear.

6 Generalised Additive Models

The previous data modelling is now performed with GAM.

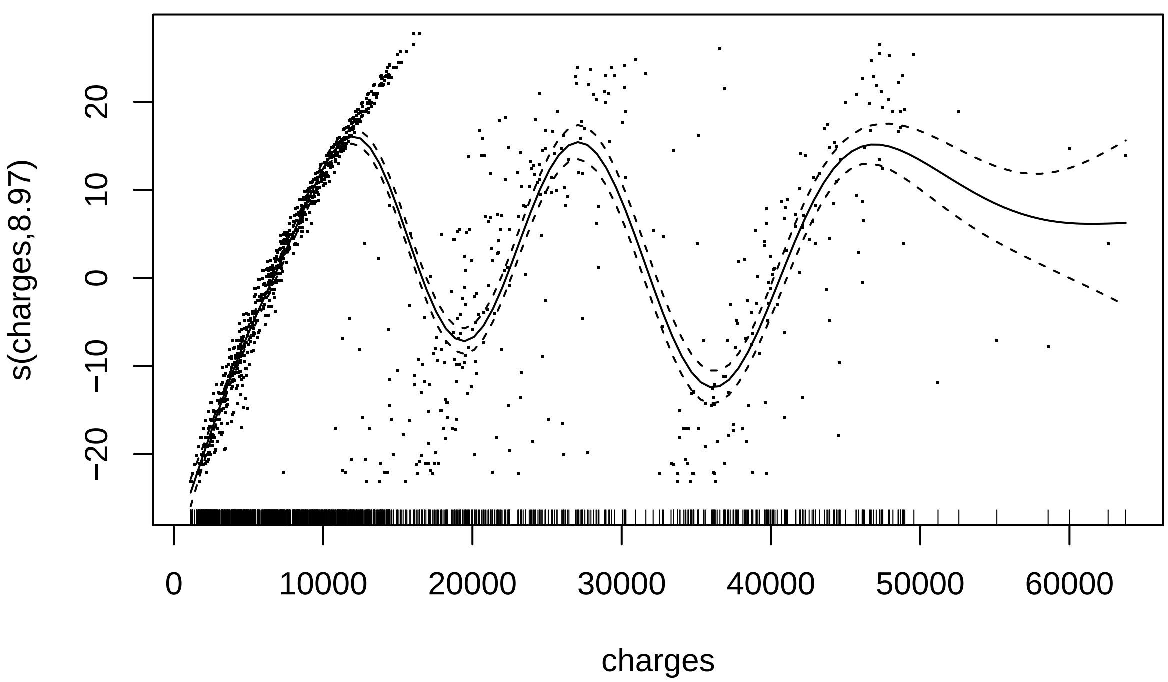
```
gam.insurance.1 <- gam(age ~ sex + s(charges) + children,
                       data = insurance)
summary(gam.insurance.1)
```

```
##
## Family: gaussian
## Link function: identity
```

```
##
## Formula:
## age ~ sex + s(charges) + children
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.1364     0.3598 114.330  < 2e-16 ***
## sex1         -1.0849     0.4186  -2.591  0.00966 **
## children     -1.2719     0.1836  -6.926 6.72e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df     F p-value
## s(charges) 8.974      9 363.9  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.71   Deviance explained = 71.3%
## GCV = 57.673  Scale est. = 57.157    n = 1338
```

The s(changes) value of 8,974 is high and allows us to model the existing predictor with a smooth term. The corresponding visualization is shown below.

```
plot(gam.insurance.1, residuals = TRUE, select = 1)
```

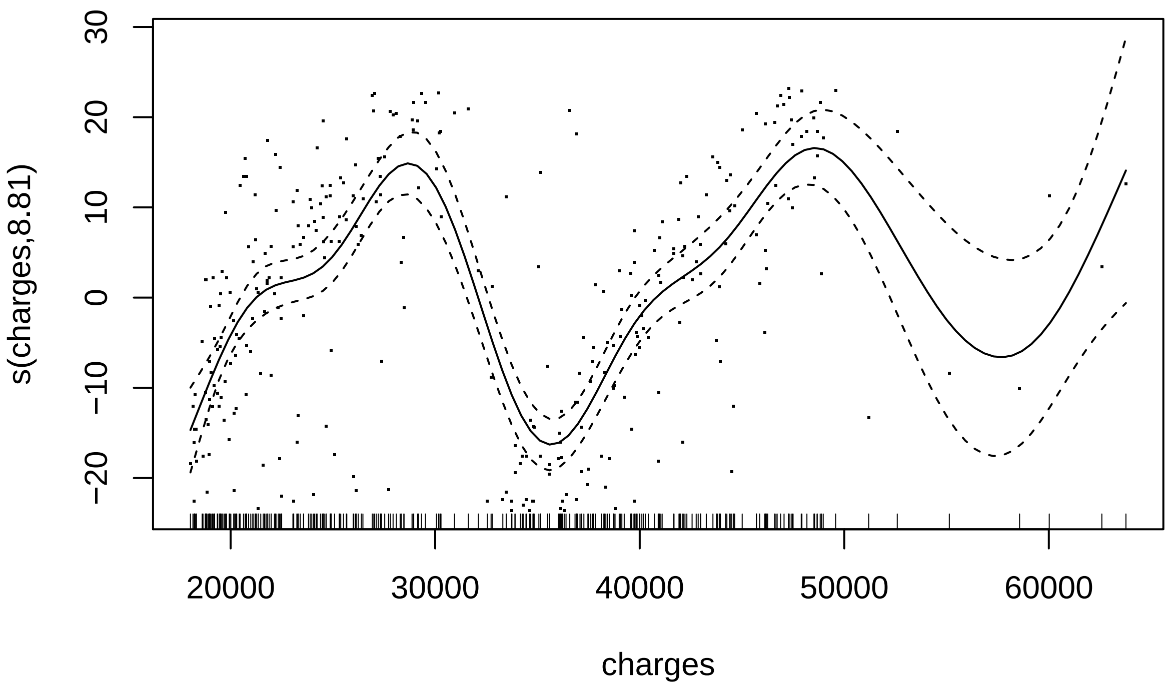


If you only consider the costs greater than 18'000 should be, the result is identical with the previous consideration. A linear model is also preferable in this case.

```r
gam.insurance.2 <- gam(age ~ sex + s(charges) + children,
                       data = insurance.part2)
summary(gam.insurance.2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## age ~ sex + s(charges) + children
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.5861     0.9591  43.360   <2e-16 ***
## sex1         -0.1992     1.1457  -0.174    0.862
## children     -0.2804     0.4936  -0.568    0.570
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df     F p-value
## s(charges) 8.815   8.99 32.15  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.474   Deviance explained = 49.2%
## GCV =  101.7  Scale est. = 97.857    n = 313
```

```r
plot(gam.insurance.2, residuals = TRUE, select = 1)
```



7 GLM and cross validation -> Carole

7.1 Generalised Linear Models for count data

7.1.1 Original data

The number of children an insured person has is analysed. We have the following data on children per person. The number of children ranges from 0 to 5 with a median of 1.

```
summary(insurance$children)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   1.095   2.000   5.000
```

7.1.2 Poisson model

To model count data (number of children) the poisson model is used. An analysis performed beforehand showed that only the variables "charges" and "smoker" have a significant impact on the number of children.

```
glm.children <- glm(children ~ smoker+charges,
                    data=insurance,
                    family = "poisson")
summary(glm.children)
```

```
##
## Call:
## glm(formula = children ~ smoker + charges, family = "poisson",
##     data = insurance)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8561  -1.4318  -0.1057   0.7768   2.9717
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.706e-02  4.213e-02  -0.880   0.3790
## smoker1     -3.239e-01  1.058e-01  -3.061   0.0022 **
## charges      1.419e-05  3.365e-06   4.217 2.48e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2001.6  on 1337  degrees of freedom
## Residual deviance: 1984.1  on 1335  degrees of freedom
## AIC: 3879.4
##
## Number of Fisher Scoring iterations: 5
```

To get the coefficients, the log transformation needs to be reversed:

```r
exp(coef(glm.children))
```

```
## (Intercept)     smoker1     charges
##   0.9636169   0.7233085   1.0000142
```

Smoker (factor): The model shows that for the factor smoker (yes/no), a smoker has on average 72% of the number of children a non-smoker has. The more common-sense interpretation might be the other way around, that people who have 1 or more children smoke less, but for the moment we have no proof of that.

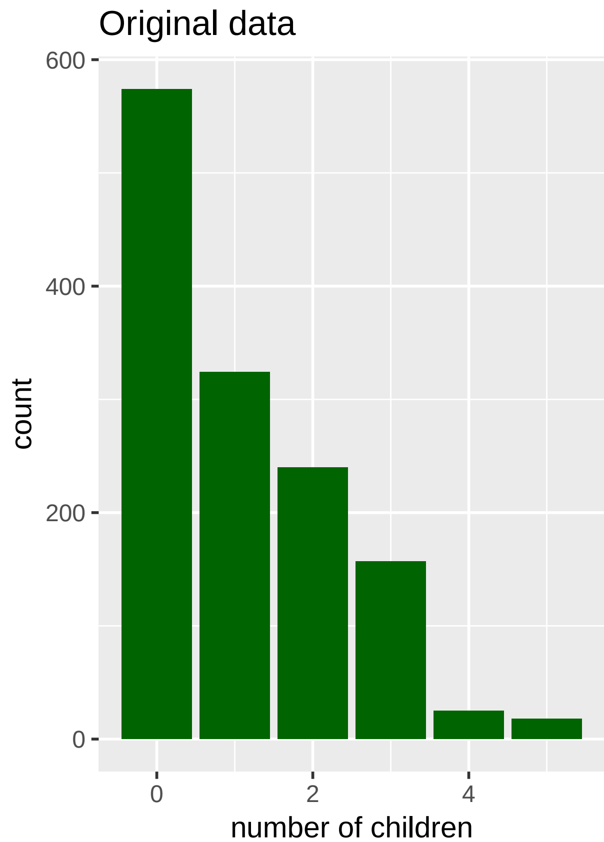
Charges: A person with higher charges will on average have more children. If charges are increased by 1000 dollars, the calculated number of children increases by 1.4%.

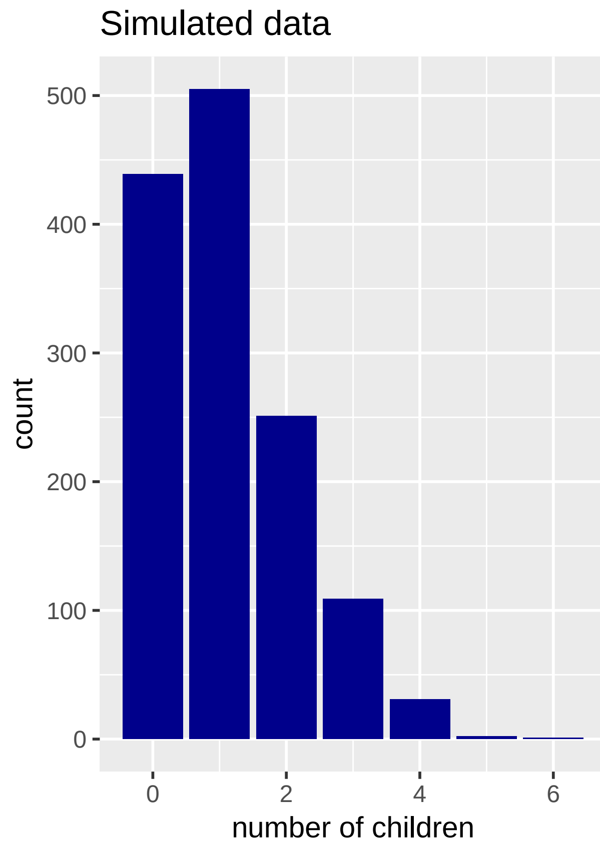
7.1.3 Simulation of data and comparison

With the calculated model, data is simulated:

```
##      sim_1
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :1.000
##  Mean   :1.102
##  3rd Qu.:2.000
##  Max.   :6.000
```

The original and the simulated data are compared visually. The number of children from the simulated data (0-6) seem to be plausible. The distribution has a strong downwards trend starting at 1 like the original data. However the model does not seem to generate enough data with 0 children.





7.2 Generalised Linear Models for binomial data

A model is fitted that predicts if a person is a smoker or not. Only the significant values age, bmi and charges are used.

```
glm.smoker.2 <- glm(smoker ~ age+bmi+charges,
                    data=insurance,
                    family = "binomial")
summary(glm.smoker.2)
```

```
##
## Call:
## glm(formula = smoker ~ age + bmi + charges, family = "binomial",
##     data = insurance)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -3.09442  -0.10998  -0.04475  -0.00970   1.53727
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.311e+00  1.029e+00   5.163 2.43e-07 ***
## age         -9.875e-02  1.300e-02  -7.597 3.02e-14 ***
## bmi         -3.481e-01  4.309e-02  -8.078 6.60e-16 ***
## charges      3.822e-04  2.917e-05  13.104  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1356.63  on 1337  degrees of freedom
## Residual deviance:  311.98  on 1334  degrees of freedom
## AIC: 319.98
##
## Number of Fisher Scoring iterations: 8
```

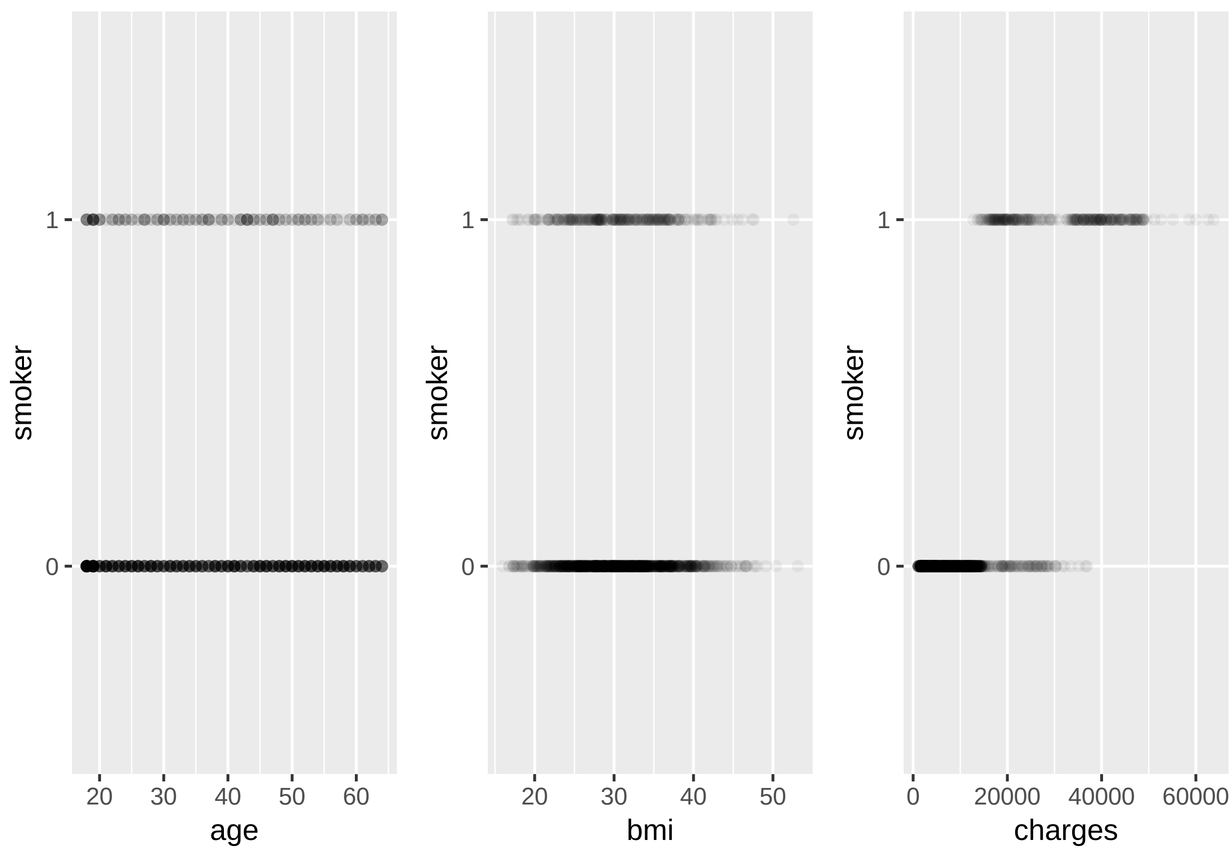
```
exp(coef(glm.smoker.2))
```

```
## (Intercept)         age         bmi     charges
## 202.5686249   0.9059677   0.7060520   1.0003823
```

Age and BMI has a negative effect on smoker. This means the higher a persons BMI and age, the lower the probability that the person is a smoker. Charges has a positive effect. This means the higher a persons charges, the higher is the possibility that the person smokes.

7.2.1 Graphical analysis

This can also be explored graphically, at least for charges it is clearly visible that smokers have higher charges.



7.2.2 Estimating the model performance

The predicted values are transformed into binary (beforehand they inicated the probability) and compared with the actual data.

```r
fitted.smoker.disc <- ifelse(fitted(glm.smoker.2) < 0.5,
                             yes = 0, no = 1)
head(fitted.smoker.disc)
```

```
## 1 2 3 4 5 6
## 1 0 0 1 0 0
```

```r
d.obs.fit.smoker <- data.frame(obs = insurance$smoker,
                               fitted = fitted.smoker.disc)
head(d.obs.fit.smoker)
```

```
##   obs fitted
## 1   1      1
## 2   0      0
## 3   0      0
## 4   0      1
## 5   0      0
## 6   0      0
```

We observe the following fit:

```
##    fit
## obs    0    1
##   0 1028   36
##   1   23  251
```

7.3 Cross Validation

Three linear models are cross validated:

```r
lm.1 <- lm(data=insurance, charges~children+smoker+bmi+age+region+sex)
lm.2 <- lm(data=insurance, charges~children+smoker)
lm.3 <- lm(data=insurance, charges~(poly(bmi, degree=3))
           +(poly(age, degree=2))+children+smoker)
```

The in sample performance, using R Squared as measure is the following:

```r
summary(lm.1)$r.squared
```

```
## [1] 0.750913
```

```r
summary(lm.2)$r.squared
```

```
## [1] 0.6236038
```

```r
summary(lm.3)$r.squared
```

```
## [1] 0.754424
```

The out of sample performance is computed by using 50:50 training and test data and repeating the process 100 times.

```r
set.seed(5)
r.squared.lm.1 <- c()
r.squared.lm.2 <- c()
r.squared.lm.3 <- c()
for(i in 1:100){

  # prepare data
```

```r
  train.YES <- sample(x=c(TRUE,FALSE),
                      size=nrow(insurance),
                      replace = TRUE)

  table(train.YES)

  insurance.train <- insurance[train.YES, ]
  insurance.test <- insurance[!train.YES, ]

  # fit model with train data

  lm.1.train <- lm(formula = formula(lm.1),
                   data = insurance.train)

  lm.2.train <- lm(formula = formula(lm.2),
                   data = insurance.train)

  lm.3.train <- lm(formula = formula(lm.3),
                   data = insurance.train)

  # make prediction on test data
  lm.1.predict <- predict(lm.1.train,
                          newdata = insurance.test)

  lm.2.predict <- predict(lm.2.train,
                          newdata = insurance.test)

  lm.3.predict <- predict(lm.3.train,
                          newdata = insurance.test)

  # compute r.squared and save in list

  r.squared.lm.1[i] <- cor(lm.1.predict,
                           insurance.test$charges)^2

  r.squared.lm.2[i] <- cor(lm.2.predict,
                           insurance.test$charges)^2

  r.squared.lm.3[i] <- cor(lm.3.predict,
                           insurance.test$charges)^2
}
```

The out of sample performance, using R Squared as measure is the following:

```r
#lm.1
mean(r.squared.lm.1)
```

```
## [1] 0.7474233
```

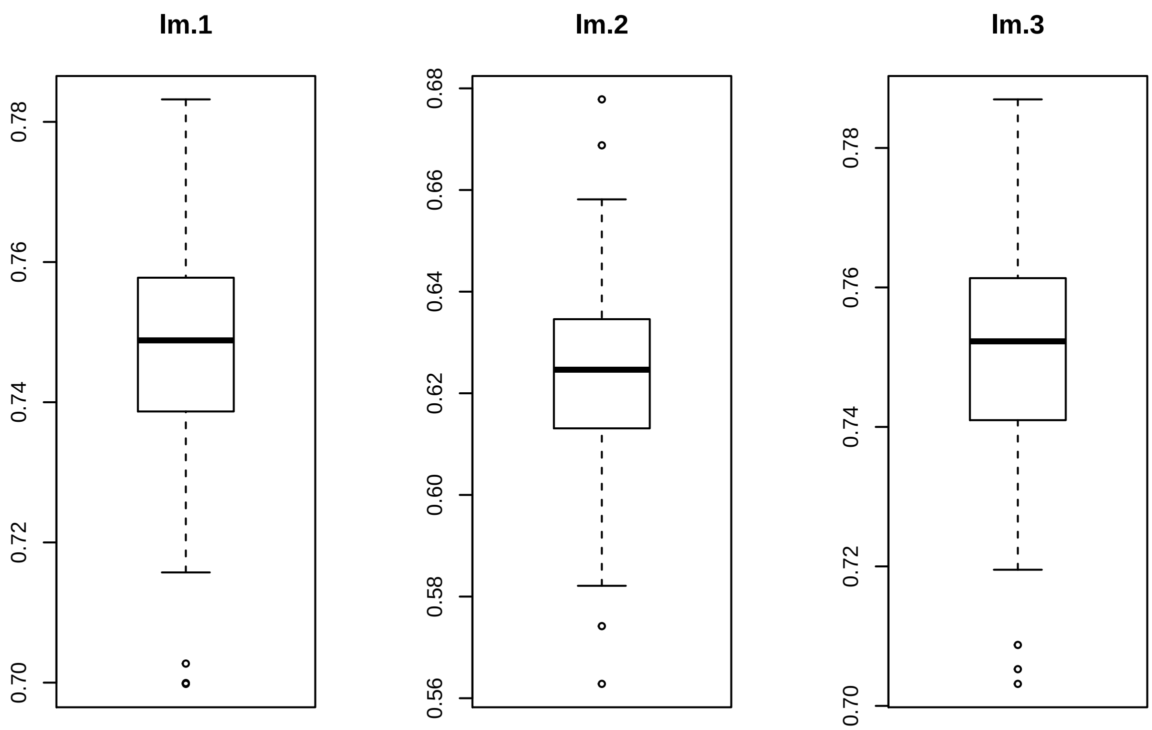
```r
#lm.2
mean(r.squared.lm.2)
```

```
## [1] 0.6235398
```

```r
#lm.3
mean(r.squared.lm.3)
```

```
## [1] 0.7505379
```

```
par(mfrow=c(1,3))
boxplot(r.squared.lm.1, main="lm.1")
boxplot(r.squared.lm.2, main="lm.2")
boxplot(r.squared.lm.3, main="lm.3")
```



It can be observed that lm.3 performs slightly better but it is also the most complicated. lm.1 might be the better model, the performance is just slightly lower and it is simpler.

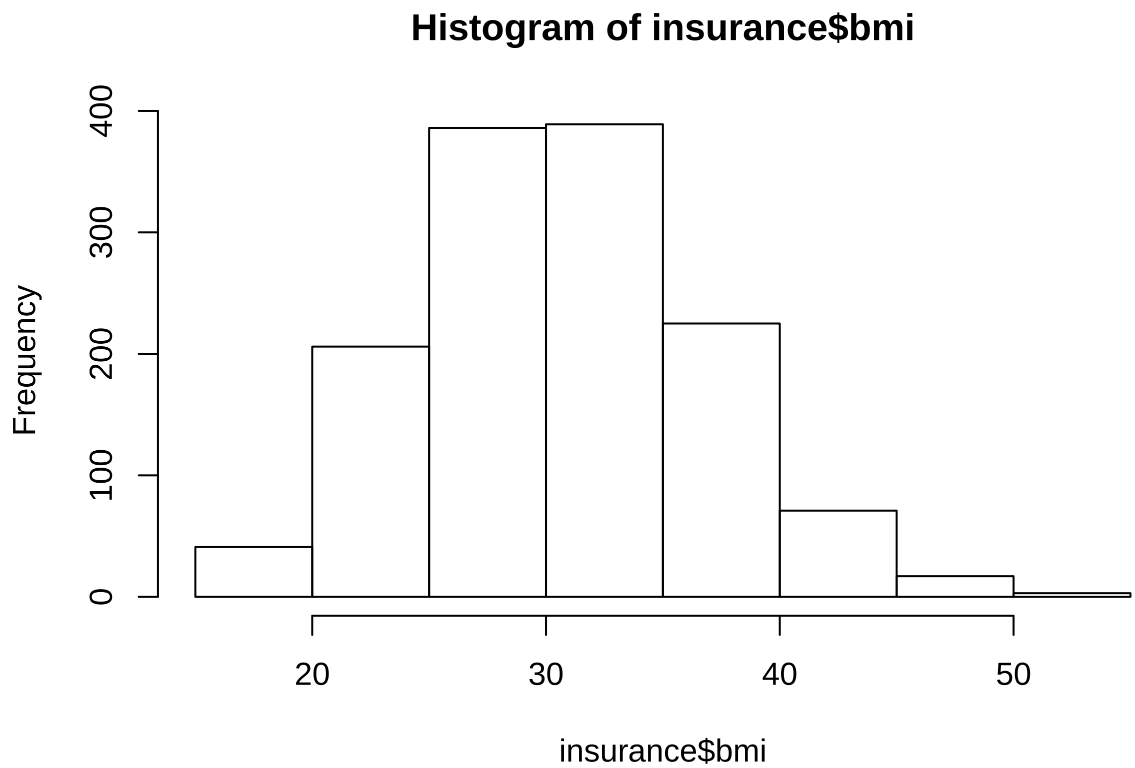
8 Decision Trees -> Adriana

8.1 Regression trees

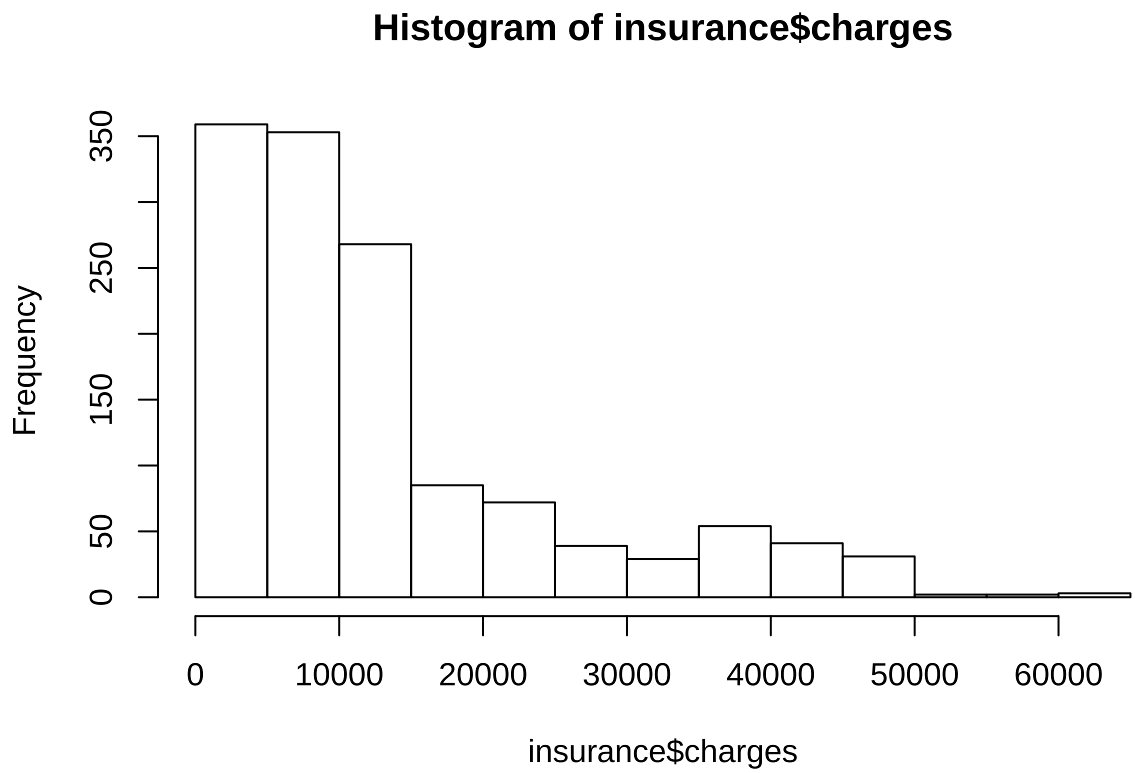
8.1.1 Inspectig the Data

To start of with our regression trees, we throw a glance at the continuous variables:

```
hist(insurance$bmi)
```




```
hist(insurance$charges)
```



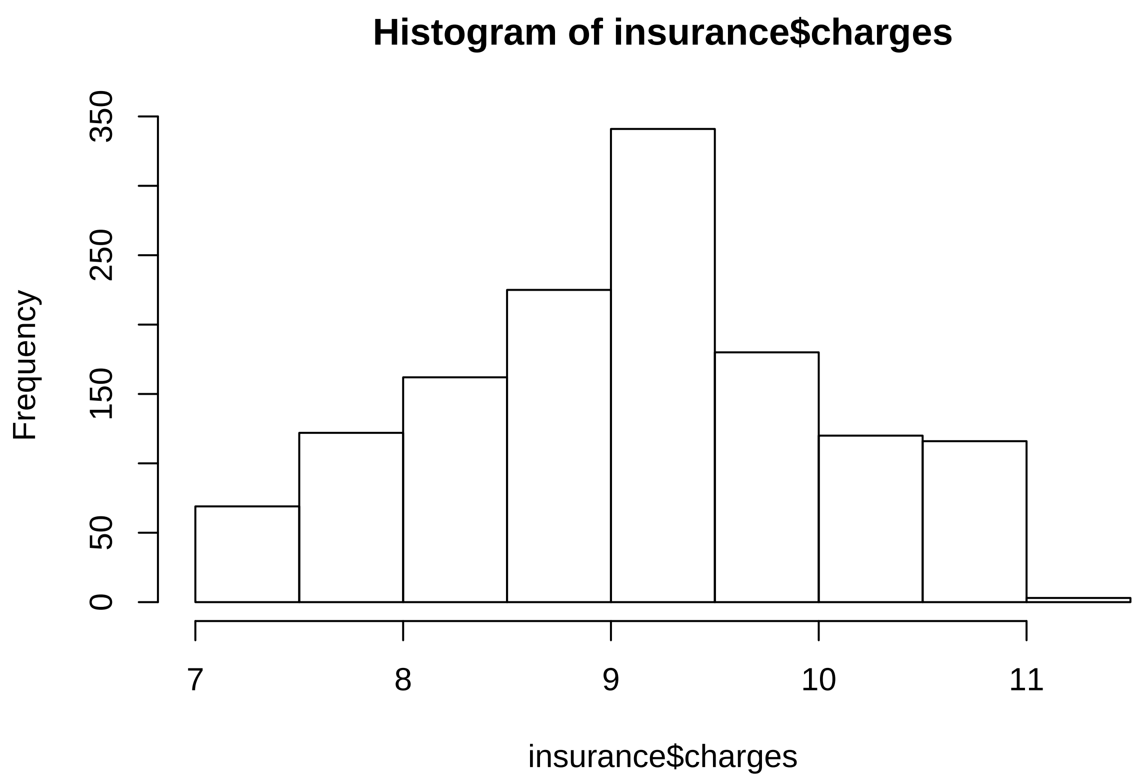
One can see that BMI seems to be normally distributed whereas charges follows the pattern of a 1/x-function.

We thus normalize "charges" by logarithming it. (Attention: Please do only execute the code block (convertion to log) below once as executing it several times will apply the logarithmic function several times)

```
insurance$charges <- log(insurance$charges)
```

Please note that hence, all charges values for this chaptre will strictly be indicated in log-value and not the true one.

```
hist(insurance$charges)
```

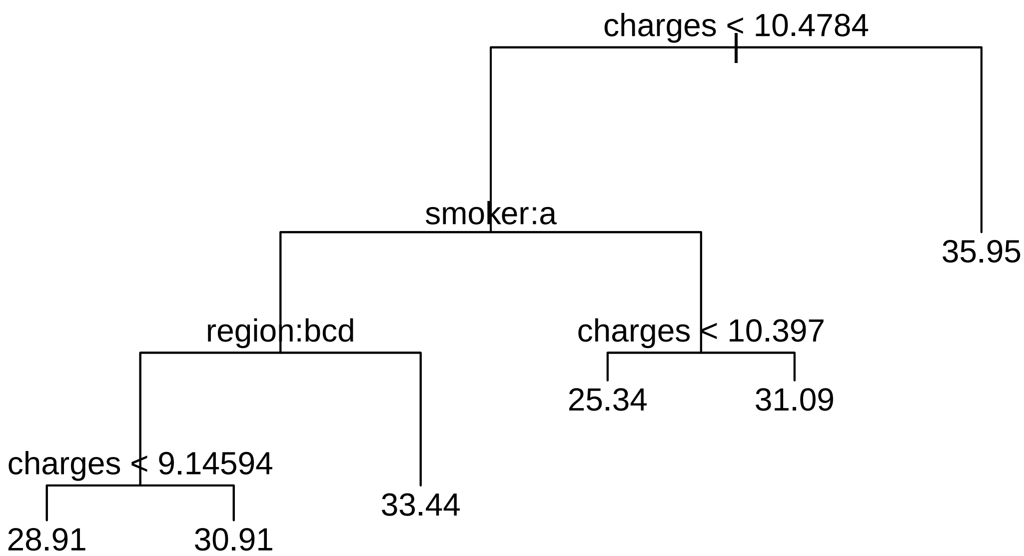


Applying a log-transformation solves this issue.

To begin with, we make the rookie mistake to use all the data, which we will then later change by appying a train-test approach. This is done to empasize the importance of not using all the data to train one's model by showing the differences in methods.

8.1.2 Regression tree - BMI

```
set.seed(99999)
tree.regression.bmi <- tree(bmi~., data=insurance)
plot(tree.regression.bmi)
text(tree.regression.bmi)
```



We here receive a tree with six terminal nodes. Thanks to the tree-function, the tree is already created by recursive binary splitting. As the variables (internal nodes) determining the BMI, “charges” have been applied as well as if one smokes and from which region one comes. This can also be seen in below summary:

```
summary(tree.regression.bmi)
```

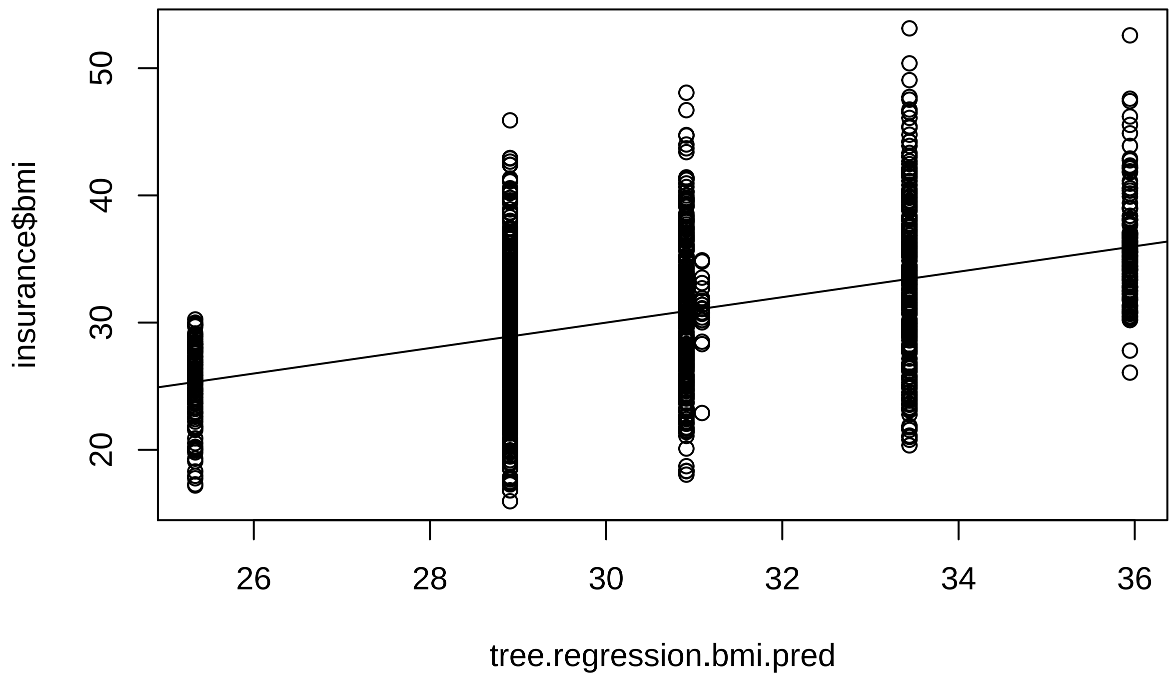
```
##
## Regression tree:
## tree(formula = bmi ~ ., data = insurance)
## Variables actually used in tree construction:
## [1] "charges" "smoker"  "region"
## Number of terminal nodes:  6
## Residual mean deviance:  29.23 = 38930 / 1332
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -13.09000  -3.62700  -0.07606   0.00000   3.26700  19.69000
```

Our residual mean deviance, aka R-squared-error ist at 29.23.

8.1.2.1 Prediction

We here establish a prediction for our BMI tree and also compare the predictions with the true value of the BMI:

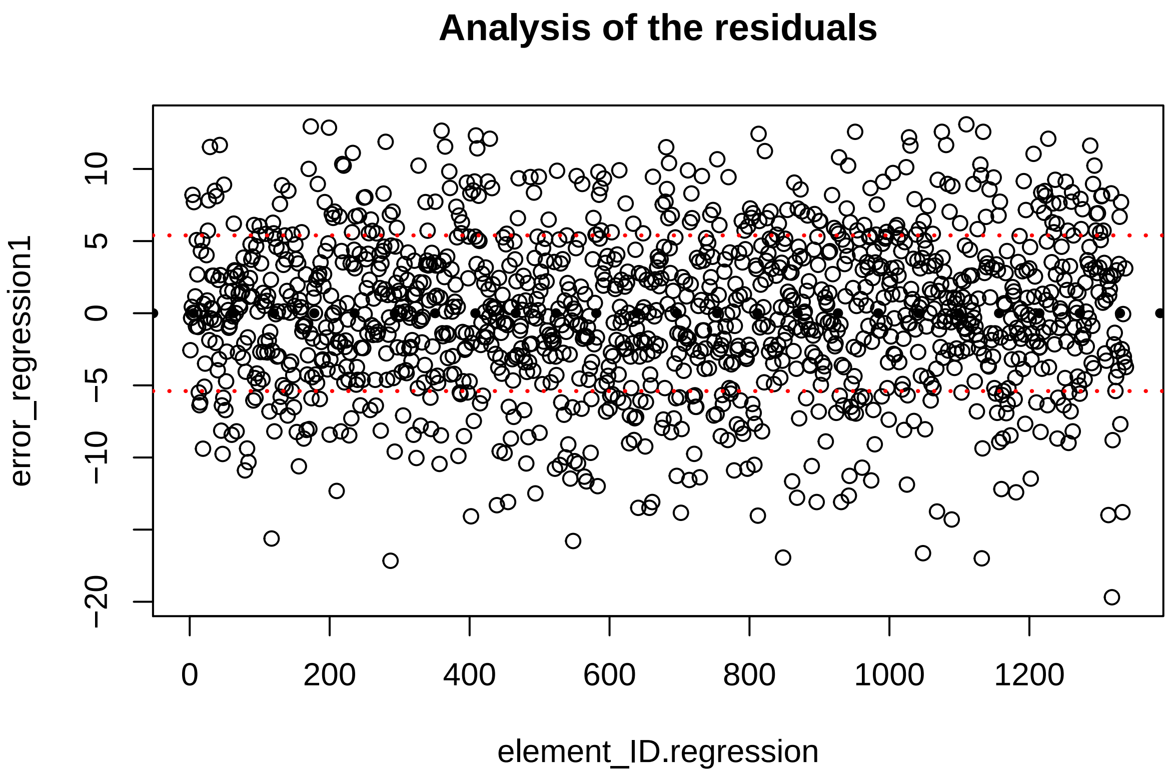
```
tree.regression.bmi.pred <- predict(tree.regression.bmi, insurance, type="vector")
plot(tree.regression.bmi.pred, insurance$bmi)
abline(0 ,1)
```



8.1.2.2 Comparison of Prediction <-> True Values

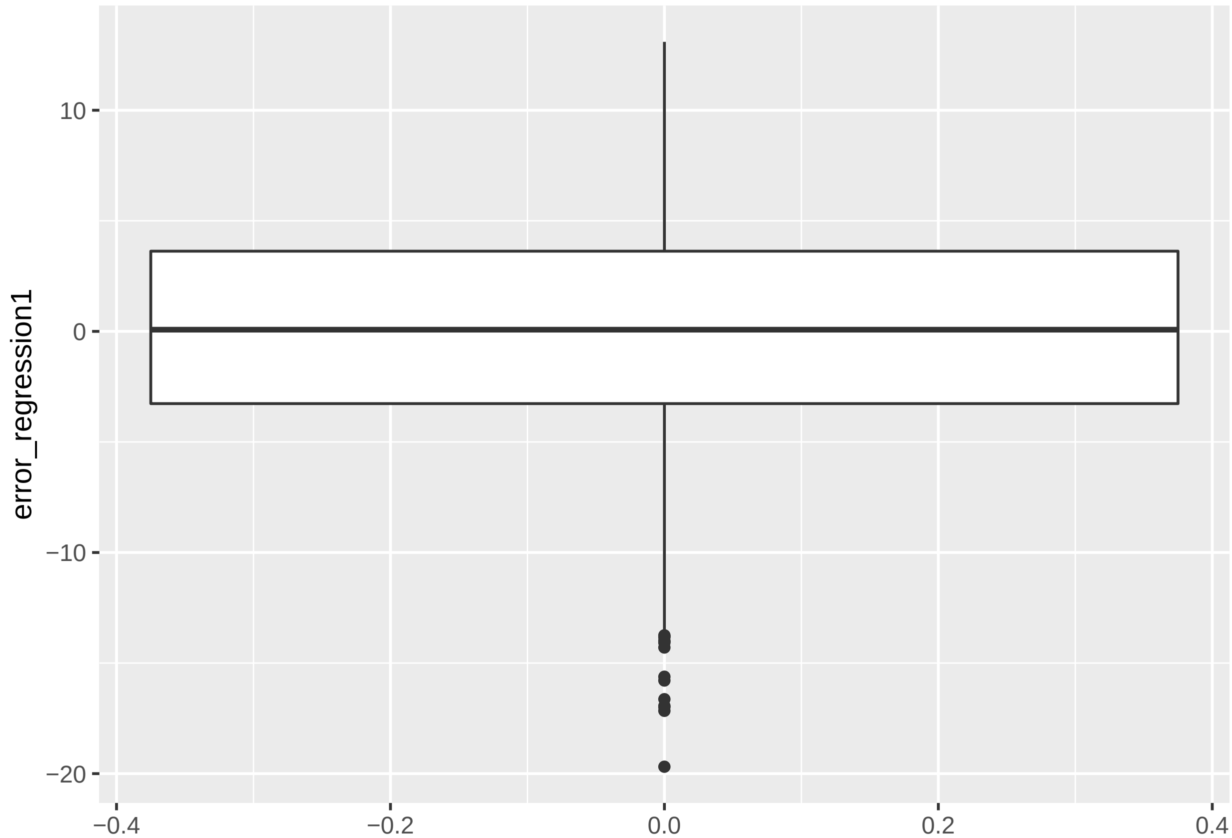
Further, we examine the residuals:

```
error_regression1 <- tree.regression.bmi.pred-insurance$bmi
element_ID.regression <- 1:length(error_regression1)
plot(element_ID.regression, error_regression1)
title(main="Analysis of the residuals")
abline(0 ,0, lwd=5,lty="dotted")
abline(sd(error_regression1) ,0, lwd=2, col="red", lty="dotted")
abline(-sd(error_regression1) ,0, lwd=2, col="red", lty="dotted")
```



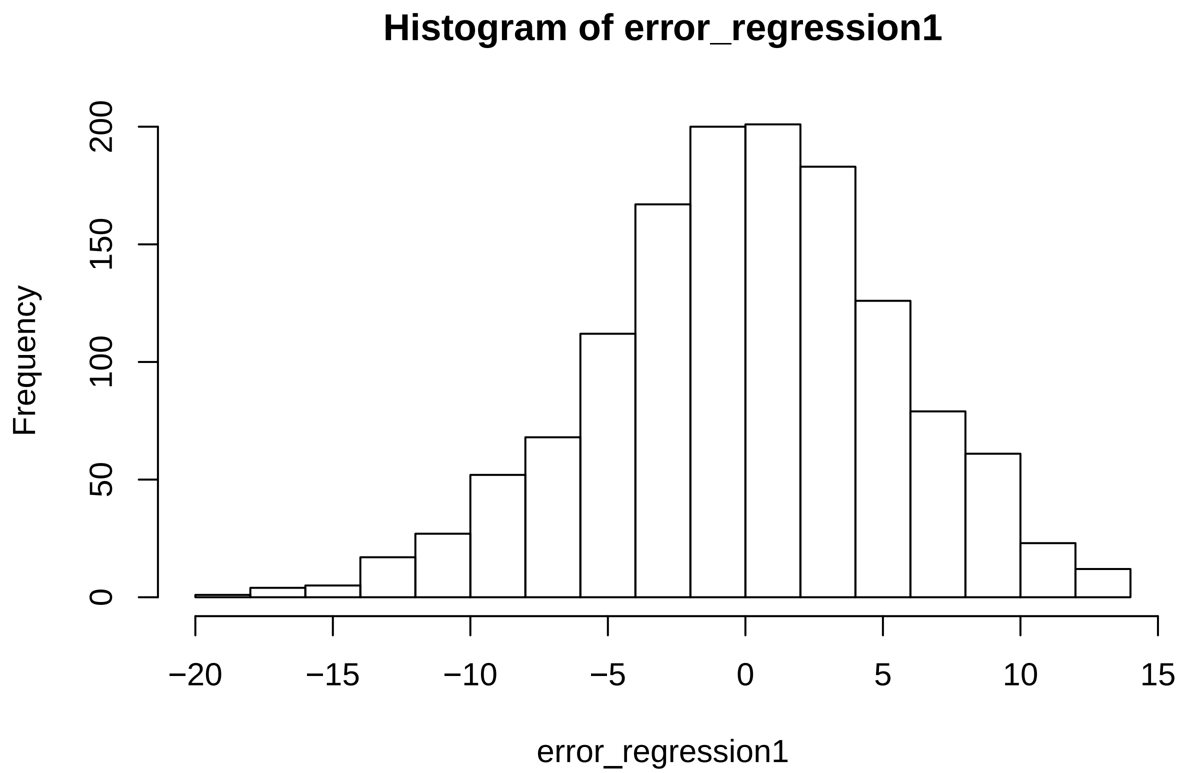
As can be seen, the residual analysis clearly supports the data to be normally distributed. So it is that most of the residuals are distributed fairly symmetrically around 0. A large portion of the residuals is furthermore within the margins of +- 1 standard deviation.

```
error_regression1_dataframe <- tibble(element_ID.regression, error_regression1)
ggplot(data=error_regression1_dataframe) + geom_boxplot(aes(y=error_regression1))
```



A boxplot further demonstrates the rather normally distributed data. It is apparent for there to be more outliers in the negative y-axis area than the positive y-axis area.

```r
hist(error_regression1)
```




```
RSS_bmi <- sum((insurance[3]-tree.regression.bmi.pred)^2)
MSE_bmi <- RSS_bmi/length(tree.regression.bmi.pred)
deviation_bmi <- sqrt(MSE_bmi)
cat(RSS_bmi)
```

```
## 38928.55
```

```
cat("\n")
```

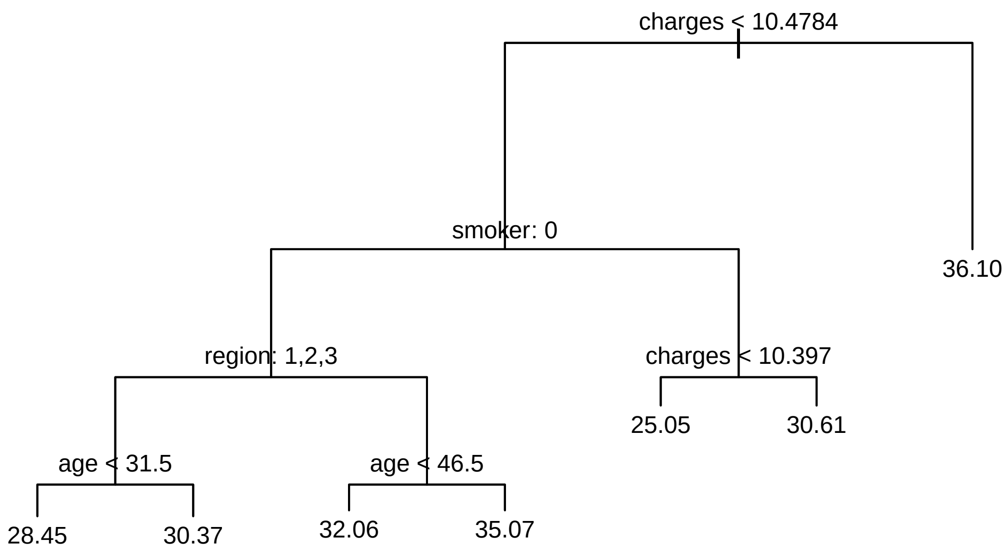
```
cat(deviation_bmi)
```

```
## 5.39394
```

So the average estimation of a BMI is about 5.39 BMI points away from the true value (median).

We now switch to the train-test approach.

```
ratio <- 0.7
total <- nrow(insurance)
train <- sample(1:total, as.integer(total * ratio))
tree.regression.bmi2 <- tree(bmi~., insurance, subset=train)
plot(tree.regression.bmi2)
text(tree.regression.bmi2, pretty=12, cex=0.75)
```



By working with the test data, we already can see at this point a change, meaning the overfit of working with our entire dataset. Were there previously six end nodes, we now have seven end notes, with variables used "charges", "smoker", "region" and "age", as also can be seen here:

```
summary(tree.regression.bmi2)
```

```
##
## Regression tree:
## tree(formula = bmi ~ ., data = insurance, subset = train)
## Variables actually used in tree construction:
## [1] "charges" "smoker"  "region"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  27.84 = 25860 / 929
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -13.55000  -3.46800  -0.09678   0.00000   3.35700  16.48000
```

Our residual mean deviance has now gone from 29.23 to 27.84.

8.1.2.3 Error Analysis

```
tree.regression.bmi2.pred <- predict(tree.regression.bmi2, insurance[-train,], type="vector")
(RSS.2.train <- mean(((insurance[train,][3]-predict(tree.regression.bmi2, insurance[train,], type="vect
```

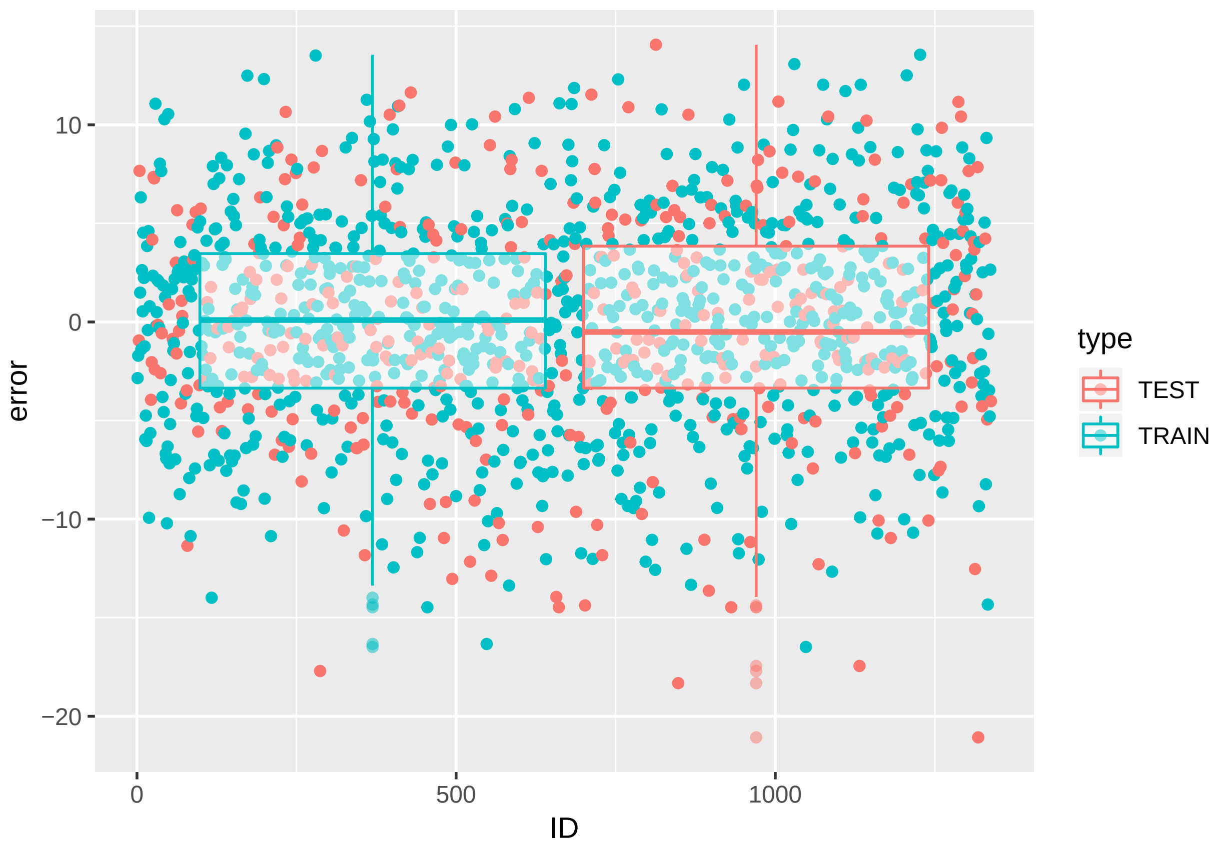
```
## [1] 27.63265
```

```
(RSS.2.test <- mean(((insurance[-train,][3]-tree.regression.bmi2.pred)^2)$bmi))
```

```
## [1] 32.85365
```

Interesstingly, we receive a RSS for the train dataset of 30.23 whereas for the test set, the RSS is 28.17.

```
errors.2.in <- predict(tree.regression.bmi2, insurance[train,], type="vector")-insurance[train,]$bmi
element.2.in <- as.integer(names(errors.2.in))
errors.2.in_dataframe <- tibble(element.2.in,errors.2.in,"TRAIN")
colnames(errors.2.in_dataframe) <- c('ID','error','type')
errors.2 <- predict(tree.regression.bmi2, insurance[-train,], type="vector")-insurance[-train,]$bmi
element.2 <- 1:length(errors.2)
element.2 <- as.integer(names(errors.2))
errors.2.out_dataframe <- tibble(element.2,errors.2,"TEST")
colnames(errors.2.out_dataframe) <- c('ID','error','type')
errors.2_dataframe <- bind_rows(errors.2.in_dataframe,errors.2.out_dataframe)
errors.2_dataframe <- arrange(errors.2_dataframe, ID)
ggplot(data = errors.2_dataframe, mapping = aes(x = ID,y = error, color = type)) +
  geom_point() + geom_boxplot(alpha = 0.5)
```



Both dataframes show the about same distribution as can be seen in the boxplot.

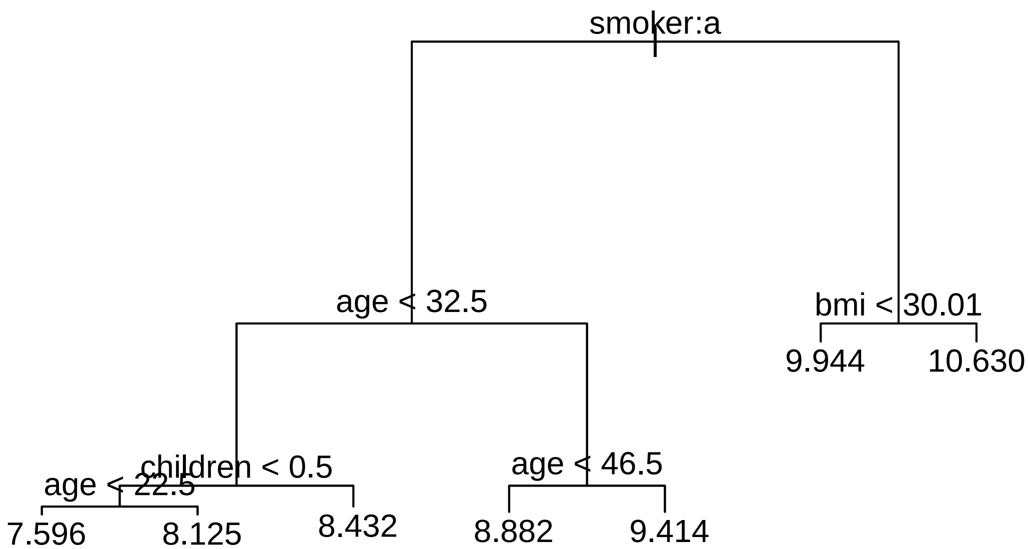
8.1.3 Regression tree - Charges

We now examine the behaviour of the variable, “charges”. First off again with the entire data values.

```
summary(insurance$charges)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.023   8.464   9.147   9.099   9.720  11.063
```

```r
tree.regression.charges <- tree(charges~., data=insurance)
plot(tree.regression.charges)
text(tree.regression.charges)
```



We receive a tree with seven end nodes and four influencing variables, as also the summary further demonstrates:

```r
summary(tree.regression.charges)
```

```
##
## Regression tree:
## tree(formula = charges ~ ., data = insurance)
## Variables actually used in tree construction:
## [1] "smoker"   "age"      "children" "bmi"
## Number of terminal nodes:  7
## Residual mean deviance:  0.1592 = 211.9 / 1331
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.98730 -0.19690 -0.04999  0.00000  0.09942  2.45100
```

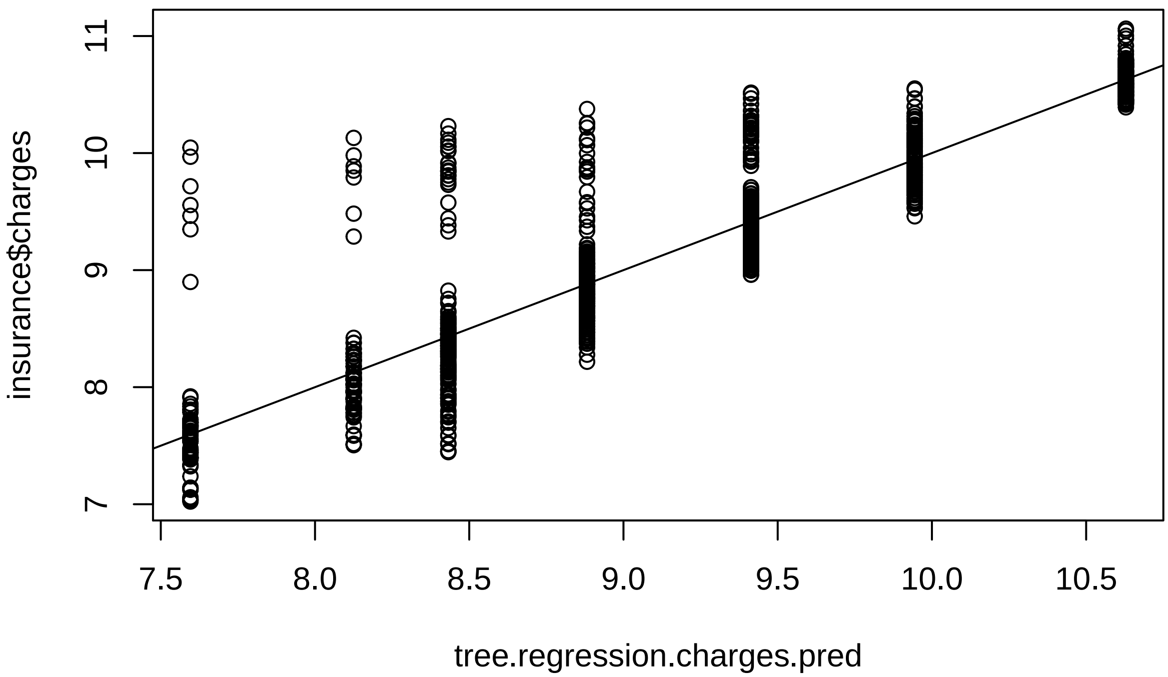
Our residual mean deviance (R-squared-error) us at 0.1592. Note that this value is also log transformed just like the other numbers regarding distribution.

8.1.3.1 Prediction

We are now again prediction the data that has already been used for training the modell.

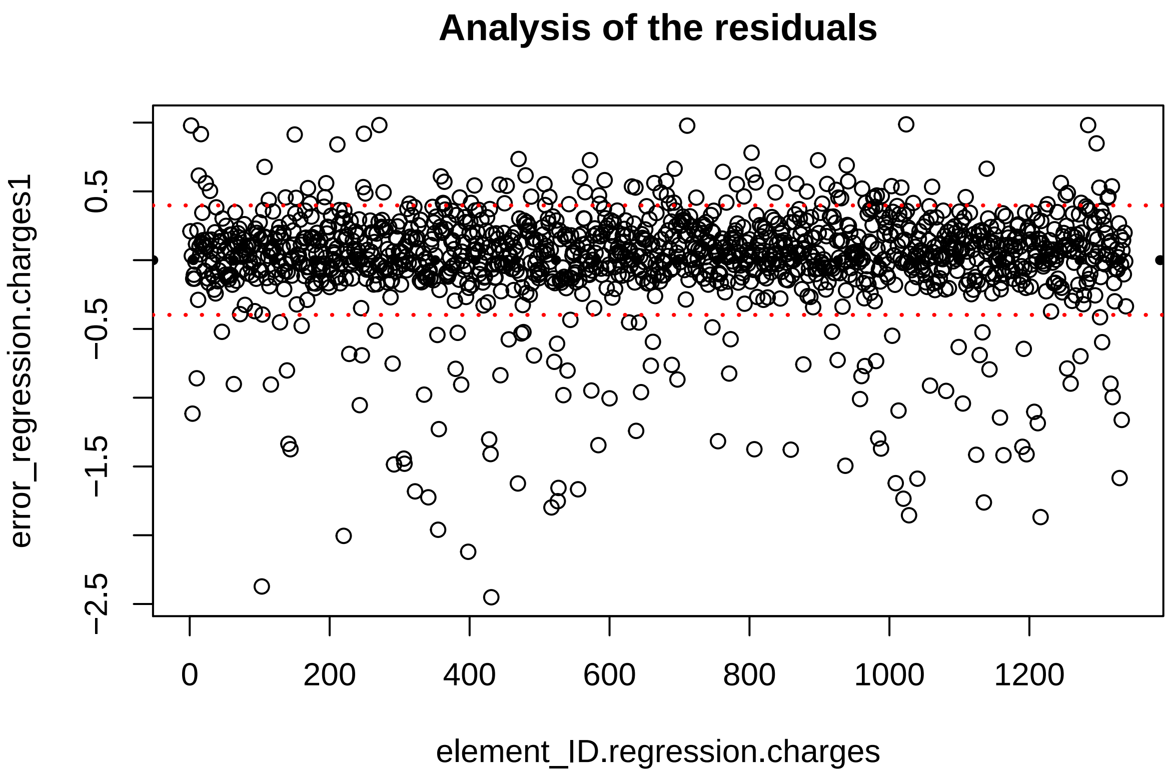
```r
tree.regression.charges.pred <- predict(tree.regression.charges, insurance, type="vector")
plot(tree.regression.charges.pred, insurance$charges)
```

```r
abline(0 ,1)
```



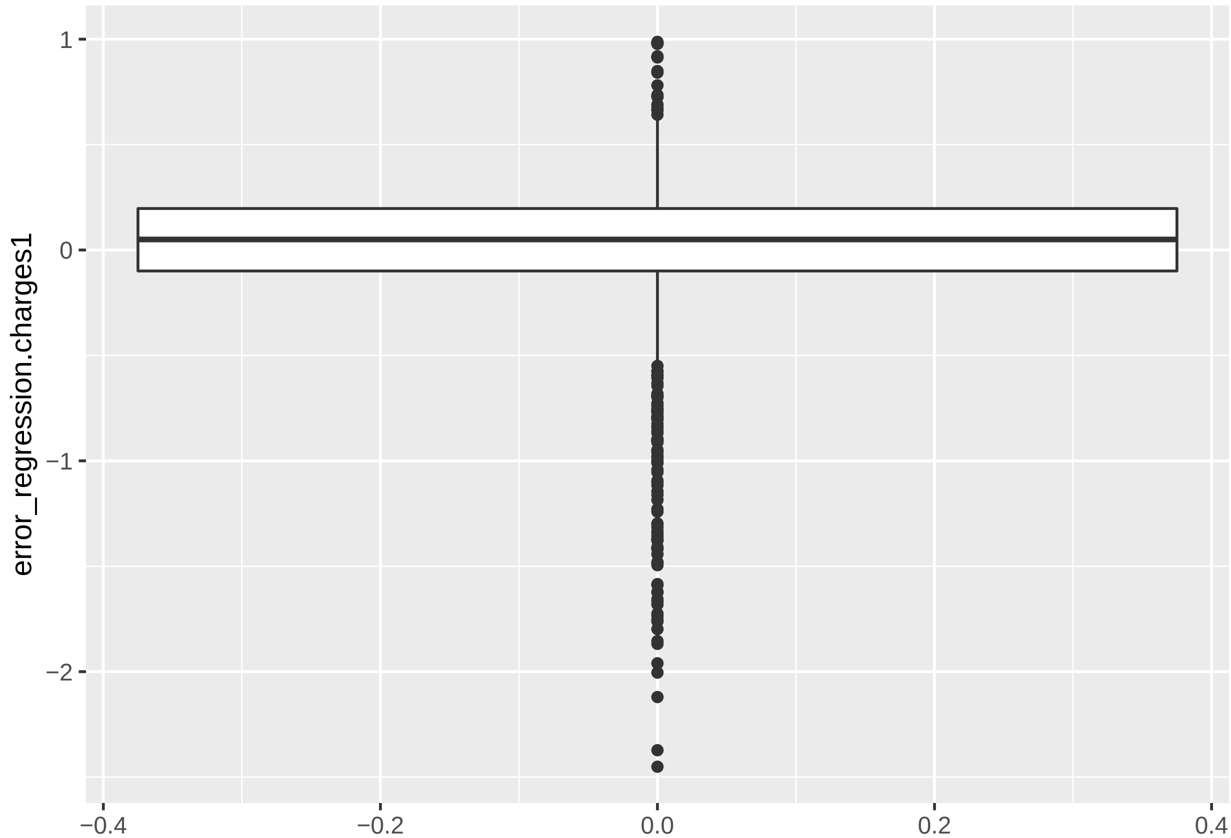
8.1.3.2 Comparison Prediction <-> True Values

```r
error_regression.charges1 <- tree.regression.charges.pred-insurance$charges
element_ID.regression.charges <- 1:length(error_regression.charges1)
plot(element_ID.regression.charges, error_regression.charges1)
title(main="Analysis of the residuals")
abline(mean(error_regression.charges1),0, lwd=5,lty="dotted")
abline(sd(error_regression.charges1) ,0, lwd=2, col="red", lty="dotted")
abline(-sd(error_regression.charges1) ,0, lwd=2, col="red", lty="dotted")
```



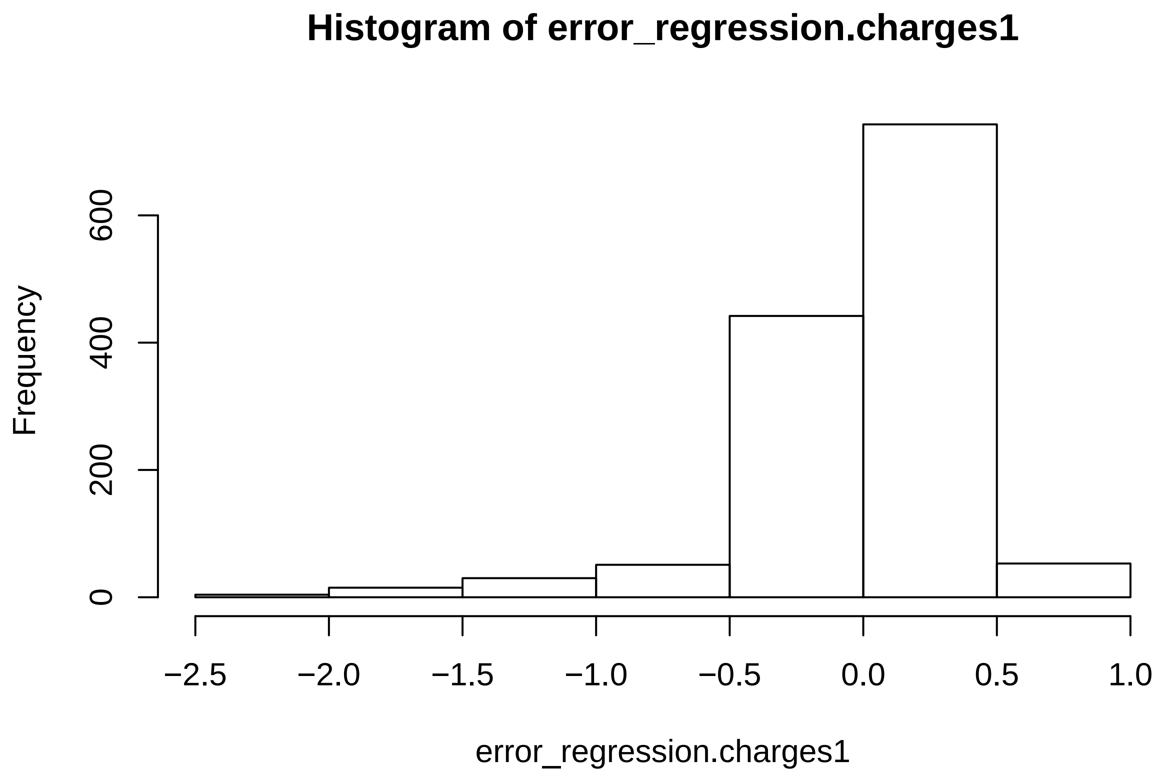
We can furthermore clearly see that the residuals are roughly normalized, with an approximate mean of 0.

```
error_regression1_dataframe.charges <- tibble(element_ID.regression.charges, error_regression.charges1)
ggplot(data=error_regression1_dataframe.charges) + geom_boxplot(aes(y=error_regression.charges1))
```



The boxplot clearly shows that there are more outliers in the negative area of the y-axis.

```
hist(error_regression.charges1)
```



Here goes to further show that the data points are slightly skewed.

```
RSS_charges <- sum((insurance[7]-tree.regression.charges.pred)^2)
MSE_charges <- RSS_charges/length(tree.regression.charges.pred)
deviation_charges <- sqrt(MSE_charges)

cat(RSS_charges)
```

```
## 211.8598
```

```
cat("\n")
```

```
cat(deviation_charges)
```

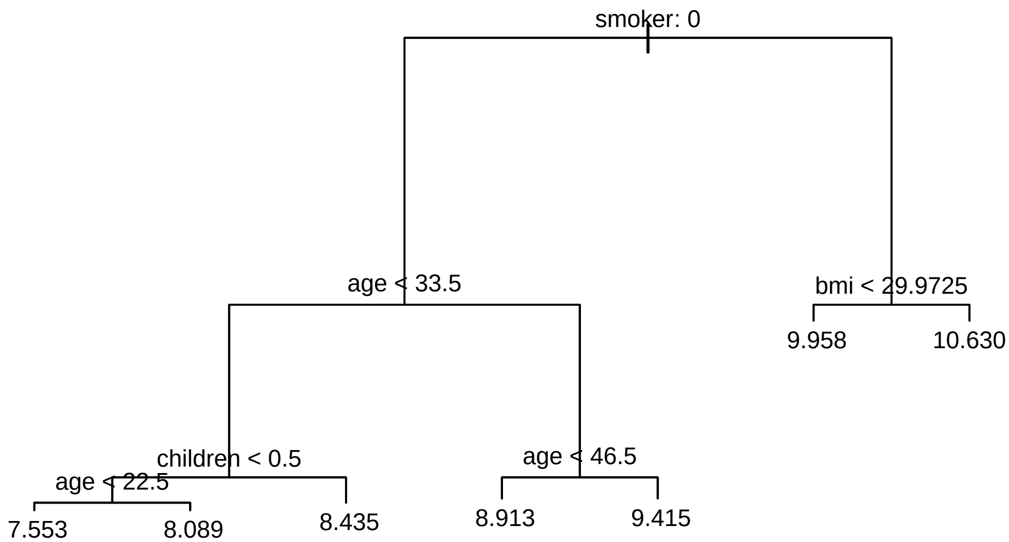
```
## 0.3979204
```

RSS here is at 211.86, the deviation at 0.4.

8.1.3.3 Train-Test-Approach

Again, we use the train-test approach to demonstrate the importance to not use all the data to train one's modell.

```
ratio <- 0.7
total <- nrow(insurance)
train.charges <- sample(1:total, as.integer(total * ratio))
tree.regression.charges2 <- tree(charges~., insurance, subset=train)
plot(tree.regression.charges2)
text(tree.regression.charges2, pretty=12, cex=0.75)
```



We now receive a tree with exactly the same number of nodes but where the criterias have different values.

```
summary(tree.regression.charges2)
```

```
##
## Regression tree:
## tree(formula = charges ~ ., data = insurance, subset = train)
## Variables actually used in tree construction:
## [1] "smoker"   "age"      "children" "bmi"
## Number of terminal nodes:  7
## Residual mean deviance:  0.1475 = 137 / 929
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.98970 -0.18750 -0.04011  0.00000  0.10500  2.41600
```

Our residual mean deivane is now a bit lower at 0.1475 (previously: 0.1592)

```
tree.regression.charges2.pred <- predict(tree.regression.charges2, insurance[-train,], type="vector")
(RSS.2.train.charges <- mean(((insurance[train,][7]-predict(tree.regression.charges2, insurance[train,]
```

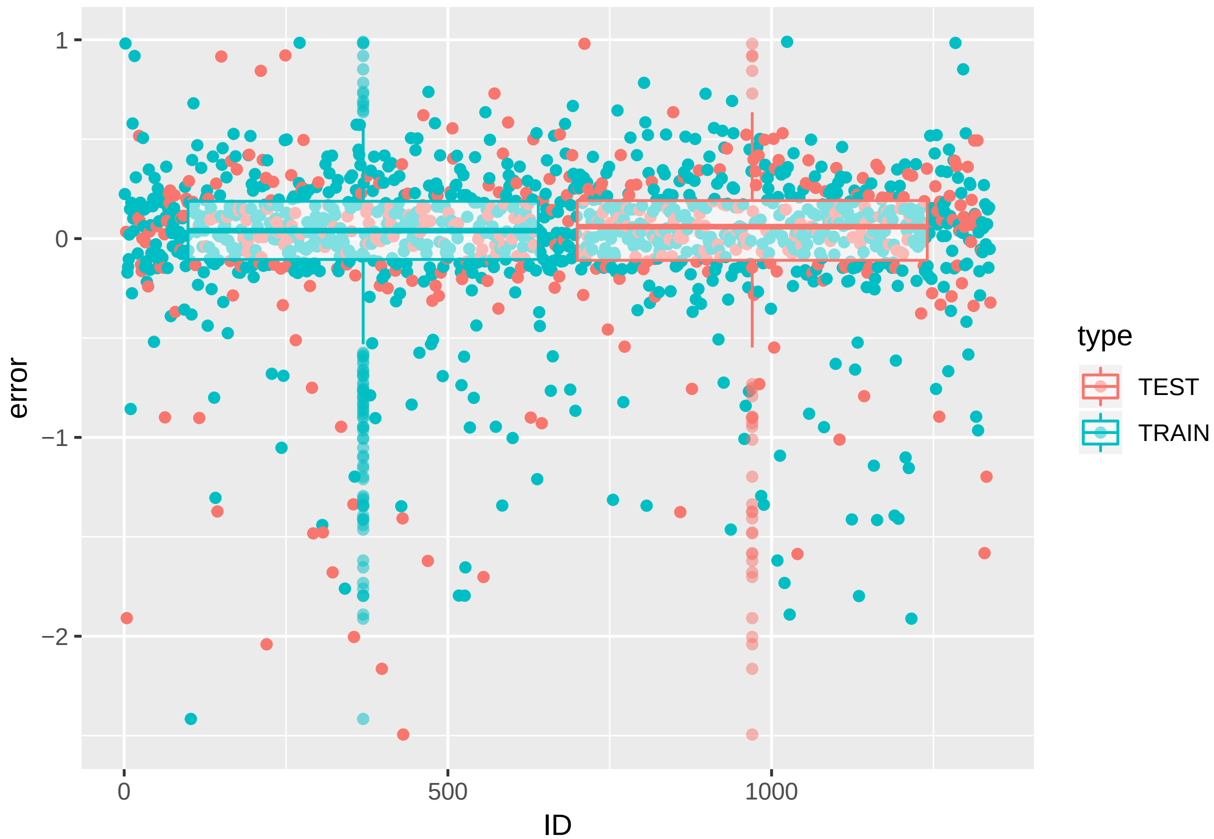
```
## [1] 0.146418
```

```
(RSS.2.test.charges <- mean(((insurance[-train,][7]-tree.regression.charges2.pred)^2)$charges))
```

```
## [1] 0.196484
```

8.1.3.4 Errors

```r
errors.2.in.charges <- predict(tree.regression.charges2, insurance[train,], type="vector")-insurance[tra
element.2.in.charges <- as.integer(names(errors.2.in.charges))
errors.2.in_dataframe.charges <- tibble(element.2.in.charges,errors.2.in.charges,"TRAIN")
colnames(errors.2.in_dataframe.charges) <- c('ID','error','type')
errors.2.charges <- predict(tree.regression.charges2, insurance[-train,], type="vector")-insurance[-trai
element.2.charges <- 1:length(errors.2.charges)
element.2.charges <- as.integer(names(errors.2.charges))
errors.2.out_dataframe.charges <- tibble(element.2.charges,errors.2.charges,"TEST")
colnames(errors.2.out_dataframe.charges) <- c('ID','error','type')
errors.2_dataframe.charges <- bind_rows(errors.2.in_dataframe.charges,errors.2.out_dataframe.charges)
errors.2_dataframe.charges <- arrange(errors.2_dataframe.charges, ID)
ggplot(data = errors.2_dataframe.charges, mapping = aes(x = ID,y = error, color = type)) +
  geom_point() + geom_boxplot(alpha = 0.5)
```



There are a few differences, especially in regard of the outliers between the test and the train data. Roughly, one can say however that the boxplot is about the same for train and test. Also, the outliers in the negative area are for both data sets more spread.

8.2 Classification Trees

We here examine a decision tree for a non continuous (hence categorical) variable.

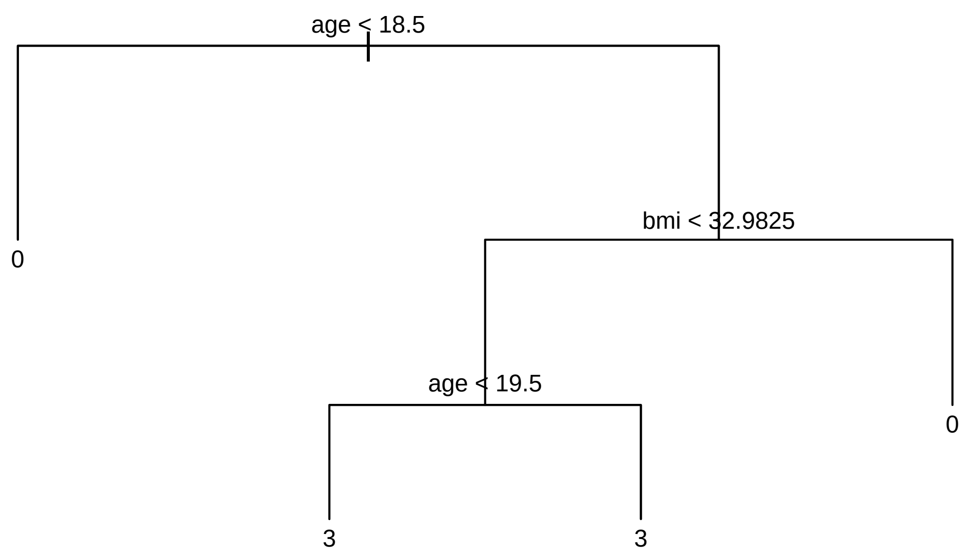
```r
tree.classification.region_all <- tree(region~., data=insurance)
summary(tree.classification.region_all)
```

```
##
## Classification tree:
```

```
## tree(formula = region ~ ., data = insurance)
## Variables actually used in tree construction:
## [1] "age" "bmi"
## Number of terminal nodes:  4
## Residual mean deviance:  2.604 = 3474 / 1334
## Misclassification error rate: 0.6547 = 876 / 1338
```

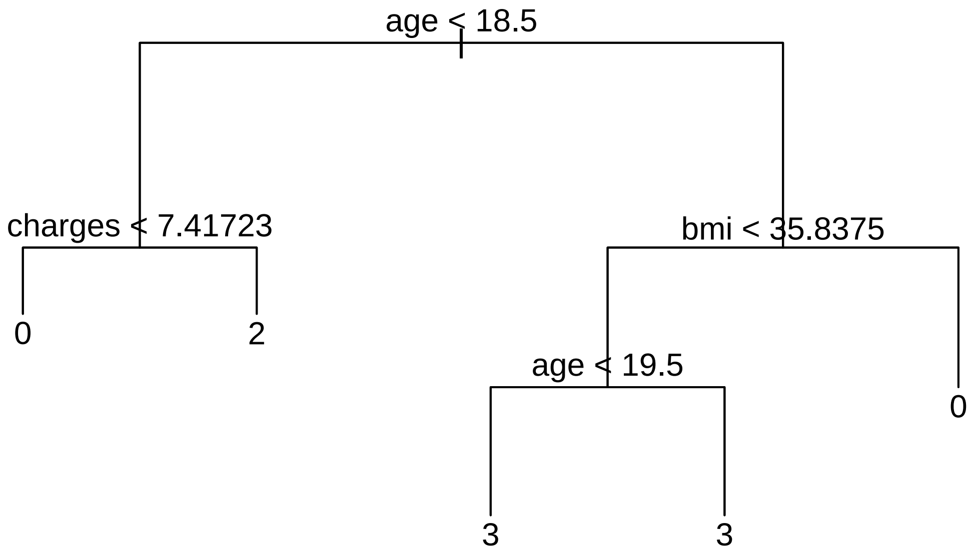
We here receive a classification tree with four end nodes. Furthermore our RMS is at 2.604 and the misclassification rate is at at 65.47%.

```r
plot(tree.classification.region_all)
text(tree.classification.region_all, pretty=1, cex=0.75)
```



It appears that the high misclassication rate is partly due to only 2 of the 4 factors being applied in the classification tree. We thus go into testing mode:

```r
tree.classification.region <- tree(region~., data=insurance, subset=train)
plot(tree.classification.region)
text(tree.classification.region)
```



Working with our training set, we see a clear change: we now have five terminal nodes as well as the factor of "northeast" that is not part of the equation

```
tree.classification.region.pred <- predict(tree.classification.region, insurance[train,], type="class")
```

8.2.1 Confusion Table for Train Data

We now examine the classification error on the train data.

```
tree.classification.region.pred.ct <- table(tree.classification.region.pred, insurance[train,]$region)
tree.classification.region.pred.correct <- 0
tree.classification.region.pred.error <- 0
for (i1 in 1:4) {
  for (i2 in 1:4) {
    if (i1 == i2) {
      tree.classification.region.pred.correct <- tree.classification.region.pred.correct + tree.classifi
    }else{
      tree.classification.region.pred.error <- tree.classification.region.pred.error + tree.classificati
    }
  }
}
tree.classification.region.pred.rate <- tree.classification.region.pred.correct/sum(tree.classification
tree.classification.region.pred.error <- 1 - tree.classification.region.pred.rate
cat(tree.classification.region.pred.rate)
```

```
## 0.357906
```

```r
cat("\n")
```

```r
cat(tree.classification.region.pred.error)
```

```
## 0.642094
```

Our “new” tree manages to reduce our classification error by about 1% for the train data.

8.2.2 Confusion Table for Test Data

```r
tree.classification.region.pred.test <- predict(tree.classification.region, insurance[-train,], type="cl
```

We now determine the classification error on our test data:

```r
(tree.classification.region.pred.test.ct <- table(tree.classification.region.pred.test, insurance[-trai
```

```
##
## tree.classification.region.pred.test  0  1  2  3
##                                     0 41 14 16  9
##                                     1  0  0  0  0
##                                     2  3  0  8  0
##                                     3 73 81 79 78
```

```r
tree.classification.region.pred.correct <- 0
tree.classification.region.pred.error <- 0
for (i1 in 1:3) {
  for (i2 in 1:3) {
    if (i1 == i2) {
      tree.classification.region.pred.correct <- tree.classification.region.pred.correct + tree.classif
    }else{
      tree.classification.region.pred.error <- tree.classification.region.pred.error + tree.classificat
    }
  }
}
(tree.classification.region.pred.rate <- tree.classification.region.pred.correct/sum(tree.classificatio
```

```
## [1] 0.1218905
```

```r
(tree.classification.region.pred.error <- 1 - tree.classification.region.pred.rate)
```

```
## [1] 0.8781095
```

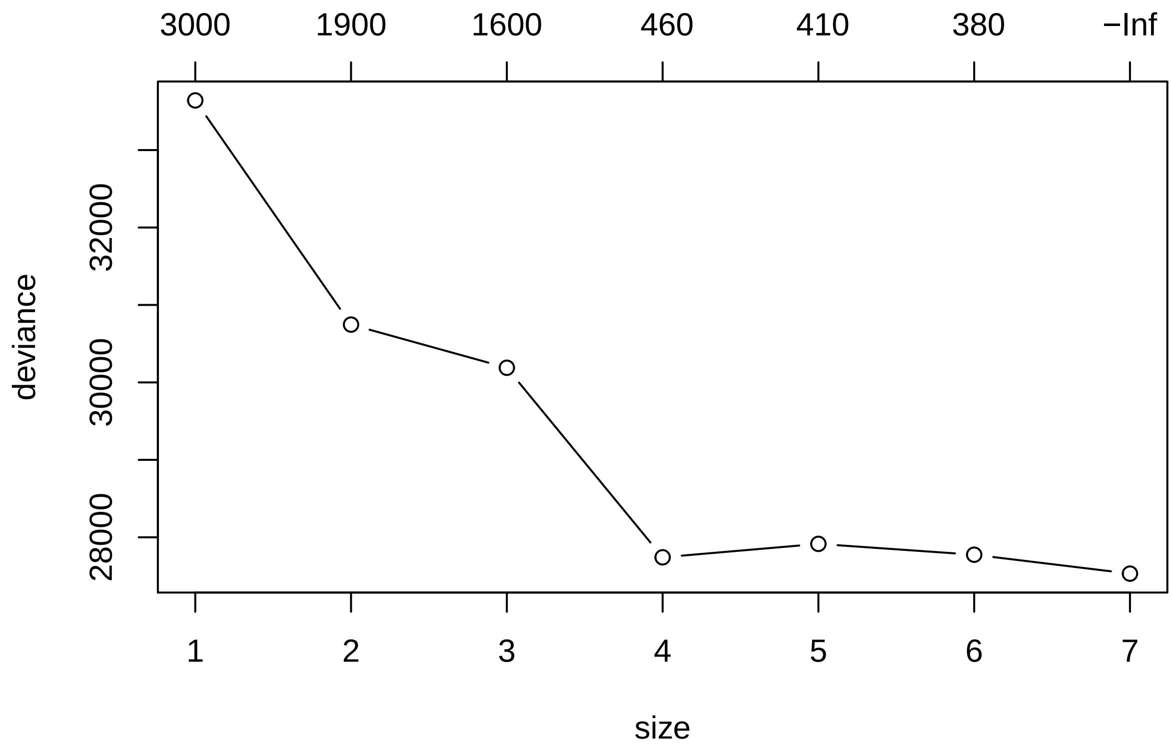
Whereas the model performed a bit better with the train data in comparison with the entire data set, the model now clearly performs in an all time bad. The training modell is thus a clear overfit.

It seems that region, considering the constant high error quote for entire data set, train set as well as test set, is not a variable that has a large connection with the other variables used.

8.3 Pruning

8.3.1 Pruning the continous BMI variable

```r
tree.regression.bmi.pruning = cv.tree(tree.regression.bmi2, FUN = prune.tree)
plot(tree.regression.bmi.pruning, type = "b")
```



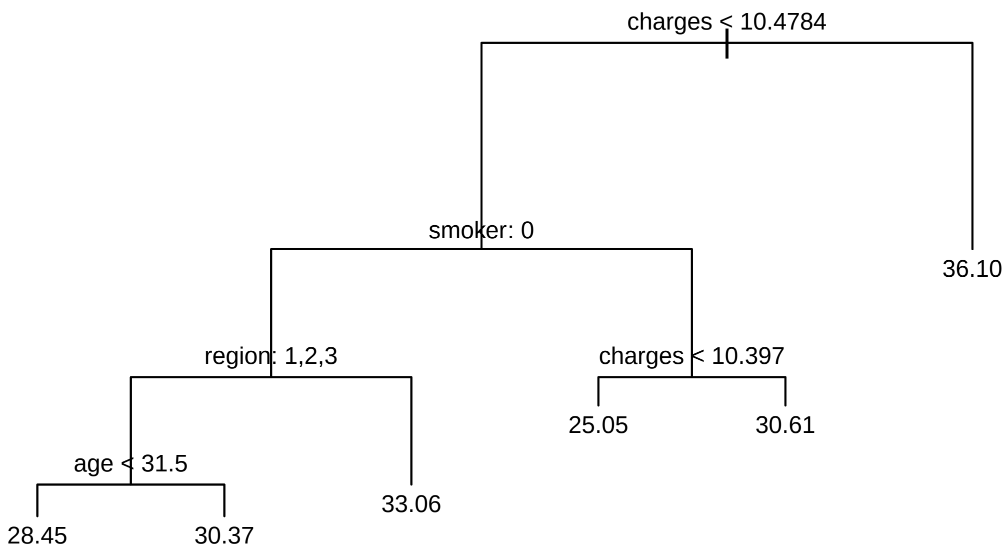
We here see that the tree with seven end nodes has the smallest cross-validation error of about 27'250. This is also further shown in below table.

```
tree.regression.bmi.pruning
```

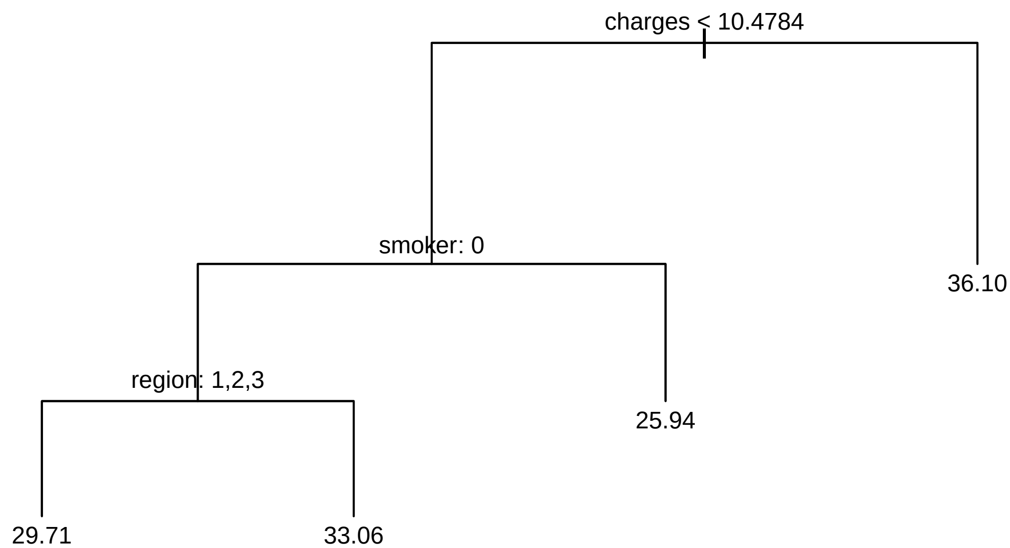
```
## $size
## [1] 7 6 5 4 3 2 1
##
## $dev
## [1] 27530.68 27774.88 27914.67 27741.44 30188.86 30746.46 33641.26
##
## $k
## [1]      -Inf  376.0091  414.7563  462.4535 1571.4867 1872.8382 3018.0179
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

What is intersting to see is that with a tree size of seven, the error rate slight rises but then falls again for a tree with 4 end nodes just to then dramatically rise again.

```
tree.regression.bmi.pruned <- prune.tree(tree.regression.bmi2, best = 6)

plot(tree.regression.bmi.pruned)
text(tree.regression.bmi.pruned, pretty=1, cex=0.75)
```




```
tree.regression.bmi.pruned2 <- prune.tree(tree.regression.bmi2, best = 4)

plot(tree.regression.bmi.pruned2)
text(tree.regression.bmi.pruned2, pretty=1, cex=0.75)
```




```r
summary(tree.regression.bmi.pruned)
```

```
##
## Regression tree:
## snip.tree(tree = tree.regression.bmi2, nodes = 9L)
## Variables actually used in tree construction:
## [1] "charges" "smoker"  "region"  "age"
## Number of terminal nodes:  6
## Residual mean deviance:  28.22 = 26240 / 930
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -13.55000  -3.57100  -0.09678   0.00000   3.33800  16.48000
```

```r
summary(tree.regression.bmi.pruned2)
```

```
##
## Regression tree:
## snip.tree(tree = tree.regression.bmi2, nodes = c(9L, 5L, 8L))
## Variables actually used in tree construction:
## [1] "charges" "smoker"  "region"
## Number of terminal nodes:  4
## Residual mean deviance:  29.1 = 27120 / 932
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -13.750  -3.680  -0.110   0.000   3.635  16.990
```

8.3.2 Pruning regions

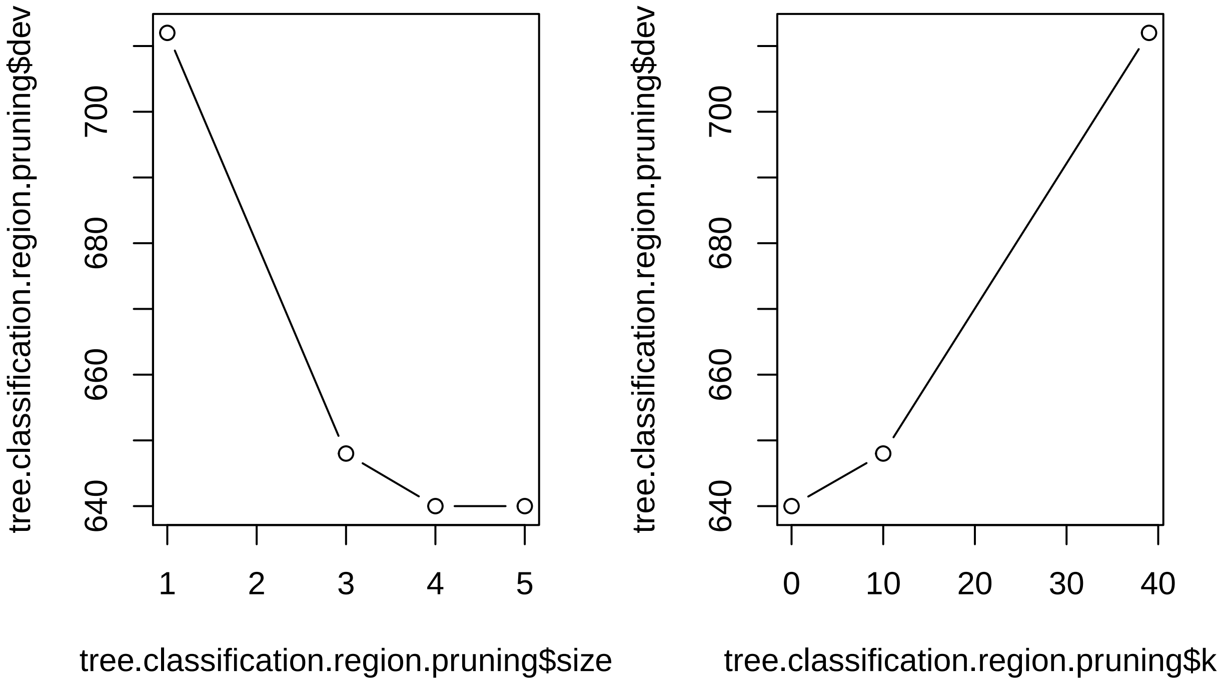
```
tree.classification.region.pruning <- cv.tree(tree.classification.region, FUN = prune.misclass)
summary(tree.classification.region.pruning)
```

```
##        Length Class  Mode
## size   4      -none- numeric
## dev    4      -none- numeric
## k      4      -none- numeric
## method 1      -none- character
```

```
tree.classification.region.pruning
```

```
## $size
## [1] 5 4 3 1
##
## $dev
## [1] 640 640 648 712
##
## $k
## [1] -Inf    0   10   39
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
par(mfrow=c(1,2))
plot(tree.classification.region.pruning$size, tree.classification.region.pruning$dev, type="b")
plot(tree.classification.region.pruning$k, tree.classification.region.pruning$dev, type="b")
```



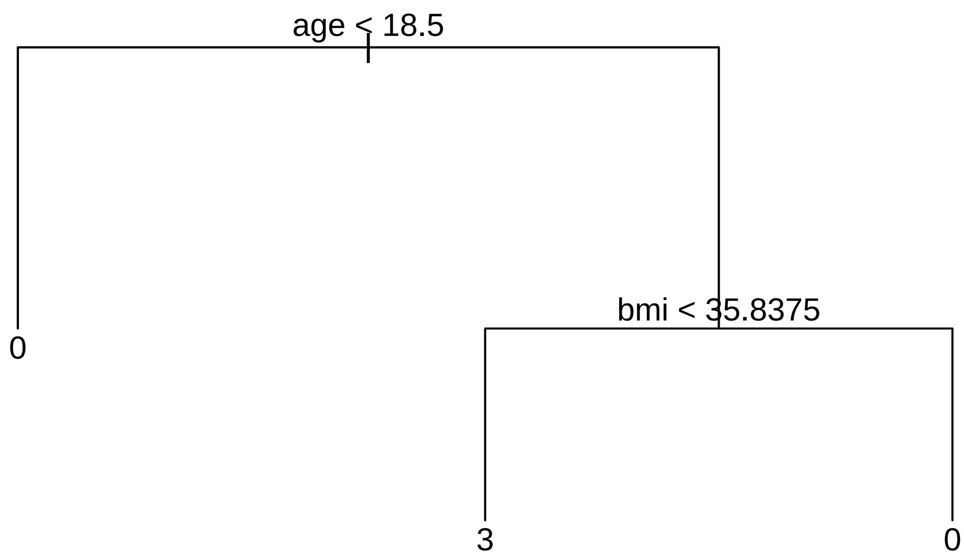

```r
par(mfrow=c(1,1))
```

Thanks to the prune.misclass function, we do not have to worry about cost complexity pruning ourselves but can simply apply the function as can be seen below:

```r
prune.tree.classification.region <- prune.misclass(tree.classification.region, best=3)

summary (prune.tree.classification.region)
```

```
##
## Classification tree:
## snip.tree(tree = tree.classification.region, nodes = c(6L, 2L
## ))
## Variables actually used in tree construction:
## [1] "age" "bmi"
## Number of terminal nodes:  3
## Residual mean deviance:  2.633 = 2456 / 933
## Misclassification error rate: 0.6528 = 611 / 936
```

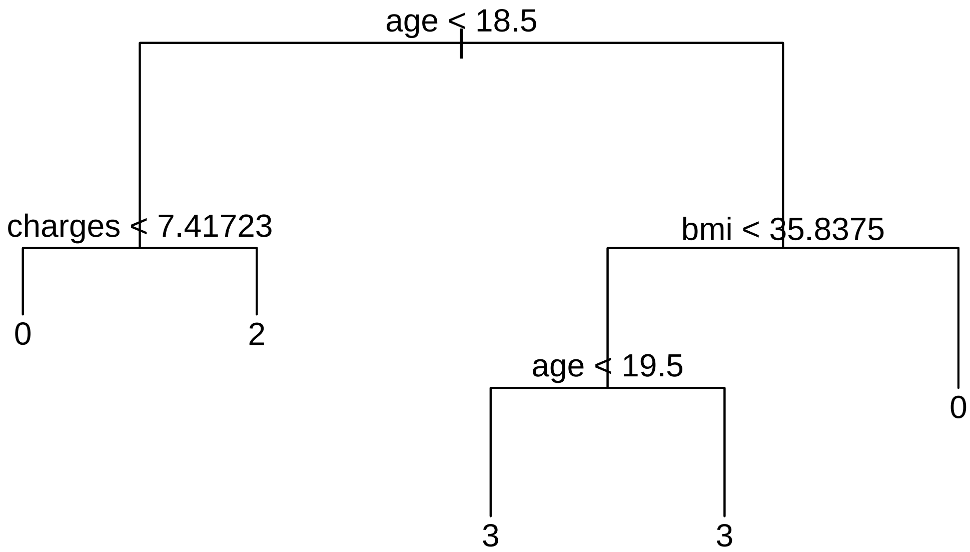
```r
plot(prune.tree.classification.region)
text(prune.tree.classification.region,pretty=0)
```




```r
prune.tree.classification.region <- prune.misclass(tree.classification.region, best=5)
summary (prune.tree.classification.region)
```

```
##
## Classification tree:
## tree(formula = region ~ ., data = insurance, subset = train)
## Variables actually used in tree construction:
## [1] "age"     "charges" "bmi"
## Number of terminal nodes:  5
## Residual mean deviance:  2.555 = 2379 / 931
## Misclassification error rate: 0.6421 = 601 / 936
```

```r
plot(prune.tree.classification.region)
text(prune.tree.classification.region,pretty=0)
```




```
prune.tree.classification.region.pred <- predict(prune.tree.classification.region,  insurance[-train,],

prune.tree.classification.region.pred.ct <- table(prune.tree.classification.region.pred, insurance[-tra
prune.tree.classification.region.pred.correct <- sum(prune.tree.classification.region.pred==insurance[-

prune.tree.classification.region.pred.testError <- 1 - prune.tree.classification.region.pred.correct
```

```
cat(tree.classification.region.pred.error)
```

```
## 0.8781095
```

```
cat("\n")
```

```
cat(prune.tree.classification.region.pred.testError)
```

```
## 0.6840796
```

8.4 Bagging

```
set.seed (5)
train = sample(1:nrow(insurance), nrow(insurance)/2)
insurance.test=insurance[-train ,]
```

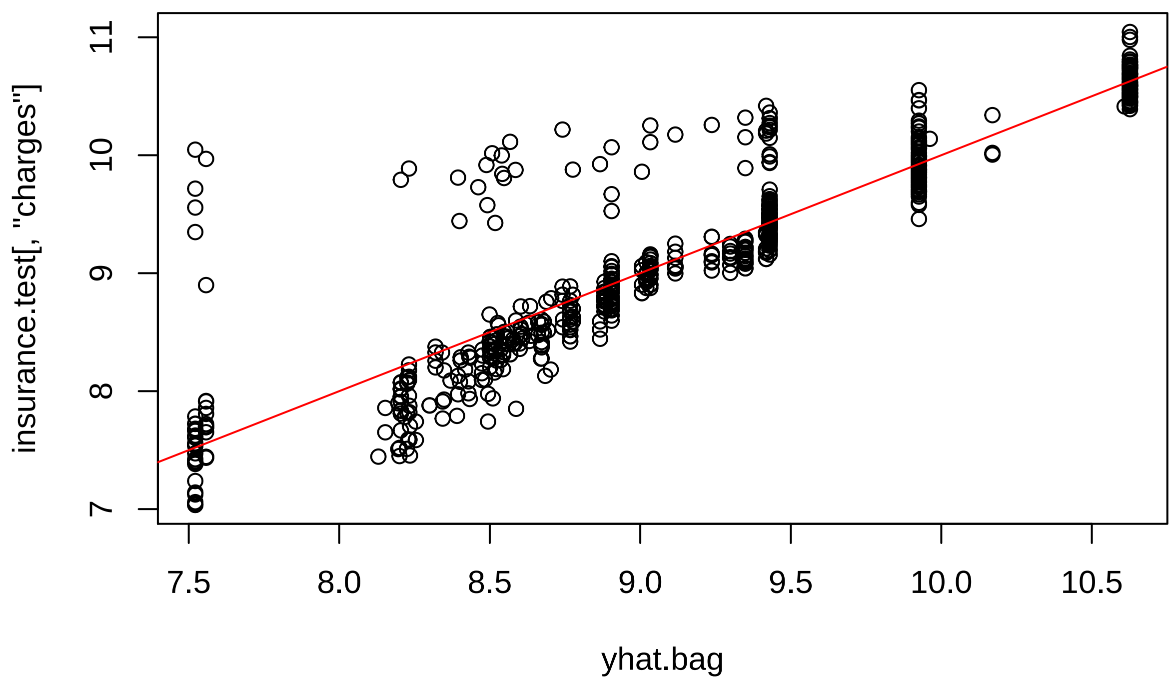
```
bag.insurance=bagging(charges~., data=insurance, subset=train, nbagg=40, coob = TRUE)
print(bag.insurance)
```

```
##
## Bagging regression trees with 40 bootstrap replications
```

```
##
## Call: bagging.data.frame(formula = charges ~ ., data = insurance, subset = train,
##     nbagg = 40, coob = TRUE)
##
## Out-of-bag estimate of root mean squared error:  0.3887
```

```
#bag.insurance still needs to be squared at this point
```

```
yhat.bag = predict(bag.insurance, newdata=insurance[-train,])
plot(yhat.bag, insurance.test[,"charges"])
abline(0,1, col="red")
```




```
mean((yhat.bag-insurance.test[,"charges"])^2)
```

```
## [1] 0.1557692
```

We also test this

```
## test with randomForest
bag.insurance.2=randomForest(charges~., data=insurance, subset=train, mtry=6, importance =TRUE)
# mtry = 7 means that we should use all 13 predictors for each split of the tree, hence, do bagging (no
print(bag.insurance.2)
```

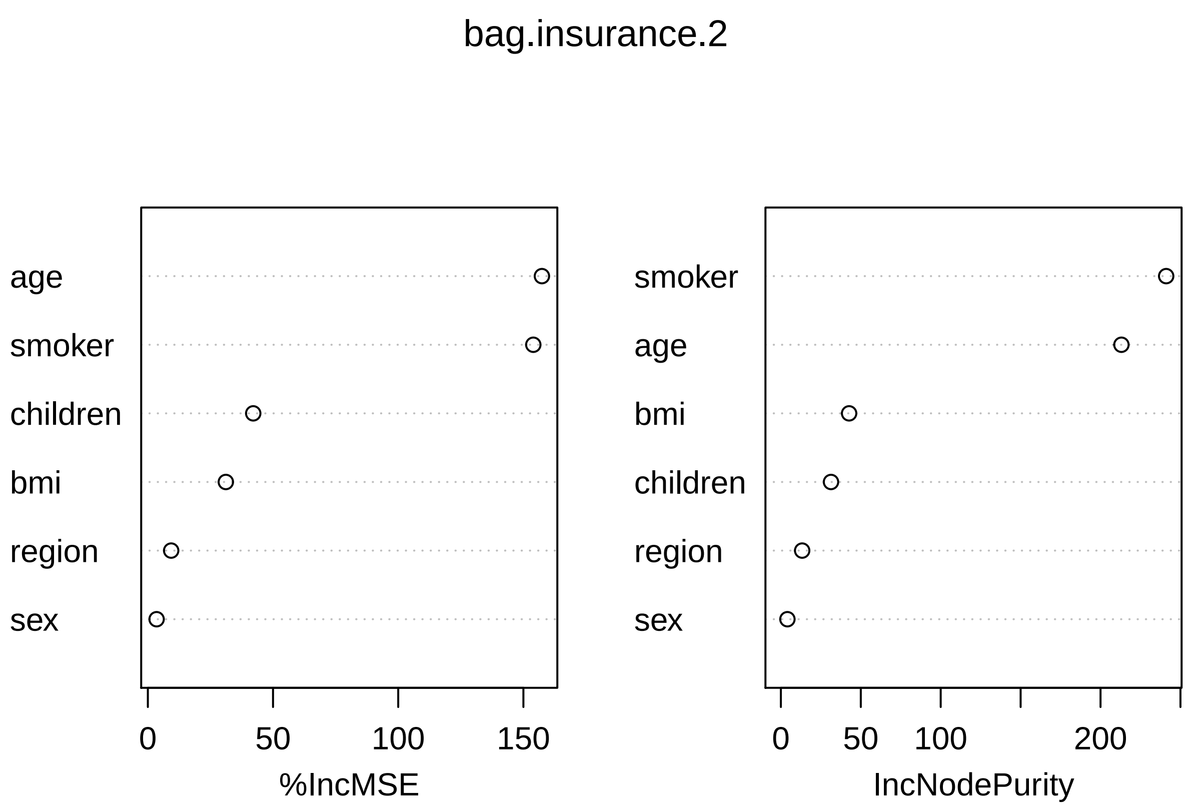
```
##
## Call:
##  randomForest(formula = charges ~ ., data = insurance, mtry = 6,      importance = TRUE, subset = tra
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 6
```

```
##
##           Mean of squared residuals: 0.1505445
##                     % Var explained: 82
```

```
importance(bag.insurance.2)
```

```
##              %IncMSE IncNodePurity
## age       157.398119    213.067518
## sex         3.494398      4.097002
## bmi        31.150730     42.693249
## children   42.085078     31.371187
## smoker    153.956154    241.078194
## region      9.313146     13.327420
```

```
varImpPlot(bag.insurance.2)
```



8.5 Random Forests

```
set.seed(10)
rf.insurance=randomForest(charges~.,data=insurance,subset=train, mtry=2,importance =TRUE)
yhat.rf = predict(rf.insurance ,newdata=insurance[-train ,])
mean((yhat.rf-insurance.test[,"charges"])^2)
```

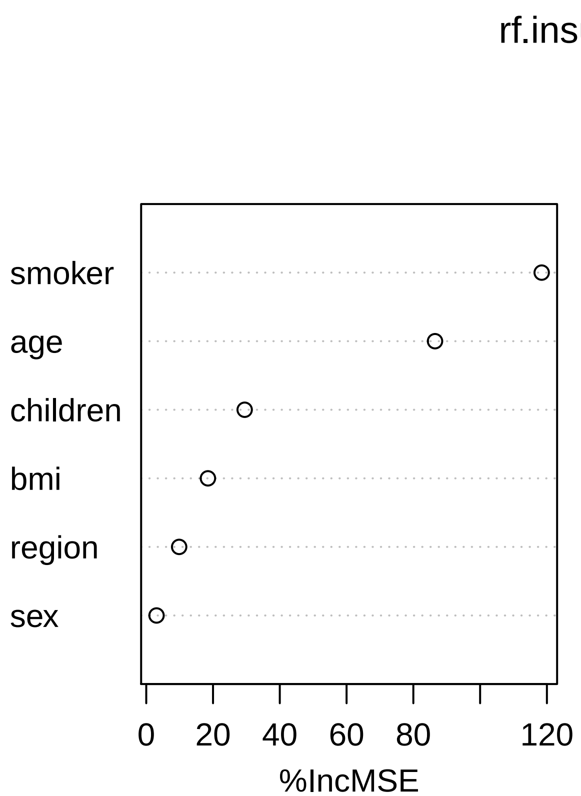
```
## [1] 0.158069
```

```
importance(rf.insurance)
```

```
##              %IncMSE IncNodePurity
```

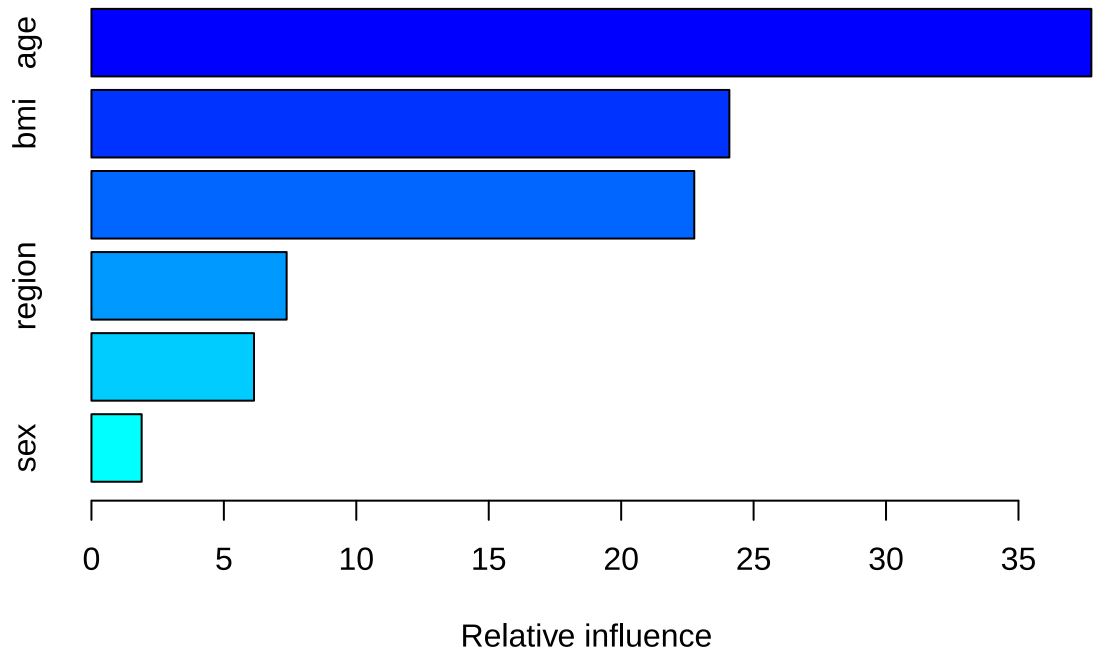
```
## age        86.489219    182.623763
## sex         3.084226      6.073046
## bmi        18.476454     50.077296
## children   29.481898     25.129388
## smoker    118.440509    225.385251
## region      9.858480     14.461133
```

```r
varImpPlot (rf.insurance)
```



8.6 Boosting

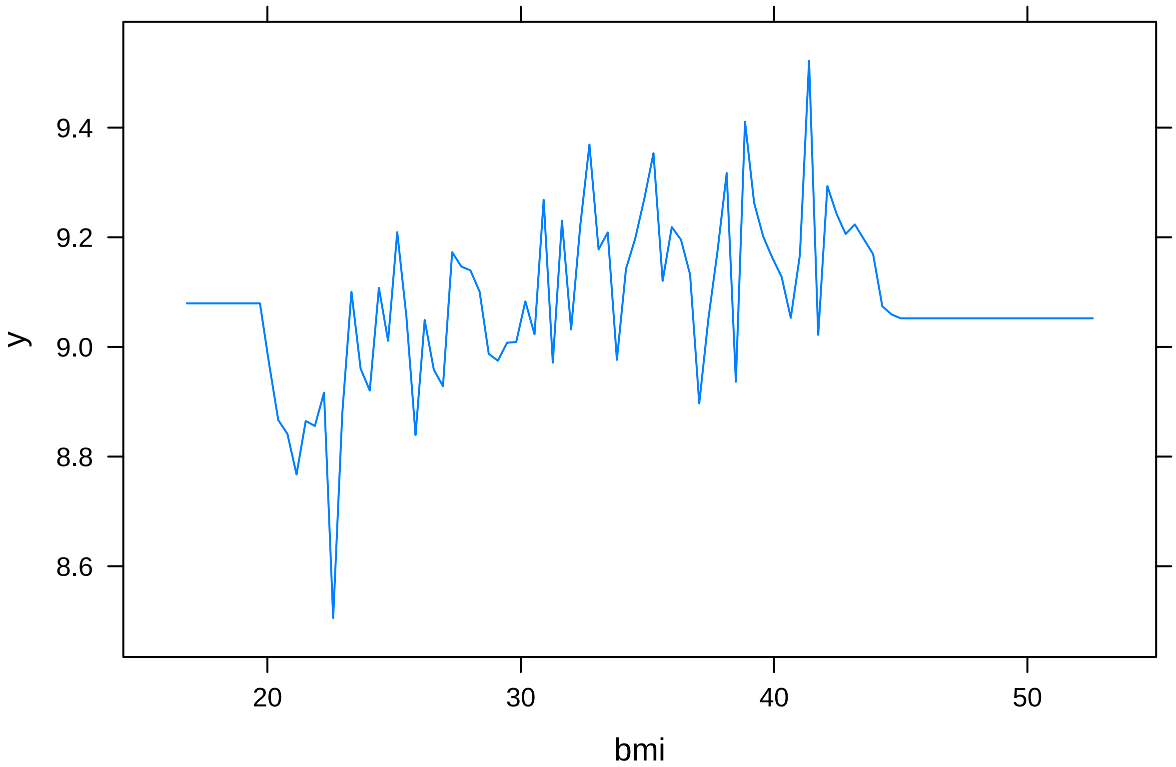
```r
set.seed (1)
boost.insurance=gbm(charges~.,data=insurance[train,],
                 distribution="gaussian",n.trees=5000, interaction.depth=4)
summary(boost.insurance)
```



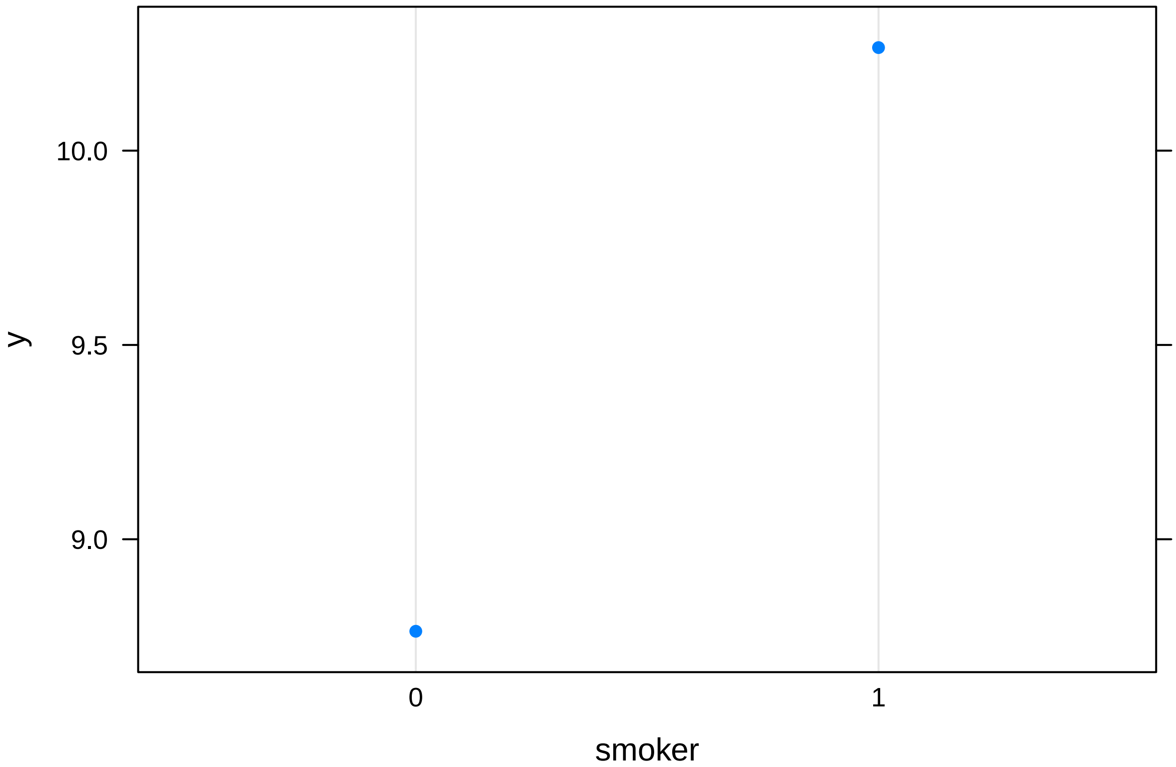

```
##                 var   rel.inf
## age             age 37.751923
## bmi             bmi 24.084978
## smoker       smoker 22.759940
## region       region  7.370245
## children   children  6.137392
## sex             sex  1.895522
```

partial dependence plots for most important factors

```
plot(boost.insurance ,i="bmi")
```




```
plot(boost.insurance ,i="smoker")
```



boosted model to predict charges on the test set. Report the MSE.

```
yhat.boost=predict(boost.insurance, newdata=insurance[-train,], n.trees=5000)
mean((yhat.boost -insurance.test[,"charges"])^2)
```

```
## [1] 0.2383602
```

resetting shrinkage parameter

```
boost.insurance = gbm(charges~.,data=insurance[train,],distribution="gaussian",n.trees=5000, interactio
yhat.boost2=predict(boost.insurance,newdata=insurance[-train,], n.trees=5000)
mean((yhat.boost2 -insurance.test[,"charges"])^2)
```

```
## [1] 0.193239
```

again resetting

```
boost.insurance2=gbm(charges~.,data=insurance[train,],distribution="gaussian",n.trees=5000, interaction
yhat.boost3=predict(boost.insurance2,newdata=insurance[-train,], n.trees=5000)
mean((yhat.boost -insurance.test[,"charges"])^2)
```

```
## [1] 0.2383602
```

higher test MSE is going to be the worst model –> overfitting effect

(Attention: Please do only execute the code block (applying the exponential function) below once as executing it several times will apply the function several times)

```
insurance$charges <- exp(insurance$charges) #resetting the insurance variable
```

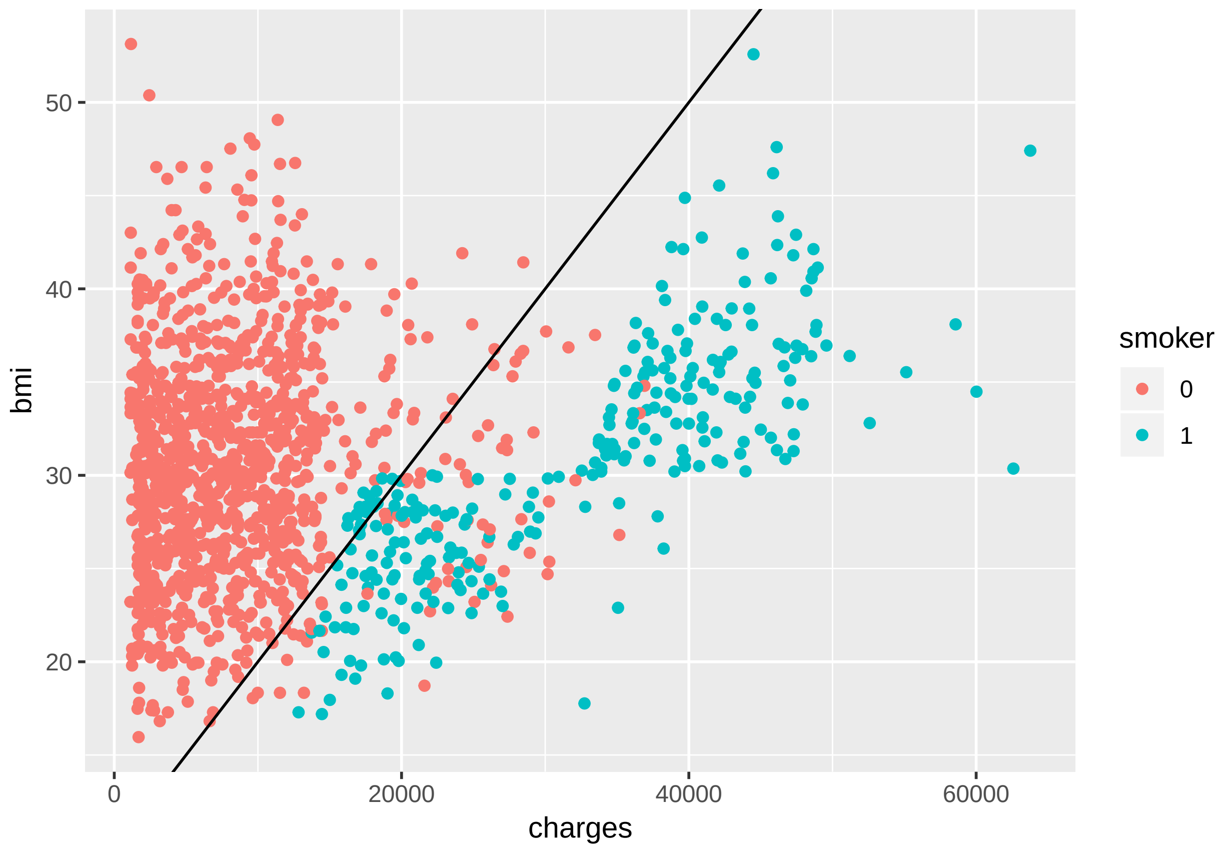
9 Support Vector Machines -> Carole

In this chapter, Support Vector Machines are applied to the insurance dataset.

9.1 two classes linear

By visual exploration, we discover that smokers and non-smokers form two groups. We fit an SVM for those two groups. The lines in the graph are the first guess of how the data could be divided.

```
# guess of hyperplane
slope <- 0.001
intercept <- 10
ggplot(data = insurance) +
  geom_point(aes(x = charges, y = bmi, color=smoker))+
  geom_abline(slope = slope, intercept = intercept)
```



A linear SVM is tuned using different cost ranges.

```
set.seed(5)
#get optimal cost
cost_range <-
  c(1e-10,
    1e-7,
    1e-5,
    0.001,
    0.0025,
    0.005,
    0.0075,
```

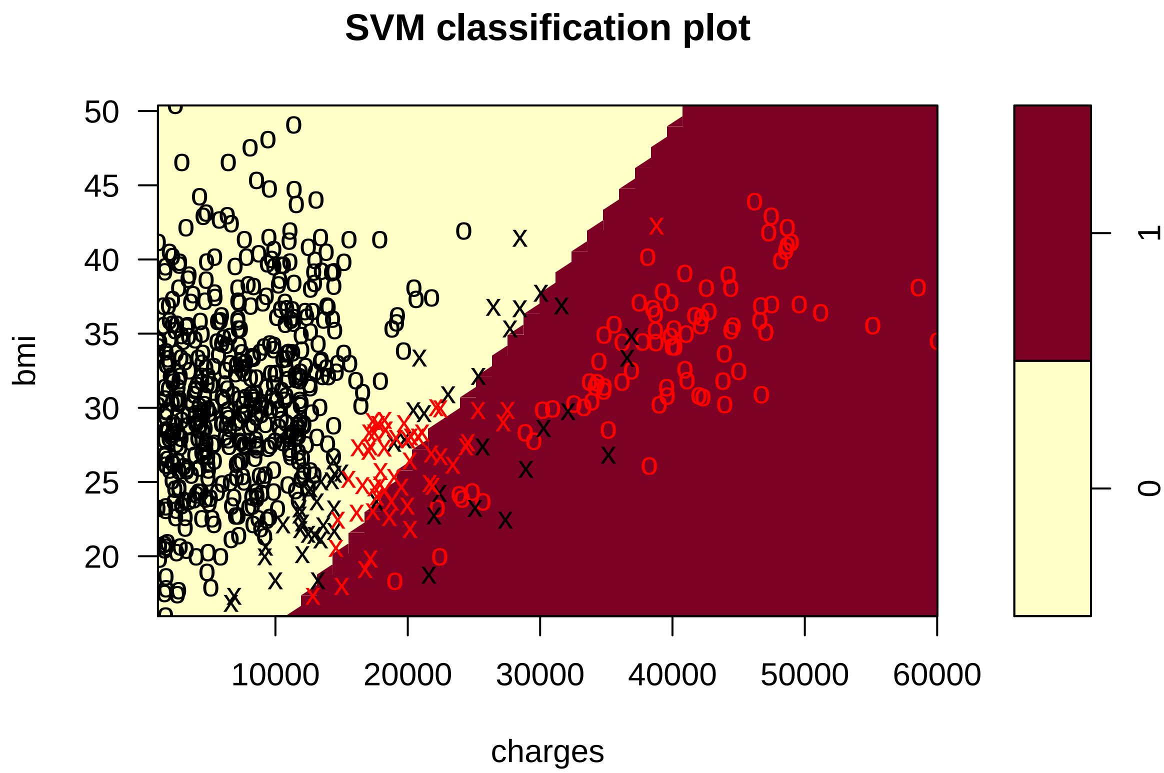
```r
    0.01,
    0.1,
    1,
    5,
    10,
    100,
    200)
tune.out.1 <- tune(
  svm,
  smoker ~ charges+bmi,
  data = insurance.train,
  kernel = "linear",
  ranges = list(cost = cost_range)
)
```

The following is the best model. A cost of 5 is used. There are 104 support vectors.

```r
#best model
bestmod <- tune.out.1$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = smoker ~ charges + bmi, data = insurance.train,
##     ranges = list(cost = cost_range), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  5
##
## Number of Support Vectors:  104
##
##  ( 52 52 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
plot(bestmod, insurance.train, bmi~charges)
```



The confision matrices and the classifiaction error rate for train and test data are:

```
# confusion matrix train
confusion.lin.train <- table(predict = predict(bestmod, insurance.train),
      truth = insurance.train$smoker)
confusion.lin.train
```

```
##          truth
## predict    0   1
##         0 514  26
##         1  18 106
```

```
#classification error rate train
(confusion.lin.train[1,2]+confusion.lin.train[2,1])/sum(confusion.lin.train[1:2,1:2])
```

```
## [1] 0.06626506
```

```
# confusion matrix linear test
confusion.lin.test <- table(predict = predict(bestmod, insurance.test),
      truth = insurance.test$smoker)
confusion.lin.test
```

```
##          truth
## predict    0   1
##         0 527 142
##         1   0   0
```

```
#classification error rate test
(confusion.lin.test[1,2]+confusion.lin.test[2,1])/sum(confusion.lin.test[1:2,1:2])
```

```
## [1] 0.2122571
```

9.2 two classes polynomial

A polynomial model was also tested but the classification error rate was slightly higher:

```
set.seed(5)
#get optimal cost and degree for polynomial
cost_range <-
  c(1e-10,
    1e-7,
    1e-5,
    0.001,
    0.0025,
    0.005,
    0.0075,
    0.01,
    0.1,
    1,
    5,
    10,
    100)
degree_range <- 2:4
tune.out.poly <- tune(
  svm,
  smoker ~ charges+bmi,
  data = insurance.train,
  kernel = "polynomial",
  ranges = list(cost = cost_range, degree = degree_range))
#best model poly
tune.out.poly$best.parameters
```

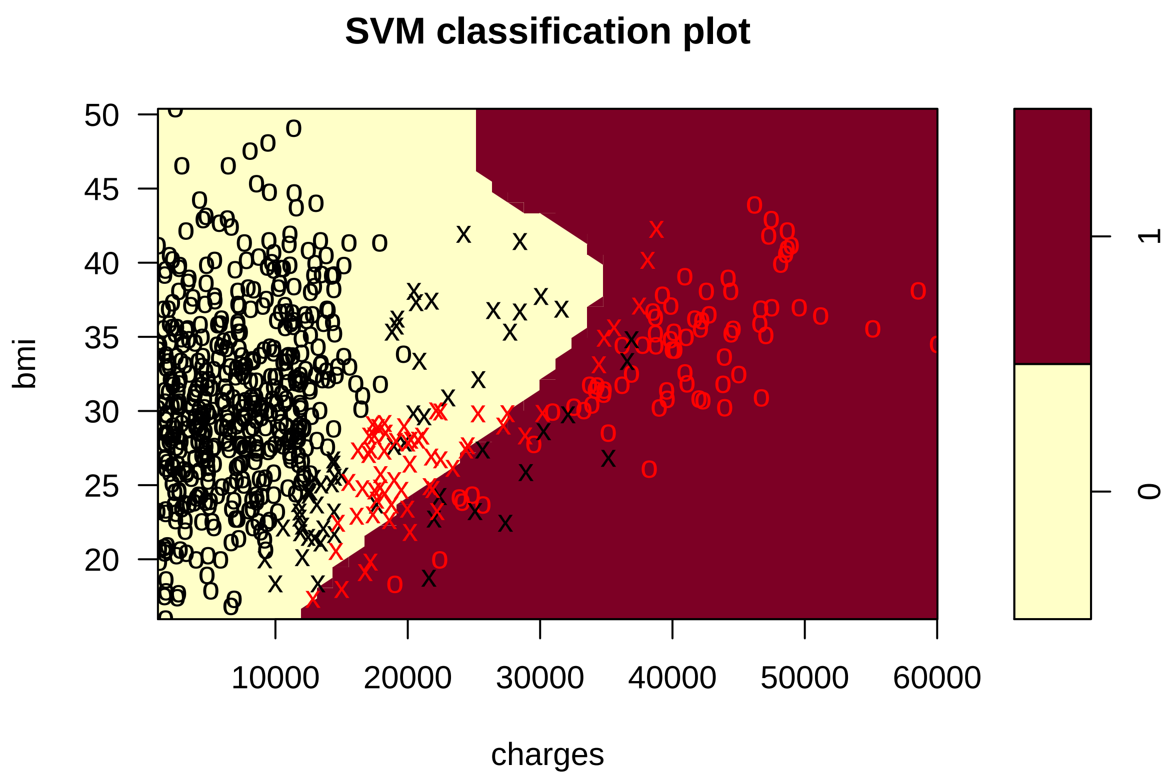
```
##    cost degree
## 26  100      3
```

```
bestmod.poly <- tune.out.poly$best.model
summary(bestmod.poly)
```

```
##
## Call:
## best.tune(method = svm, train.x = smoker ~ charges + bmi, data = insurance.train,
##     ranges = list(cost = cost_range, degree = degree_range),
##     kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  100
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  120
##
##  ( 60 60 )
##
```

```
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
plot(bestmod.poly, insurance.train, bmi~charges)
```




```
# confusion matrix
confusion.poly.train <- table(predict = predict(bestmod.poly, insurance.train),
      truth = insurance.train$smoker)
confusion.poly.test <- table(predict = predict(bestmod.poly, insurance.test),
      truth = insurance.test$smoker)
#classification error rate
(confusion.poly.train[1,2]+confusion.poly.train[2,1])/sum(confusion.poly.train[1:2,1:2])
```

```
## [1] 0.0753012
```

```
(confusion.poly.test[1,2]+confusion.poly.test[2,1])/sum(confusion.poly.test[1:2,1:2])
```

```
## [1] 0.2122571
```

10 Neural Networks

10.1 Neural Networks using "neuralnet"

In this section we will create and train a neural network which should be able to classify if a person is a smoker (output) based on the information of age, bmi and charges that are provided as input factors for the model. The assumption of the input variables is retrieved from our analysis already done in section 5. Generalized Linear Models.

10.1.1 Cleaning and Normalization

This part will be needed to ensure a cleansed and normalized database for a proper modeling of neural networks.

```
#setwd("~/GitHub/machinelearning/machinelearning/03_rmarkdown") #adrianas extrawurst
insurance <- read.csv("../01_data/insurance.csv", header=TRUE)
str(insurance)
```

```
## 'data.frame':    1338 obs. of  7 variables:
##  $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 1 1 2 1 ...
##  $ bmi     : num  27.9 33.8 33 22.7 28.9 ...
##  $ children: int  0 1 3 0 0 0 1 3 2 0 ...
##  $ smoker  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...
##  $ region  : Factor w/ 4 levels "northeast","northwest",..: 4 3 3 2 2 3 3 2 1 2 ...
##  $ charges : num  16885 1726 4449 21984 3867 ...
```

```
# smoker = 1 / nonsmoker = 2
insurance$smoker <- as.character(insurance$smoker)
insurance$smoker[insurance$smoker == "yes"] <- "1"
insurance$smoker[insurance$smoker == "no"] <- "2"
insurance$smoker <- as.factor(insurance$smoker)
# female = 1 / male = 0
insurance$sex <- as.character(insurance$sex)
insurance$sex[insurance$sex == "female"] <- "1"
insurance$sex[insurance$sex == "male"] <- "0"
insurance$sex <- as.factor(insurance$sex)
# region / SE = 1 / SW = 0 / NE = 2 / NW = 3
insurance$region <- as.character(insurance$region)
insurance$region[insurance$region == "southwest"] <- "1"
insurance$region[insurance$region == "southeast"] <- "0"
insurance$region[insurance$region == "northeast"] <- "2"
insurance$region[insurance$region == "northwest"] <- "3"
insurance$region <- as.factor(insurance$region)
head(insurance)
```

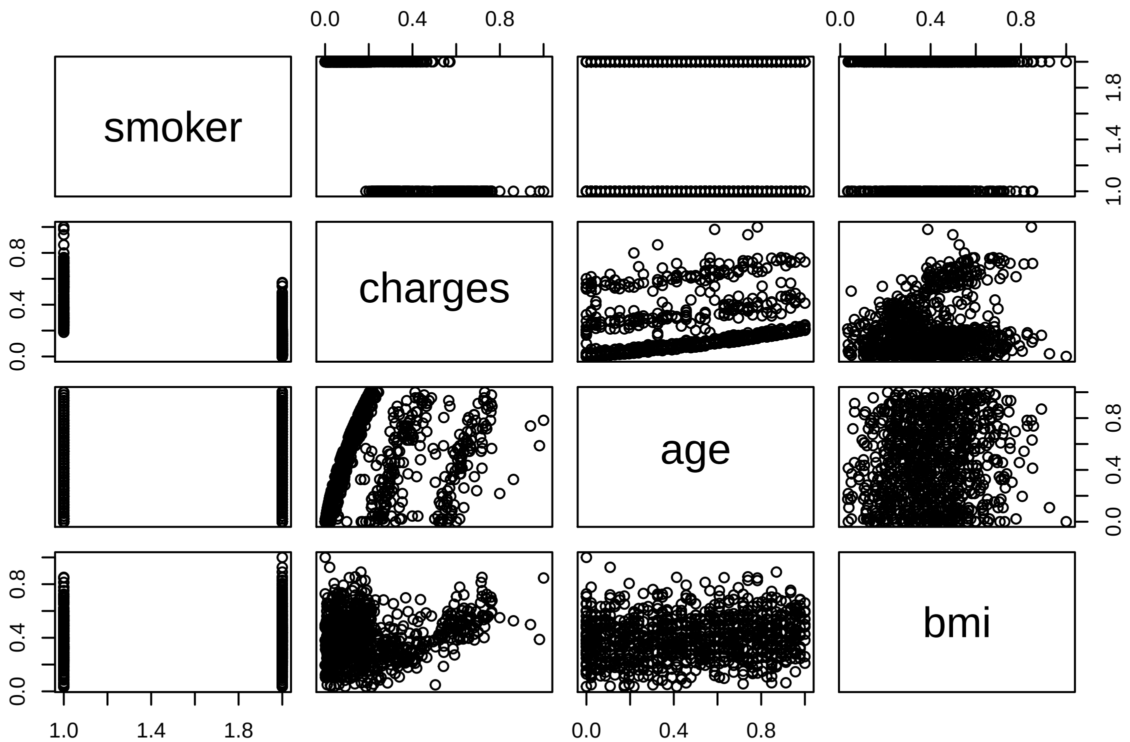
```
##   age sex    bmi children smoker region   charges
## 1  19   1 27.900        0      1      1 16884.924
## 2  18   0 33.770        1      2      0  1725.552
## 3  28   0 33.000        3      2      0  4449.462
## 4  33   0 22.705        0      2      3 21984.471
## 5  32   0 28.880        0      2      3  3866.855
## 6  31   1 25.740        0      2      0  3756.622
```

```
options(scipen = 99) # penalty for displaying scientific notation
options(digits = 4) # suggested number of digits to display

data <-dplyr::select(insurance, smoker, charges, age, bmi)

data$charges <- (data$charges - min(data$charges))/(max(data$charges) - min(data$charges)) # Min-Max No
data$age <- (data$age - min(data$age))/(max(data$age) - min(data$age)) # Min-Max Normalization
data$bmi <- (data$bmi - min(data$bmi))/(max(data$bmi)-min(data$bmi)) # Min-Max Normalization
```

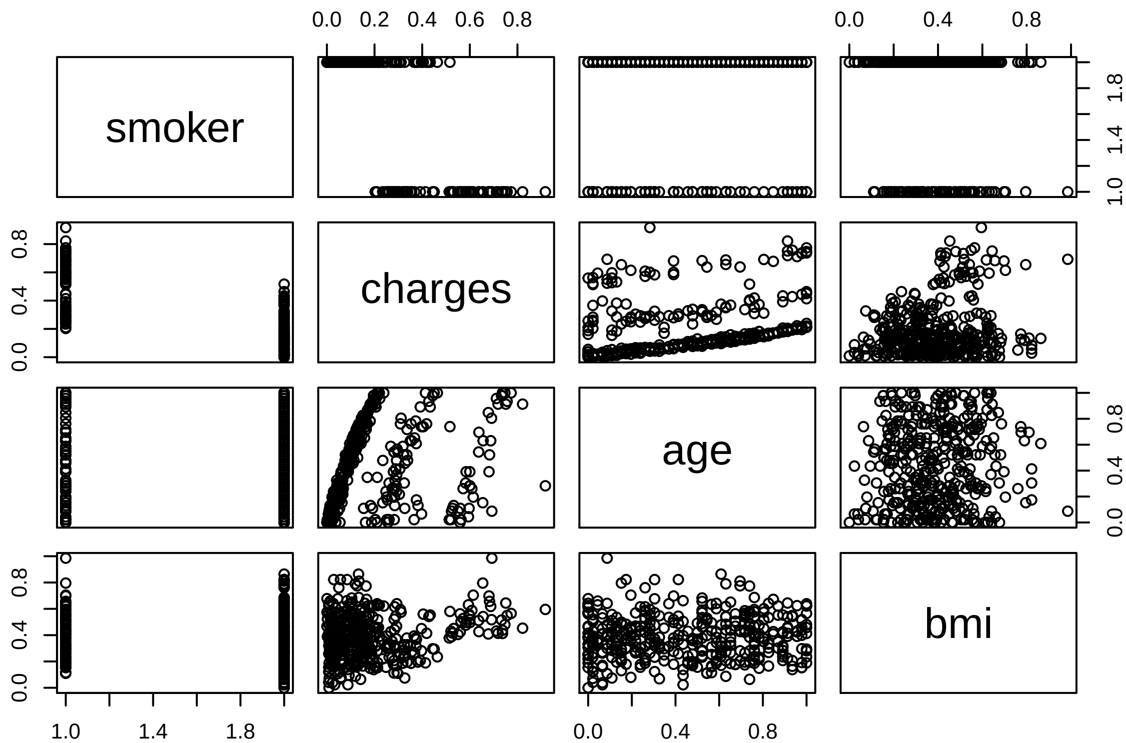
10.1.2 Partition into training and test subsets

As shown above we already cleansed and normalized data. Now we will create two subsets of the insurance data set. Therefore we will execute a partition, so we will have a training set covering 70% of the original dataset while the test set will cover the remaining 30%.

```
set.seed(222)
ind <- sample(2, nrow(data), replace = TRUE, prob = c(0.7, 0.3))
training <- data[ind==1,]
testing <- data[ind==2,]
par(mfrow = c(1, 1))
plot(training)
```




```
plot(testing)
```

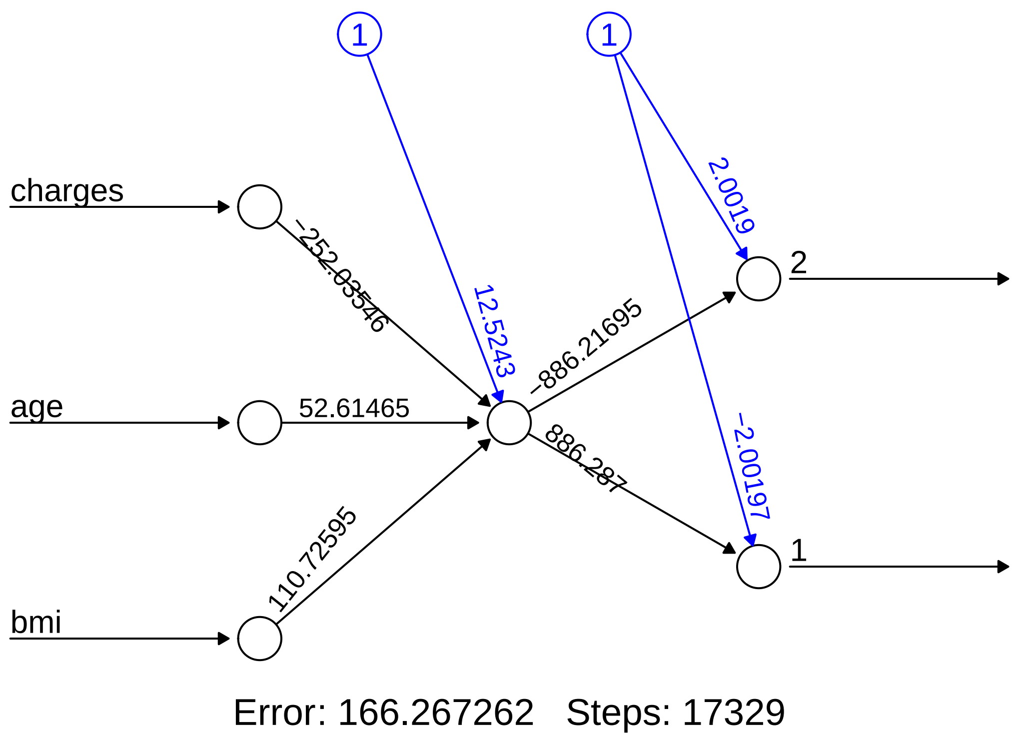


10.1.3 Creation and Training of the Neural Network

10.1.3.1 Not deep NN - one single hidden layer

10.1.3.1.1 Creation of NN_1

```
set.seed(444)
nn <- neuralnet(smoker~charges + age + bmi,
                data = training,
                hidden = 1,
                err.fct = "ce",
                linear.output = FALSE)
#summary(nn)
#print(nn)
plot(nn, rep='best')
```



Prediction on training subset

```
prediction_nn <- predict(nn,training)
#prediction_nn
class_nn <- apply(prediction_nn, 1, which.max)
```

10.1.3.1.2 Confusion Matrix for training

```
table(training$smoker, class_nn)
```

```
##    class_nn
##       1   2
##   1 197   0
##   2  27 711
```

```
#training$smoker[1]
#class_nn[1]
```

10.1.3.1.3 Performance for training

```
levels(training$smoker)[class_nn[1]]
```

```
## [1] "2"
```

```
corrects=sum(levels(training$smoker)[class_nn]==training$smoker)
errors=sum(levels(training$smoker)[class_nn]!=training$smoker)
(perf.nn=corrects/(corrects+errors))
```

```
## [1] 0.9711
```

```r
print(paste('training accuracy: ',perf.nn*100,"%"))
```

```
## [1] "training accuracy:  97.1122994652406 %"
```

10.1.3.1.4 Prediction on test subset

```r
prediction_nn <- predict(nn,testing)
#prediction_nn
class_nn <- apply(prediction_nn, 1, which.max)
```

10.1.3.1.5 Confusion Matrix for testing

```r
table(testing$smoker, class_nn)
```

```
##    class_nn
##       1   2
##   1  75   2
##   2  13 313
```

```r
#training$smoker[1]
#class_nn[1]
```

10.1.3.1.6 Performance for testing

```r
levels(testing$smoker)[class_nn[1]]
```

```
## [1] "1"
```

```r
corrects=sum(levels(testing$smoker)[class_nn]==testing$smoker)
errors=sum(levels(testing$smoker)[class_nn]!=testing$smoker)
(perf.nn=corrects/(corrects+errors))
```

```
## [1] 0.9628
```

```r
print(paste('testing accuracy: ',perf.nn*100,"%"))
```

```
## [1] "testing accuracy:  96.2779156327543 %"
```

10.1.3.2 Deep NN - two hidden layers

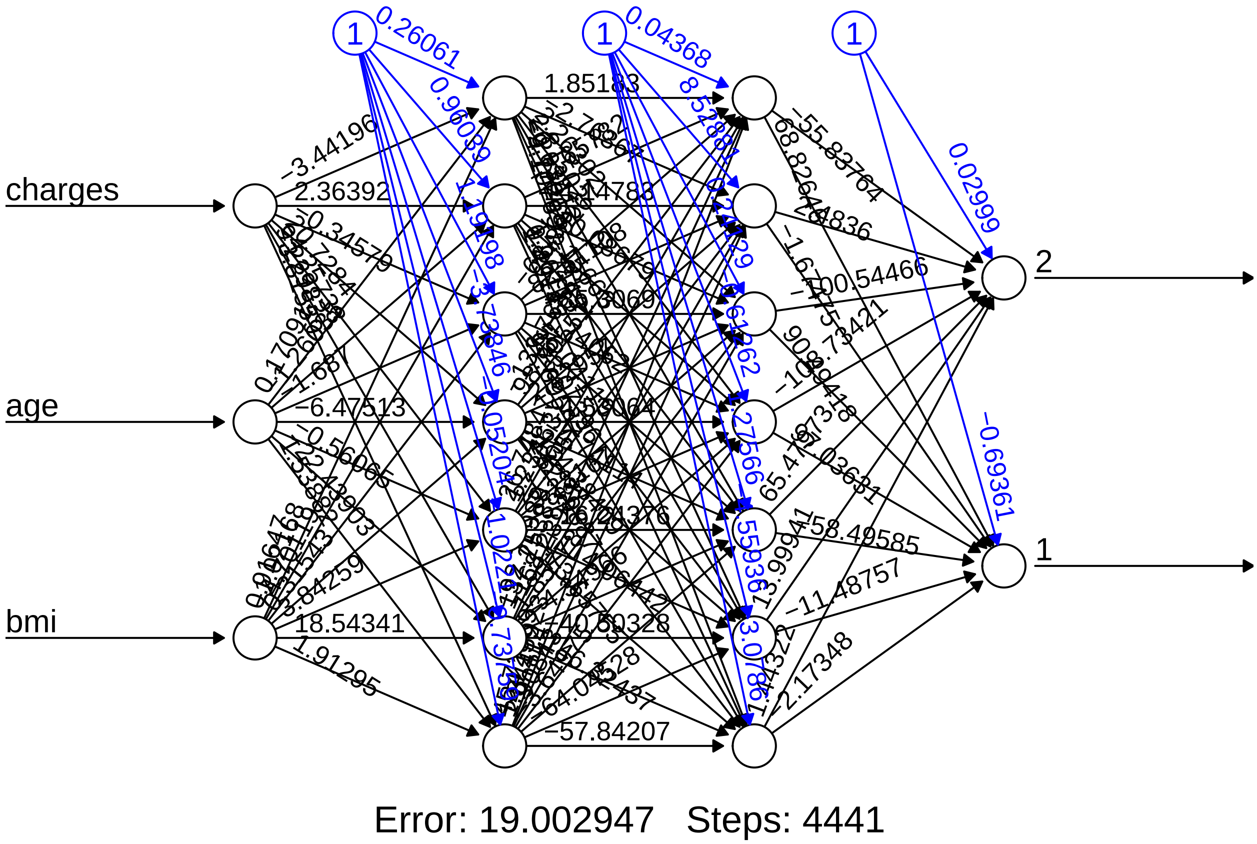
10.1.3.2.1 Creation of NN_2

```r
set.seed(444)
nn2 <- neuralnet(smoker~charges + age + bmi,
               data = training,
               hidden = c(7,7),
               threshold = 0.0025,
               lifesign = 'full',
               lifesign.step = 500,
               linear.output = FALSE)
```

```
## hidden: 7, 7    thresh: 0.0025    rep: 1/1    steps:     500 min thresh: 0.11825064758526
##                                                         1000 min thresh: 0.112617356849393
##                                                         1500 min thresh: 0.112617356849393
##                                                         2000 min thresh: 0.107500337348156
##                                                         2500 min thresh: 0.0500175621736521
##                                                         3000 min thresh: 0.0219775031983741
```

```
##                                                          3500 min thresh: 0.0129098584782895
##                                                          4000 min thresh: 0.00367818486599488
##                                                          4441 error: 19.00295 time: 7.47 secs
```

```r
#summary(nn)
#print(nn)
plot(nn2, rep='best')
```



Prediction on training subset

```r
prediction_nn2 <- predict(nn2,training)
#prediction_nn
class_nn <- apply(prediction_nn2, 1, which.max)
```

10.1.3.2.2 Confusion Matrix for training

```r
table(training$smoker, class_nn)
```

```
##    class_nn
##       1   2
##   1 197   0
##   2  19 719
```

```r
#training$smoker[1]
#class_nn[1]
```

10.1.3.2.3 Performance for

```r
levels(training$smoker)[class_nn[1]]
```

```
## [1] "2"
```

```r
corrects=sum(levels(training$smoker)[class_nn]==training$smoker)
errors=sum(levels(training$smoker)[class_nn]!=training$smoker)
(perf.nn=corrects/(corrects+errors))
```

```
## [1] 0.9797
```

```r
print(paste('training accuracy: ',perf.nn*100,"%"))
```

```
## [1] "training accuracy:  97.9679144385027 %"
```

10.1.3.2.4 Prediction on test subset

```r
prediction_nn2 <- predict(nn2,testing)
#prediction_nn
class_nn <- apply(prediction_nn2, 1, which.max)
```

10.1.3.2.5 Confusion Matrix for testing

```r
table(testing$smoker, class_nn)
```

```
##    class_nn
##       1   2
##   1  74   3
##   2  12 314
```

```r
#training$smoker[1]
#class_nn[1]
```

10.1.3.2.6 Performance for testing

```r
levels(testing$smoker)[class_nn[1]]
```

```
## [1] "1"
```

```r
corrects=sum(levels(testing$smoker)[class_nn]==testing$smoker)
errors=sum(levels(testing$smoker)[class_nn]!=testing$smoker)
(perf.nn=corrects/(corrects+errors))
```

```
## [1] 0.9628
```

```r
print(paste('testing accuracy: ',perf.nn*100,"%"))
```

```
## [1] "testing accuracy:  96.2779156327543 %"
```

10.1.3.3 Deep NN - three hidden layers

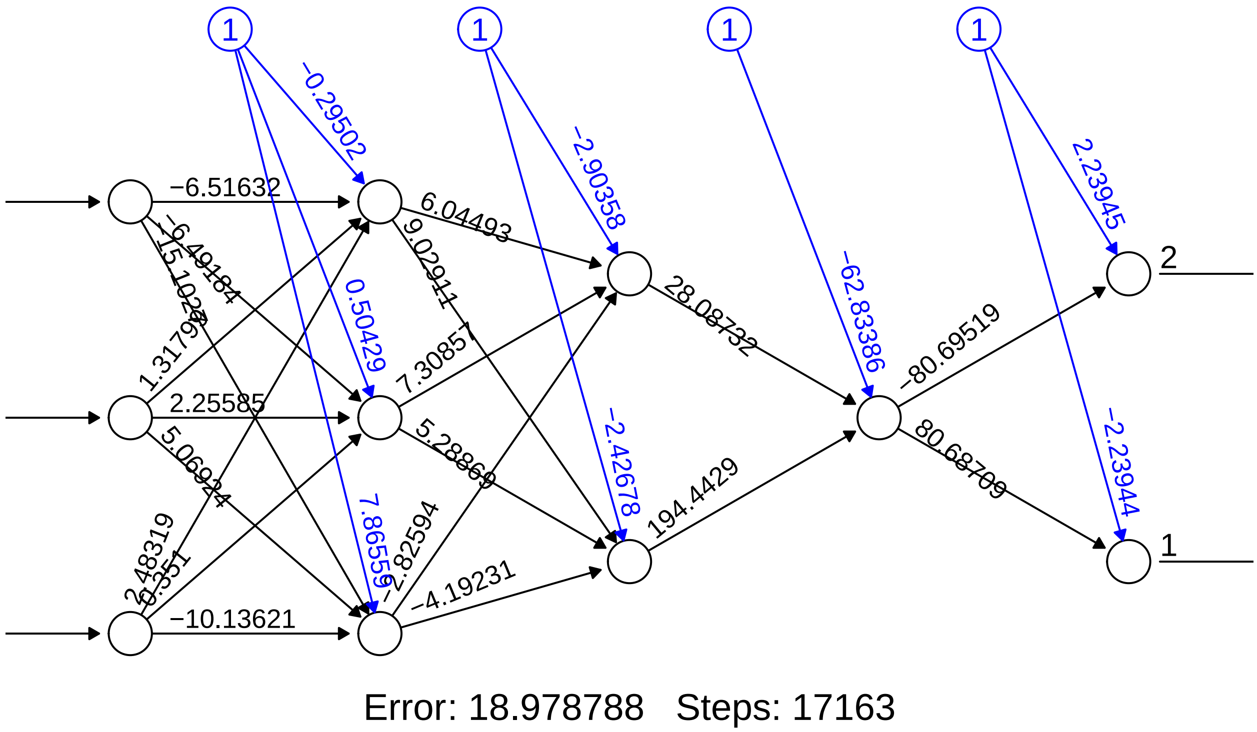
10.1.3.3.1 Creation of NN_3

```r
set.seed(444)
nn3 <- neuralnet(smoker~charges + age + bmi,
               data = training,
               hidden = c(3,2,1),
               threshold = 0.0025,
               lifesign = 'full',
```

```r
              lifesign.step = 500,
              linear.output = FALSE)
```

```
## hidden: 3, 2, 1    thresh: 0.0025    rep: 1/1    steps:     500  min thresh: 0.0938559883189038
##                                                            1000  min thresh: 0.0938559883189038
##                                                            1500  min thresh: 0.0938559883189038
##                                                            2000  min thresh: 0.0938559883189038
##                                                            2500  min thresh: 0.0779042619497504
##                                                            3000  min thresh: 0.0477729129630322
##                                                            3500  min thresh: 0.0477729129630322
##                                                            4000  min thresh: 0.0477729129630322
##                                                            4500  min thresh: 0.0477729129630322
##                                                            5000  min thresh: 0.0477729129630322
##                                                            5500  min thresh: 0.0477729129630322
##                                                            6000  min thresh: 0.0477729129630322
##                                                            6500  min thresh: 0.0477729129630322
##                                                            7000  min thresh: 0.0477729129630322
##                                                            7500  min thresh: 0.0401919383687264
##                                                            8000  min thresh: 0.0319441973901314
##                                                            8500  min thresh: 0.0271607588021149
##                                                            9000  min thresh: 0.0231670821049527
##                                                            9500  min thresh: 0.0231670821049527
##                                                           10000  min thresh: 0.0160246751435115
##                                                           10500  min thresh: 0.0143015470518307
##                                                           11000  min thresh: 0.00978614370100106
##                                                           11500  min thresh: 0.00978614370100106
##                                                           12000  min thresh: 0.00813136952436977
##                                                           12500  min thresh: 0.00670291691221209
##                                                           13000  min thresh: 0.00588460075826878
##                                                           13500  min thresh: 0.00568340157694122
##                                                           14000  min thresh: 0.00431292899953402
##                                                           14500  min thresh: 0.00405437903080155
##                                                           15000  min thresh: 0.00396128911212355
##                                                           15500  min thresh: 0.00308676494427534
##                                                           16000  min thresh: 0.00289723264713905
##                                                           16500  min thresh: 0.00274623242108184
##                                                           17000  min thresh: 0.00274623242108184
##                                                           17163  error: 18.97879 time: 16.13 secs
```

```r
#summary(nn)
#print(nn)
plot(nn3, rep='best')
```



Prediction on training subset

```
prediction_nn3 <- predict(nn3,training)
#prediction_nn
class_nn <- apply(prediction_nn3, 1, which.max)
```

10.1.3.3.2 Confusion Matrix for training

```
table(training$smoker, class_nn)
```

```
##    class_nn
##       1   2
##   1 197   0
##   2  21 717
```

```
#training$smoker[1]
#class_nn[1]
```

10.1.3.3.3 Performance for training

```
levels(training$smoker)[class_nn[1]]
```

```
## [1] "2"
```

```
corrects=sum(levels(training$smoker)[class_nn]==training$smoker)
errors=sum(levels(training$smoker)[class_nn]!=training$smoker)
(perf.nn=corrects/(corrects+errors))
```

```
## [1] 0.9775
```

```r
print(paste('training accuracy: ',perf.nn*100,"%"))
```

```
## [1] "training accuracy:  97.7540106951872 %"
```

10.1.3.3.4 Prediction on test subset

```r
prediction_nn3 <- predict(nn3,testing)
#prediction_nn
class_nn <- apply(prediction_nn3, 1, which.max)
```

10.1.3.3.5 Confusion Matrix for testing

```r
table(testing$smoker, class_nn)
```

```
##    class_nn
##       1   2
##   1  76   1
##   2   9 317
```

```r
#training$smoker[1]
#class_nn[1]
```

10.1.3.3.6 Performance for testing

```r
levels(testing$smoker)[class_nn[1]]
```

```
## [1] "1"
```

```r
corrects=sum(levels(testing$smoker)[class_nn]==testing$smoker)
errors=sum(levels(testing$smoker)[class_nn]!=testing$smoker)
(perf.nn=corrects/(corrects+errors))
```

```
## [1] 0.9752
```

```r
print(paste('testing accuracy: ',perf.nn*100,"%"))
```

```
## [1] "testing accuracy:  97.5186104218362 %"
```

10.2 Neural Networks using "h20"

10.2.1 Creation NN

10.2.2 Prediction on training

10.2.3 Performance on training

10.2.4 Prediction on testing

10.2.5 Performance on testing

10.3 Results and method comparison

Neural Network	Training Performance	Testing Performance
NN	97.11 %	96.28 %
NN2	97.97 %	96.28 %
NN3	97.75 %	97.52 %
h2o	96.68 %	96.52 %

In the above table you can see the last results for all created models which are nearly the same. Based on the sample results, there is no huge difference between using the neuralnet package or the h2o package. Moreover the parameters for the hidden layer were chosen by a trial and error approach. We tried to create different levels for the hidden parameter, to explore if the performance will increase with the given complexity. Since the results are not reflecting this assumption and all performance results are in the same range, all models can be used for an adequate classification, whether a person is a smoker or not.

11 Cross validation and discussion