Statistics | HW3

Problem 3: Regression techniques

```r
library("glmnet")
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```r
ceodata <- read.csv('ceo.csv')
ceodata$X <- NULL
head(ceodata)
```

```
##   salary totcomp tenure age  sales profits assets
## 1   3030    8138      7  61 161315    2956 257389
## 2   6050   14530      0  51 144416   22071 237545
## 3   3571    7433     11  63 139208    4430  49271
## 4   3300   13464      6  60 100697    6370  92630
## 5  10000   68285     18  63 100469    9296 355935
## 6   9375   42381      6  57  81667    6328  86100
```

Task 1

1a

The idea of the lasso regression is to penalize the magnitude of coefficients of features along with minimizing squared residuals.
In lasso regression coefficients are penalized by adding $\lambda\|\beta\|_1$ to optimization objective.
λ is the parameter which balances how much coefficients should be penalized (e.g. if $\lambda = 0$ we get simple linear regression).
Lasso regression performes not only regularization, but also feature selection, because It can set some coefficients to zero.
Lasso uses L1 norm which is not differentiable at all points (e.g. for 1 dimention It is not differentialbe at origin).
Therefore, gradient descent can't be used (because gradient is not defined), instead coordinate descent is used as an optimization algorithm.

Main benefit of lasso regression usage comparing to simle linear regression is when we have a huge number of features and want to get sparse solution (features with 0 coefficients can be ignored), because It's can be really hard to implement stepwise selection techniques in high dimensionality cases.

1b

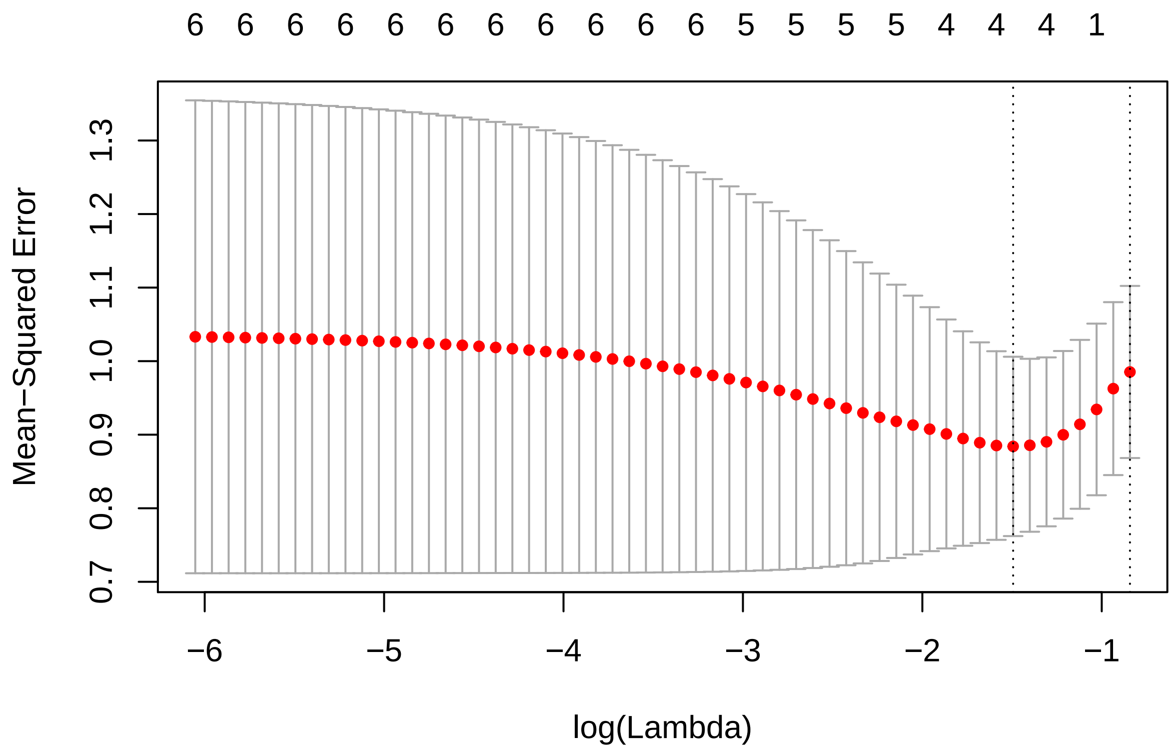
Lasso regression penalize by adding sum of absolute values of coefficients, that depend on the magnitude of each variable.
It is only reasonably to use this way of penalization when variables are scaled.

1c

```r
set.seed(100)
# scale and normalize data
normalized.ceodata <- data.frame(scale(ceodata))
head(normalized.ceodata)
```

```
##      salary      totcomp     tenure         age    sales   profits
## 1 0.5819707 -0.006399956 -0.1011859  0.66555607 9.262358  1.462226
## 2 2.3351688  0.196059181 -0.9500081 -0.80359733 8.217169 13.854141
## 3 0.8960370 -0.028730008  0.3838554  0.95938675 7.895059  2.417794
## 4 0.7387136  0.162294875 -0.2224462  0.51864073 5.513186  3.675462
## 5 4.6282589  1.898686047  1.2326776  0.95938675 5.499084  5.572335
## 6 4.2654282  1.078207090 -0.2224462  0.07789471 4.336196  3.648234
##      assets
## 1 3.5622969
## 2 3.2553947
## 3 0.3435979
## 4 1.0141769
## 5 5.0863838
## 6 0.9131856
```

```r
x <- model.matrix(salary~.+0, normalized.ceodata)
y <- normalized.ceodata$salary

# cross validation for glmnet, used to choose lambda
cv.lasso <- cv.glmnet(x, y, alpha=1);
plot(cv.lasso)
```




```
log(cv.lasso$lambda.min)
```

```
## [1] -1.49392
```

```
coef(cv.lasso, s = "lambda.min")
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -1.181642e-16
## totcomp      7.625390e-02
## tenure       .
## age          .
## sales        3.251541e-02
## profits      2.406526e-02
## assets       1.708017e-01
```

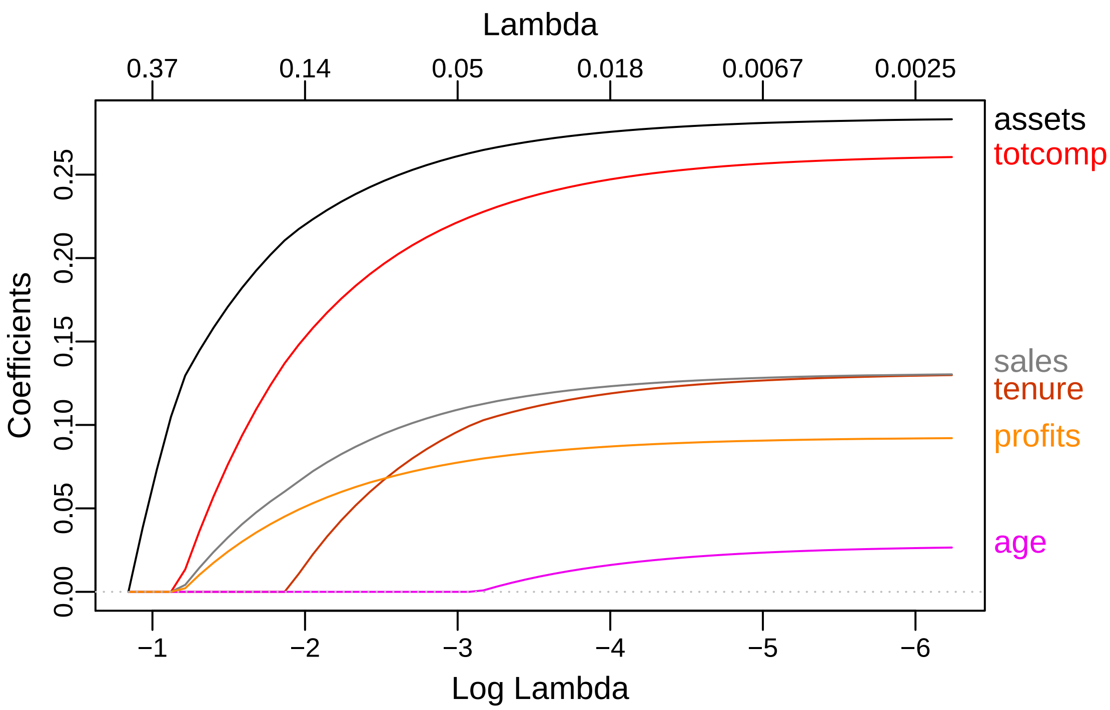
Using cross validation, we obtained most efficient lambda = 0.224491. As can be seen from plot, using this lambda leads to having 4 non-zero coefficients.

```
# plot estimated parameters as functions of lambda
library(plotmo)
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

```
lasso <- glmnet(x, y, alpha=1)
plot_glmnet(lasso)
```

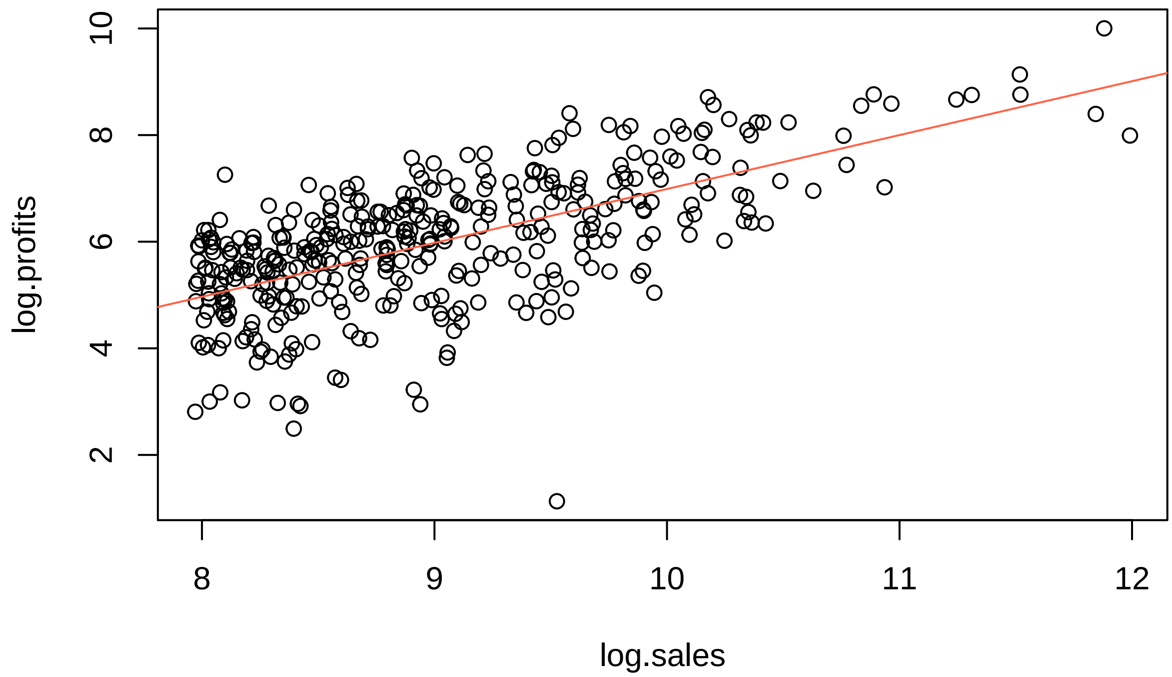


Plot of estimated parameters as functions of lambda

Task 2

2a

```
cleaned.data <- ceodata[ceodata$profits > 0, ]

log.profits <- log(cleaned.data$profits)
log.sales <- log(cleaned.data$sales)

lm.fitted <- lm(log.profits~log.sales)
plot(log.sales, log.profits)
abline(lm.fitted, col='tomato', lwd=1)
```

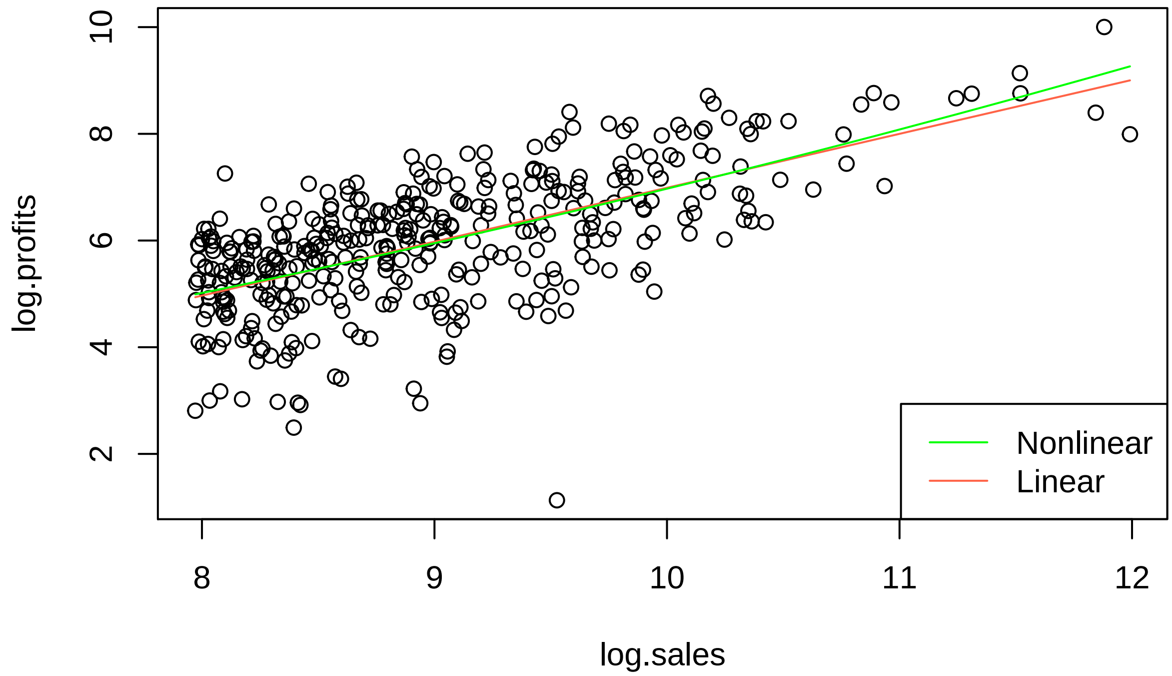


2b

```r
y <- log.profits
x <- log.sales

nls.fitted <- nls(y ~ b0 + b1 * x^b2, start=list(b0=0, b1=1, b2=2))
y_hat <- predict(nls.fitted)
summary(nls.fitted)
```

```
##
## Formula: y ~ b0 + b1 * x^b2
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## b0  1.11154    2.95752   0.376   0.7072
## b1  0.08904    0.25650   0.347   0.7287
## b2  1.81833    1.03079   1.764   0.0785 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9583 on 385 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 5.888e-08
```

```r
# plot linear vs nonlinear regression curves
plot(log.sales, log.profits)
lines(x, predict(lm.fitted), col='tomato', lwd=1)
lines(x, predict(nls.fitted), col="green", lwd=1)
legend("bottomright", legend=c("Nonlinear", "Linear"),col=c("green", "tomato"), lwd=1)
```



We can see on the plot that nonlinear regression curve coincides with linear one almost everywhere except top right part.

```r
loss.functions = function(x, x.hat)
{
  res = c(mean((x - x.hat) ^ 2),
          mean(abs(x - x.hat)),
          mean(abs((x - x.hat) / x )))
  names(res) = c("MSE","MAE", "MAPE")
  return(res);
}
# linear
loss.functions(y, predict(lm.fitted))
```

```
##       MSE       MAE      MAPE
## 0.9127472 0.7507421 0.1512142
```

```r
# nonlinear
loss.functions(y, predict(nls.fitted))
```

```
##       MSE       MAE      MAPE
## 0.9112678 0.7502607 0.1511123
```

MSE, MAE and MAPE errors for linear and nonlinear regression models.
We can see that errors for both models are pretty much the same, but nonlinear model is slightly better than linear.

2c

TODO dependent variable does not have constant variance

Task 3

3a

In Nadaraya–Watson regression function m is defined as: $\widehat{m}_h(x) = \frac{\sum_{i=1}^{n} K_h(x - x_i) y_i}{\sum_{j=1}^{n} K_h(x - x_j)}$,
where K_h is a kernel with a bandwidth h.
Bandwidth is a bias-variance tradeoff.
If we choose too large bandwidth, we get oversmoothed regression line and hence underfitting.
If we choose too small bandwidth, we get undersmoothed regression line and hence overfitting.

3b

```
library("readxl")
library("np")
```

```
## Nonparametric Kernel Methods for Mixed Datatypes (version 0.60-6)
## [vignette("np_faq",package="np") provides answers to frequently asked questions]
## [vignette("np",package="np") an overview]
## [vignette("entropy_np",package="np") an overview of entropy-based methods]
```

```
bw1 <- suppressWarnings(npregbw(log.profits ~ log.sales))
```

```
##
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 /
Multistart 1 of 1 |
Multistart 1 of 1 |
```

```
bw1
```

```
##
## Regression Data (388 observations, 1 variable(s)):
##
##                log.sales
## Bandwidth(s): 0.2374689
##
## Regression Type: Local-Constant
## Bandwidth Selection Method: Least Squares Cross-Validation
## Formula: log.profits ~ log.sales
## Bandwidth Type: Fixed
## Objective Function Value: 0.9317115 (achieved on multistart 1)
```

```
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

Optimal bandwidth selection method: Least Squares Cross-Validation.

```r
bw2 <- npregbw(log.profits ~ log.sales, bwmethod="cv.aic")
```

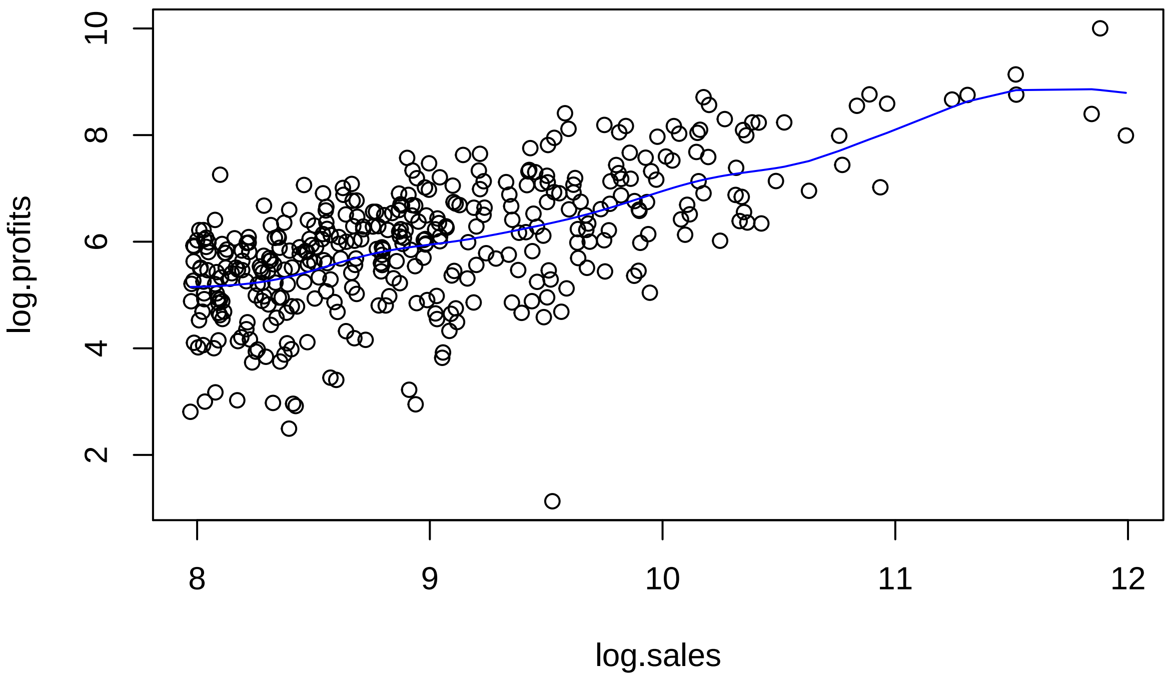
```
##
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 /
Multistart 1 of 1 |
Multistart 1 of 1 |
```

```r
bw2
```

```
##
## Regression Data (388 observations, 1 variable(s)):
##
##                log.sales
## Bandwidth(s): 0.2532013
##
## Regression Type: Local-Constant
## Bandwidth Selection Method: Expected Kullback-Leibler Cross-Validation
## Formula: log.profits ~ log.sales
## Bandwidth Type: Fixed
## Objective Function Value: 0.9392053 (achieved on multistart 1)
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

Optimal bandwidth selection method: Expected Kullback-Leibler Cross-Validation

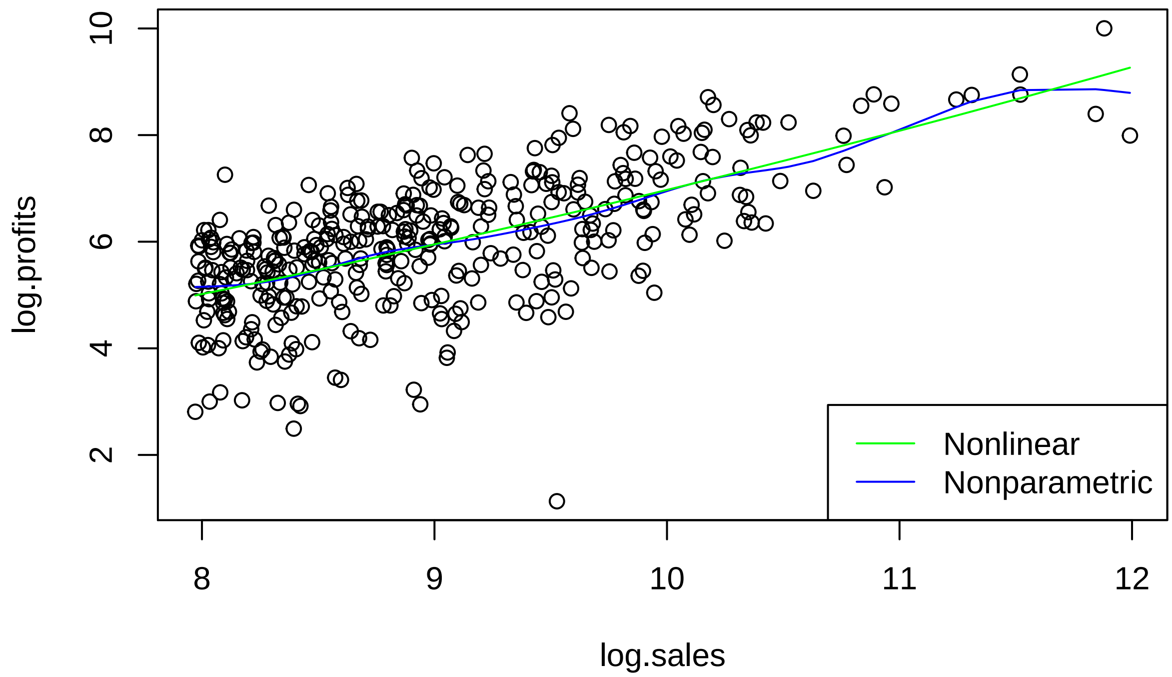
```r
non.parametric.fitted <- npreg(bws = bw1)
plot(log.sales, log.profits)
lines(log.sales, predict(non.parametric.fitted), col='blue', lwd=1)
```



Here we use optimal bandwidth found by Least Squares Cross-Validation method.

3c

```
plot(log.sales, log.profits)
# lines(log.sales, predict(lm.fitted), col='tomato', lwd=1)
lines(log.sales, predict(non.parametric.fitted), col='blue', lwd=1)
lines(log.sales, predict(nls.fitted), col="green", lwd=1)
legend("bottomright", legend=c("Nonlinear", "Nonparametric"),col=c("green", "blue"), lwd=1)
```



We can see on the plot that nonparametric regression curve fits data better that nonlinear one.

```
# nonlinear
loss.functions(log.profits, predict(nls.fitted))
```

```
##       MSE       MAE      MAPE
## 0.9112678 0.7502607 0.1511123
```

```
# nonparametric
loss.functions(log.profits, predict(non.parametric.fitted))
```

```
##       MSE       MAE      MAPE
## 0.9015954 0.7444075 0.1498249
```

MSE, MAE and MAPE errors for nonlinear and nonparametric regression models.
We can see that nonparametric model are better than nonlinear based on these errors.

Task 4

4a

```
new.ceodata <- ceodata
new.ceodata$salary = NULL
new.ceodata$high.salary <- ifelse(ceodata$salary > 2000, 1, 0)

logit.model <- suppressWarnings(glm(high.salary ~ ., family=binomial('logit'), data=new.ceodata))
summary(logit.model)
```

```
##
## Call:
## glm(formula = high.salary ~ ., family = binomial("logit"), data = new.ceodata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6385  -0.6966  -0.5629   0.6952   2.1301
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.523e+00  1.103e+00  -3.193  0.00141 **
## totcomp      7.792e-05  1.903e-05   4.095 4.23e-05 ***
## tenure       2.544e-02  1.482e-02   1.716  0.08611 .
## age          2.133e-02  1.967e-02   1.084  0.27823
## sales        4.099e-05  1.688e-05   2.428  0.01518 *
## profits      4.482e-04  1.762e-04   2.543  0.01099 *
## assets       6.295e-06  3.636e-06   1.731  0.08342 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 563.34  on 446  degrees of freedom
## Residual deviance: 434.73  on 440  degrees of freedom
## AIC: 448.73
##
## Number of Fisher Scoring iterations: 7
```

```
optimal.logit.model <- suppressWarnings(step(logit.model, direction = "both"))
```

```
## Start:  AIC=448.73
## high.salary ~ totcomp + tenure + age + sales + profits + assets
##
##           Df Deviance    AIC
## - age      1   435.91 447.91
## <none>         434.73 448.73
## - tenure   1   437.63 449.63
## - assets   1   438.17 450.17
## - sales    1   441.10 453.10
## - profits  1   441.85 453.85
## - totcomp  1   467.69 479.69
##
## Step:  AIC=447.91
## high.salary ~ totcomp + tenure + sales + profits + assets
##
##           Df Deviance    AIC
## <none>         435.91 447.91
## + age      1   434.73 448.73
## - assets   1   439.48 449.48
## - tenure   1   441.56 451.56
## - sales    1   442.33 452.33
## - profits  1   443.04 453.04
## - totcomp  1   468.33 478.33
```

```r
summary(optimal.logit.model)
```

```
##
## Call:
## glm(formula = high.salary ~ totcomp + tenure + sales + profits +
##     assets, family = binomial("logit"), data = new.ceodata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6921  -0.7030  -0.5587   0.6889   2.0360
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.367e+00  2.472e-01  -9.575  < 2e-16 ***
## totcomp      7.745e-05  1.907e-05   4.061 4.89e-05 ***
## tenure       3.234e-02  1.337e-02   2.418   0.0156 *
## sales        4.085e-05  1.677e-05   2.436   0.0149 *
## profits      4.457e-04  1.758e-04   2.535   0.0112 *
## assets       6.395e-06  3.644e-06   1.755   0.0793 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 563.34  on 446  degrees of freedom
## Residual deviance: 435.91  on 441  degrees of freedom
## AIC: 447.91
##
## Number of Fisher Scoring iterations: 7
```

Stepwise model selection using AIC as criterion.

```r
coef(optimal.logit.model)
```

```
##   (Intercept)       totcomp        tenure         sales       profits
## -2.366660e+00  7.744820e-05  3.233620e-02  4.084836e-05  4.457237e-04
##        assets
##  6.394663e-06
```

```r
exp(coef(optimal.logit.model))
```

```
## (Intercept)     totcomp      tenure       sales     profits      assets
##  0.09379345  1.00007745  1.03286470  1.00004085  1.00044582  1.00000639
```

Assuming fixed totcomp, tenure, profits and assests, if sales increse by 1, odds will increase by 1.00004085. So probability of getting high salary will increase a litte bit.

```r
set.seed(100)
random_5_rows <- ceodata[sample(nrow(new.ceodata), 5), ]
random_5_rows$high_salary_hat <- predict(optimal.logit.model, newdata = random_5_rows, type = "response
random_5_rows
```

```
##     salary totcomp tenure age   sales profits   assets high_salary_hat
## 138    800    2317      2  57 10553.0  -633.0  29374.0       0.1435916
## 115   1638    5075      9  49 12745.6  1486.9 165493.3       0.6362379
## 246   4280    4349      1  46  6087.1   509.8  93836.3       0.2846376
## 26    2900    3116      2  56 30219.0  1614.0  13465.0       0.4948100
```

```
## 208   3000    7418      2  59  7208.4   387.0   5577.7       0.2270817
```

```r
z <- -2.366660e+00 + 7.744820e-05 * 2317 + 3.233620e-02 * 2 + 4.084836e-05 * 10553.0    + 4.457237e-04 
p <- 1/(1 + exp(-z))
```

formula: (calculated for first item)
zi = $_0 + _1 X_{1i} +$ ůůů $+ _k X_{ki}$
$P(Y_i = 1|X_i) = \frac{1}{1+e^{z_i}}$ \$P = \$ 0.1435917

4d

```r
threshold <- 0.5
table(optimal.logit.model$y, fitted(optimal.logit.model)>threshold)
```

```
##
##     FALSE TRUE
##   0   284   18
##   1    83   62
```

Classification table.
From classification table we can see that only (284 + 62) / 447 = 77.4049217 % predicted correctly.
18 CEOs had low salary, but our model predicted high salary for them and 83 CEOs had high salary, but model predicted low salary for them.

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
conf.matr = confusionMatrix(ifelse(fitted(optimal.logit.model)>threshold,1,0), optimal.logit.model$y, p

# sensitivity specificity
conf.matr$byClass[c(1,2)]
```

```
## Sensitivity Specificity
##   0.4275862   0.9403974
```

Sensitivity and specifity.
Sensitivity - fraction of correctly classified 1 values (high salaries) among all CEOs which have high salary.
Specifity - fraction of correctly classified 0 values (low salaries) among all CEOs which have low salary.
So, only 42% of CEOs with high salary are classified as CEOs with high salary.
Our model is really good at predicting CEOs with low salary (94% of CEOs with low salary are classified as CEOs with low salary),
and really bad at predicting CEOs with high salary. So we need to find new optimal threashold.

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:glmnet':
##
##     auc
```

```
## The following objects are masked from 'package:stats':
##
```

```
##     cov, smooth, var
```

```r
library("verification")
```

```
## Loading required package: fields

## Loading required package: spam

## Loading required package: dotCall64

## Loading required package: grid

## Spam version 2.1-2 (2017-12-21) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: maps

##
## Attaching package: 'fields'

## The following object is masked from 'package:plotrix':
##
##     color.scale

## Loading required package: boot

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

## Loading required package: CircStats

## Loading required package: MASS

## Loading required package: dtw

## Loading required package: proxy

##
## Attaching package: 'proxy'

## The following object is masked from 'package:spam':
##
##     as.matrix

## The following object is masked from 'package:Matrix':
##
##     as.matrix

## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

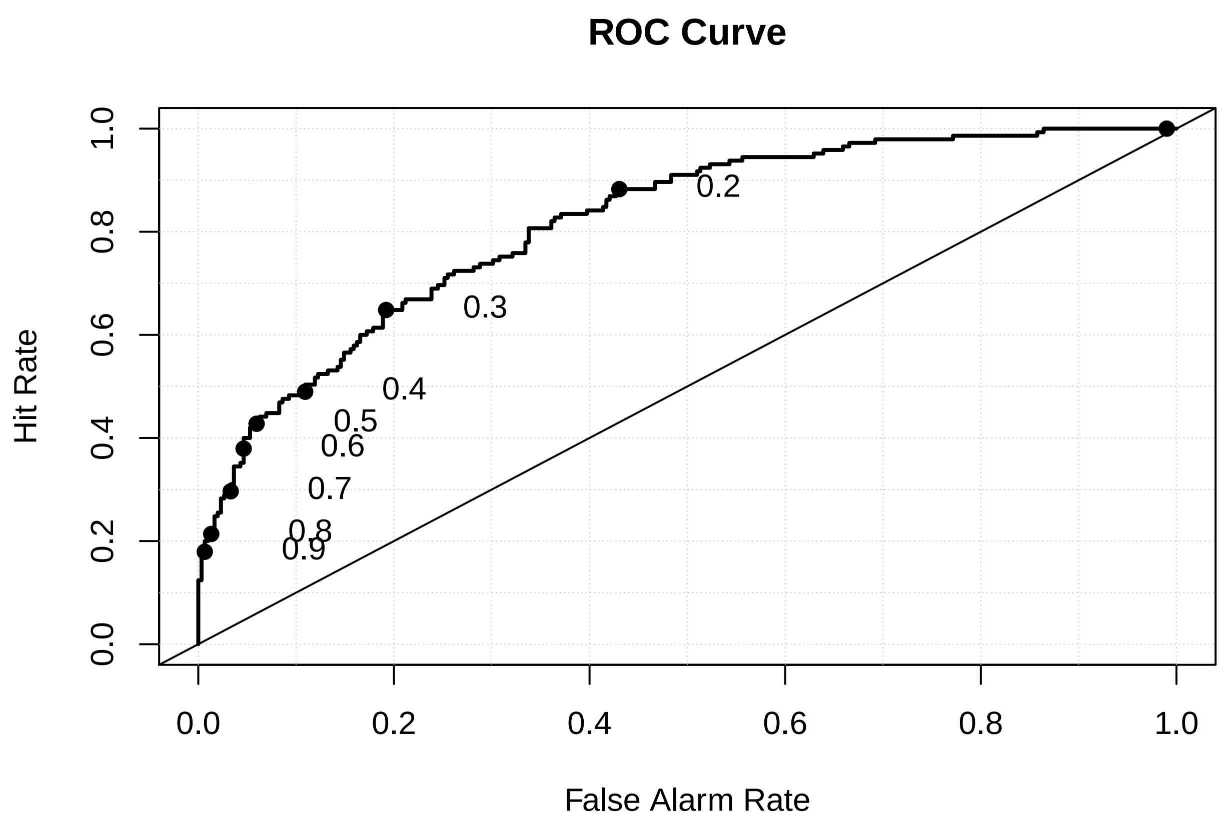
```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
## Loaded dtw v1.18-1. See ?dtw for help, citation("dtw") for use in publication.
```

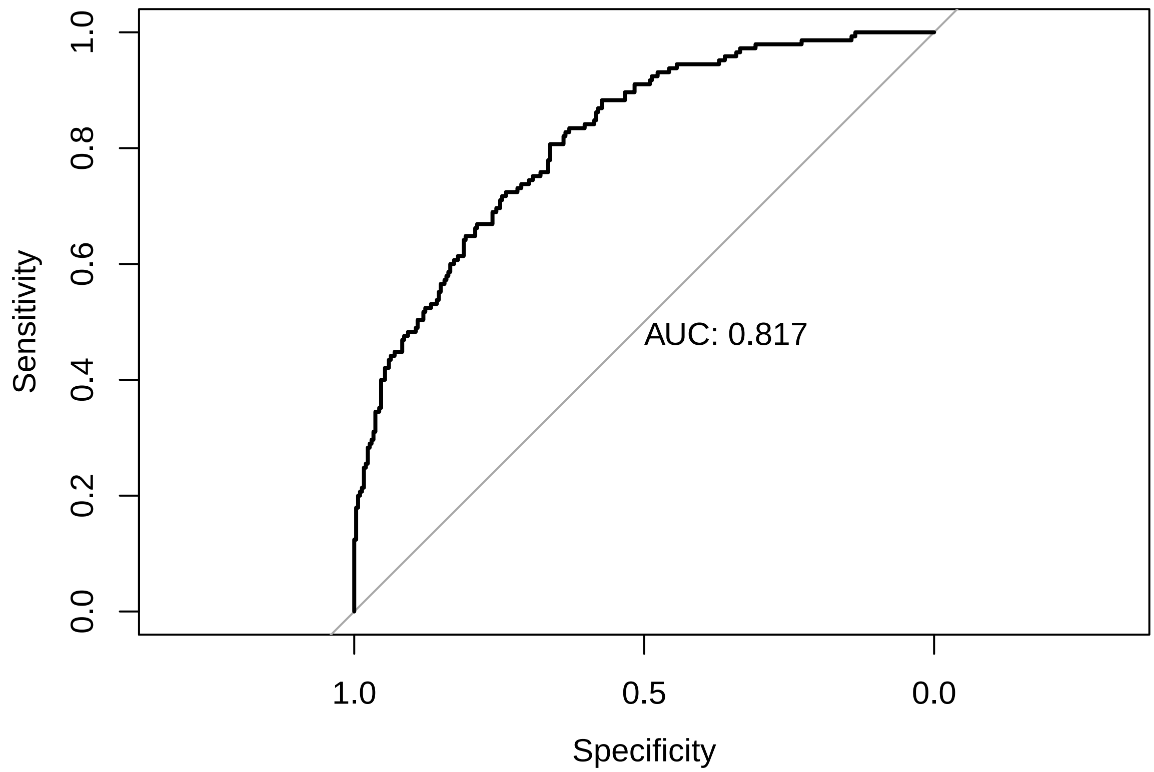
```
##
## Attaching package: 'verification'
```

```
## The following object is masked from 'package:pROC':
##
##     lines.roc
```

```r
roc.plot(optimal.logit.model$y, fitted(optimal.logit.model), )
```




```r
roc = roc(predictor = fitted(optimal.logit.model),response = optimal.logit.model$y)
plot(roc, print.auc=TRUE)
```




```
new.threshold = coords(roc, "best",ret="threshold")
```

Tow ROC curve plots from different libraries. Optimal treshold - 0.2240407.

```
table(optimal.logit.model$y, fitted(optimal.logit.model)>new.threshold)
```

```
##
##     FALSE TRUE
##   0   200  102
##   1    28  117
```

Recomputed classification table with optimal threshold.
From classification table we can see that only (200 + 117) / 447 = 70.917226 % predicted correctly.
102 CEOs had low salary, but our model predicted high salary for them and 28 CEOs had high salary, but model predicted low salary for them.

```
library(caret)
conf.matr = confusionMatrix(ifelse(fitted(optimal.logit.model)>new.threshold,1,0), optimal.logit.model$y

# sensitivity specificity
conf.matr$byClass[c(1,2)]
```

```
## Sensitivity Specificity
##   0.8068966   0.6622517
```

Sensitivity and specifity with new threshold.
Here we can see that or model is significantly better at predicting CEOs with high salary and a little bit worse at predicting CEOs with low salary.
So we obtained sensitivity and specifity both as high as possible, and improved our model.

Task 5

5a

Assume the first variable to be used for splitting is assets.
We want to find such splitting point s, that this expression $\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$ is minimal.
Where $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ are averages in R_1 and R_2. So the idea is to find such spilling point s that mimimizes kind of a variance in both of obtained rectangulars.

If we find first splitting point, we continue this procedure in obtained rectangulars recursively.

5b

```r
library("tree")
library("rpart")
library("RColorBrewer")
library("rattle")
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
rpart.ceo = rpart(salary ~ .,data=ceodata,control=rpart.control(cp = 0.001))
printcp(rpart.ceo)
```

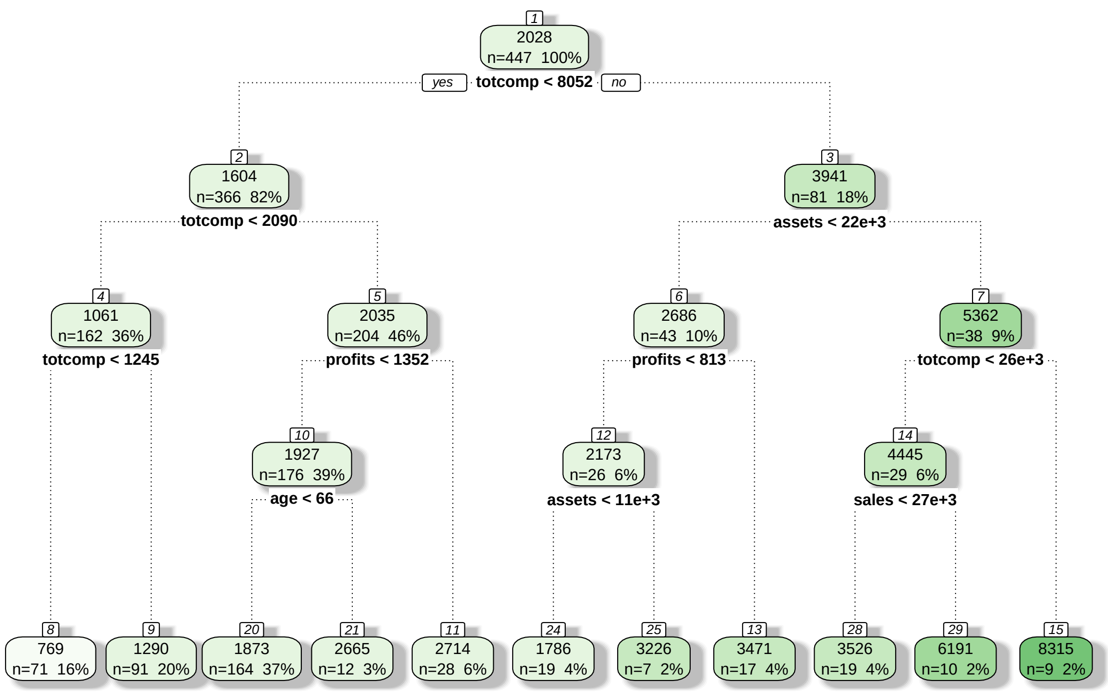
```
##
## Regression tree:
## rpart(formula = salary ~ ., data = ceodata, control = rpart.control(cp = 0.001))
##
## Variables actually used in tree construction:
## [1] age     assets  profits sales   tenure  totcomp
##
## Root node error: 1323386794/447 = 2960597
##
## n= 447
##
##           CP nsplit rel error  xerror    xstd
## 1  0.2738266      0   1.00000 1.00306 0.19977
## 2  0.1091070      1   0.72617 0.82287 0.14599
## 3  0.0777412      2   0.61707 0.83967 0.15759
## 4  0.0646524      3   0.53933 0.81310 0.15527
## 5  0.0351651      4   0.47467 0.78862 0.16152
## 6  0.0130789      5   0.43951 0.77735 0.16232
## 7  0.0113130      6   0.42643 0.75839 0.14798
## 8  0.0081763      7   0.41512 0.75394 0.14816
## 9  0.0080167      8   0.40694 0.74200 0.14746
## 10 0.0052976      9   0.39892 0.72163 0.14221
## 11 0.0032733     10   0.39363 0.72576 0.14231
## 12 0.0029769     11   0.39035 0.73295 0.14246
## 13 0.0022593     12   0.38737 0.73469 0.14247
## 14 0.0017931     13   0.38512 0.74246 0.14236
## 15 0.0016171     15   0.38153 0.73917 0.14104
## 16 0.0016099     17   0.37829 0.74226 0.14105
```

```
## 17 0.0012678     20   0.37346 0.74123 0.14104
## 18 0.0012449     21   0.37220 0.74276 0.14105
## 19 0.0010000     22   0.37095 0.74098 0.14105
```

Summary of regression tree. This tree has 19 splittings. Now, let's prune It to have at most 10 splits. For this step I use complexity parameter equals to 0.001 in order to not have too many splits.
As we can see from table, in order to get 10 splits, we need to use complexity parameter equals to 0.0052976.

```r
rpart.ceo.prunned = prune(rpart.ceo, cp = 0.0052976)
fancyRpartPlot(rpart.ceo.prunned)
```

```
## Warning in title(main = main, sub = sub): conversion failure on 'Rattle
## 2018-  -27 15:14:10 irko' in 'mbcsToSbcs': dot substituted for <d1>

## Warning in title(main = main, sub = sub): conversion failure on 'Rattle
## 2018-  -27 15:14:10 irko' in 'mbcsToSbcs': dot substituted for <81>

## Warning in title(main = main, sub = sub): conversion failure on 'Rattle
## 2018-  -27 15:14:10 irko' in 'mbcsToSbcs': dot substituted for <d1>

## Warning in title(main = main, sub = sub): conversion failure on 'Rattle
## 2018-  -27 15:14:10 irko' in 'mbcsToSbcs': dot substituted for <96>

## Warning in title(main = main, sub = sub): conversion failure on 'Rattle
## 2018-  -27 15:14:10 irko' in 'mbcsToSbcs': dot substituted for <d1>

## Warning in title(main = main, sub = sub): conversion failure on 'Rattle
## 2018-  -27 15:14:10 irko' in 'mbcsToSbcs': dot substituted for <87>
```

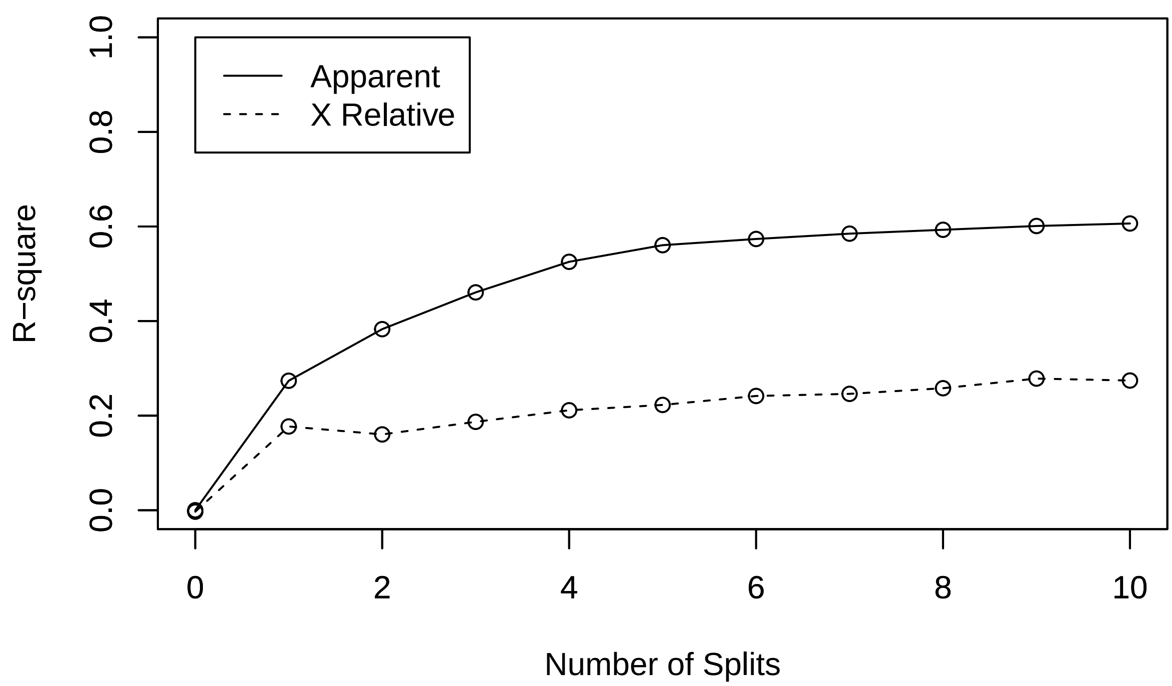


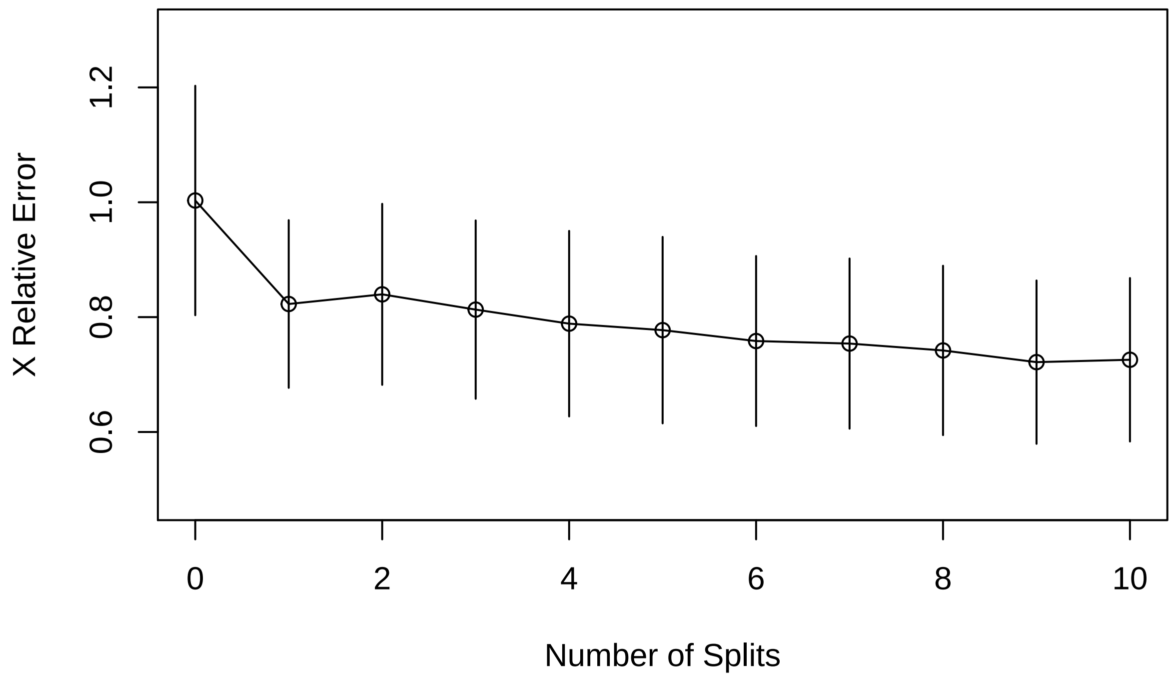
Rattle 2018−......−27 15:14:10 irko

Here we can see prunned tree with 10 splits.
Intensity of green color in nodes means value of salary: the more intense node color, the larger salary.

```r
rsq.rpart(rpart.ceo.prunned)
```

```
##
## Regression tree:
## rpart(formula = salary ~ ., data = ceodata, control = rpart.control(cp = 0.001))
##
## Variables actually used in tree construction:
## [1] age     assets  profits sales   totcomp
##
## Root node error: 1323386794/447 = 2960597
##
## n= 447
##
##           CP nsplit rel error  xerror    xstd
## 1  0.2738266      0   1.00000 1.00306 0.19977
## 2  0.1091070      1   0.72617 0.82287 0.14599
## 3  0.0777412      2   0.61707 0.83967 0.15759
## 4  0.0646524      3   0.53933 0.81310 0.15527
## 5  0.0351651      4   0.47467 0.78862 0.16152
## 6  0.0130789      5   0.43951 0.77735 0.16232
## 7  0.0113130      6   0.42643 0.75839 0.14798
## 8  0.0081763      7   0.41512 0.75394 0.14816
## 9  0.0080167      8   0.40694 0.74200 0.14746
## 10 0.0052976      9   0.39892 0.72163 0.14221
## 11 0.0052976     10   0.39363 0.72576 0.14231
```





Plolt of approximate R-squared and relative error for 10 spilts.
We can see that after first split relative error drops a lot, after next splits relative error does not change that much.

```r
rpart.ceo.prunned$method
```

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

I used rpart function for prunning. It used ANOVA (Analysis of variances) method.
from documentation: The splitting criteria is $SS_T - (SS_L + SS_R)$, where $SS_T = \sum (y_i - \bar{y})^2$ is the sum of squares for the node, and SS_R, SS_L are the sums of squares for the right and left son, respectively. This is equivalent to choosing the split to maximize the between-groups sum-of-squares in a simple analysis of variance.

5c

Tree pruning:

$$R_\alpha(T) = \frac{1}{\sum_i (y_i - \bar{y})^2} \sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

where $|T|$ is the number of terminal nodes in a tree and α is the complexity parameter.

Key properties for CARTs that guarantees that prunning using a single complexity parameter works.
1) For given α it is possible to determine the tree $T(\alpha)$ with the smallest $R_\alpha(T)$ unique.
2) If $\alpha > \beta$then $T(\alpha) = T(\beta)$ or $T(\alpha)$ is a strict subtree of $T(\beta)$.
The sequence of trees T_0 (no splits) to T_m (m splits) uniquely determines the sequence of possible α's.

So, on every splitting step, we define alpha (this can be observed above in summary tables).
And every prunned tree is a subtree of original one.
That’s why using a single complexity parameter for pruning works.