Notebook 2: Supervised learning (regression) part 2

You know the routine now :)

```
rm(list = ls(all = TRUE)) #clean up workspace
path <- "" #type the full path between the quotes and use
           #forward slashes ('/') to separate directories.
# Example:
path <- "C:/Users/koenn/OneDrive/School/DAV/Notebook 2"
options(digits = 3)
```

Reload our previous session

```
load(file.path(path,"Notebook 2 part1 image.RData"))
```

Thus far, we have used a separate validation set while trying out models.

If there is not enough data to provide this separate validation set, validation can also be done on the training set. Three methods for this are k-fold cross validation, bootstrapping and leave-one-out cross validation.

First we look at k-fold cross validation, setting k to 10 and optimize the $L1$-penalty again

```
set.seed(37729)
k <- 10 #set the number of folds to 10
msevec <- rep(0, 5) #make a vector to store mean mse per model
msevec.cv <- rep(0, k) #make a vector to store mse per fold
msetrainvec <- numeric(5) # another way to initialise a vector of 5 zeros
msetrainvec.cv <- numeric(k) # for storing mse train per fold

#create k folds
folds <- ceiling(runif(nrow(train), min = 0, max = k))
table(folds)
```

```
## folds
##   1   2   3   4   5   6   7   8   9  10
## 138 120 108 126 127 108 103 112 128 146
```

```
lambda <- c(0.001, 0.01, 0.1, 1, 10)  #lambda vector we want to try out

library(glmnet)
library(ggplot2)
library(tidyr)

for (i in 1:5) {
  #this is the tuning parameter loop
  paste("fitting with lambda = ", lambda[i])
  for (j in 1:k) {
  #this is the cross validation loop
  mod <- glmnet( x = x.train[folds!=j,],  #train on all but fold j
                 y = train$y[folds!=j], lambda = lambda[i], alpha = 0,
                 family = "gaussian", standardize = TRUE )
  pred <- predict(mod, newx = x.train[folds==j, ]) #validate on fold j
  msevec.cv[j] <- mse(train$y[folds==j], pred)

  predtrain <- predict(mod, x.train[folds != j, ]) # use training data
msetrainvec.cv[j] <- mse(train$y[folds != j], predtrain)
```

```
  }
  msevec[i] <- mean(msevec.cv) #take the average of the mse over folds
                               #as the cross validation error
  msetrainvec[i] <- mean(msetrainvec.cv)
}
cat("Best lambda is ", lambda[which.min(msevec)],
    "with average mse = ", min(msevec),
    "and average training mse = ", msetrainvec[which.min(msevec)])
```

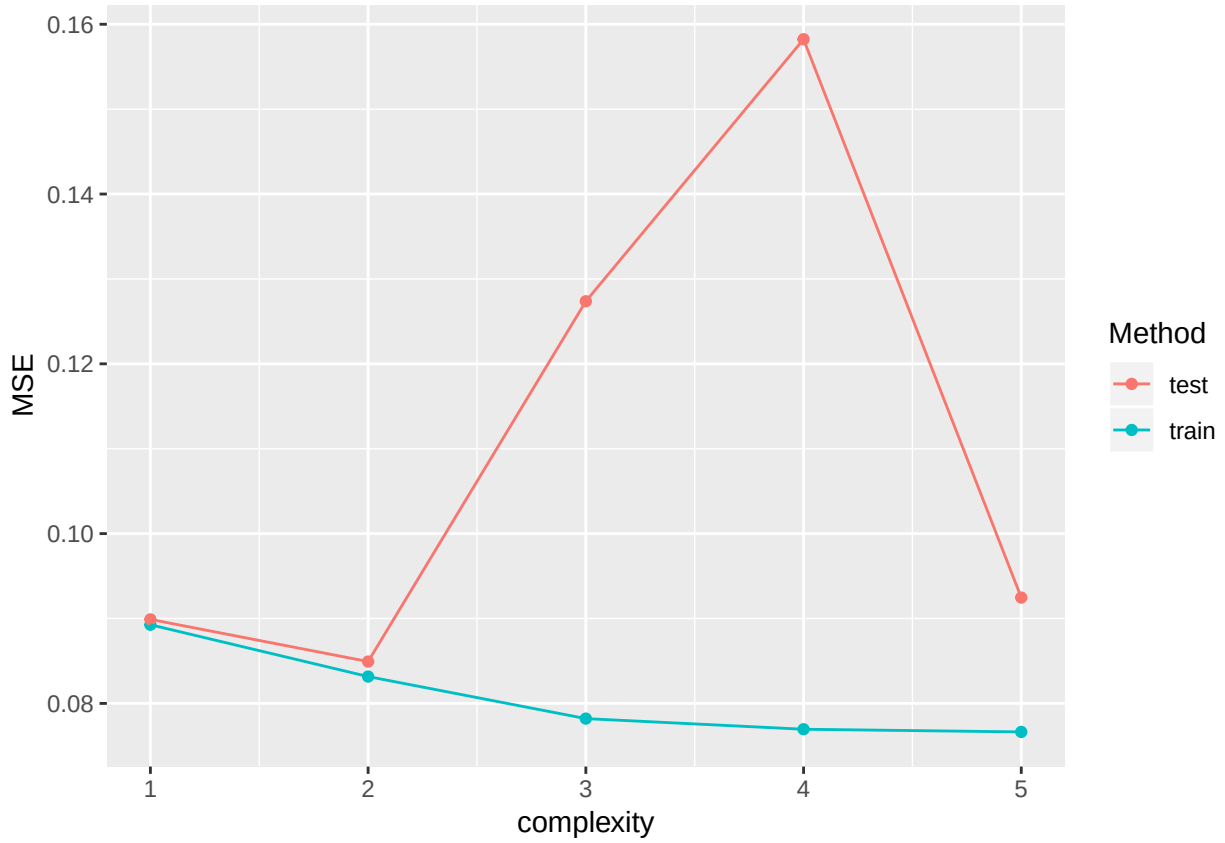
```
## Best lambda is  1 with average mse =  0.0849 and average training mse =  0.0832
```

```
data.frame(train = msetrainvec,
           test = msevec,
           # here, order() is a trick for scale: high lambda = low complexity.
           # The result here is (5, 4, 3, 2, 1)
           complexity = order(lambda, decreasing = TRUE)) %>%
  gather(key = Method, value = MSE, train, test) -> # tip: arrow works both ways
  ggdf

print(ggdf)
```

```
##    complexity Method    MSE
## 1           5  train 0.0766
## 2           4  train 0.0770
## 3           3  train 0.0782
## 4           2  train 0.0832
## 5           1  train 0.0893
## 6           5   test 0.0925
## 7           4   test 0.1582
## 8           3   test 0.1274
## 9           2   test 0.0849
## 10          1   test 0.0899
```

```
ggplot(ggdf, aes(x = complexity, y = MSE, colour = Method)) +
geom_point() + geom_line()
```



Cross validation has found a different value of the $L1$-penalty than in the train/validation setting.

Leave-one-out cross validation is obtained by setting the number of folds to the number of observations. This requires fitting as many models as observations in the training data!

```
set.seed(37729)
k <- nrow(x.train) #set the number of folds to n
msevec <- rep(0, 5)
msevec.cv <- rep(0, k)

#as many folds as observations
folds <- 1:nrow(x.train)

lambda <- c(0.001, 0.01, 0.1, 1, 10)  #lambda vector we want to try out

library(glmnet)

for (i in 1:5) {
  #this is the tuning parameter loop
  paste("fitting with lambda = ", lambda[i])
  for (j in 1:k) {
  #this is the cross validation loop
  mod <- glmnet( x = x.train[folds!=j,],  #train on all but fold j
                 y = train$y[folds!=j], lambda = lambda[i], alpha = 0, family = "gaussian", standardize
  pred <- predict(mod, newx = x.train[folds==j, ,drop = FALSE]) #validate on fold j
  msevec.cv[j] <- mse(train$y[folds==j], pred)
  }
```

```
  msevec[i] <- mean(msevec.cv)
}
cat("Best lambda is ", lambda[which.min(msevec)], "with average mse = ", min(msevec))
```

```
## Best lambda is  0.001 with average mse =  0.0802
```

Leave-one-out cross validation selects the same penalty as the train/val/test of the previous part of notebook 2 in this instance.

Another way to use the training data to obtain the generalization error is the bootstrap. This involves repeated random samples from the training data. The average out-of-sample error (on the fraction that not sampled in each bootstrap)is used as generalization error.

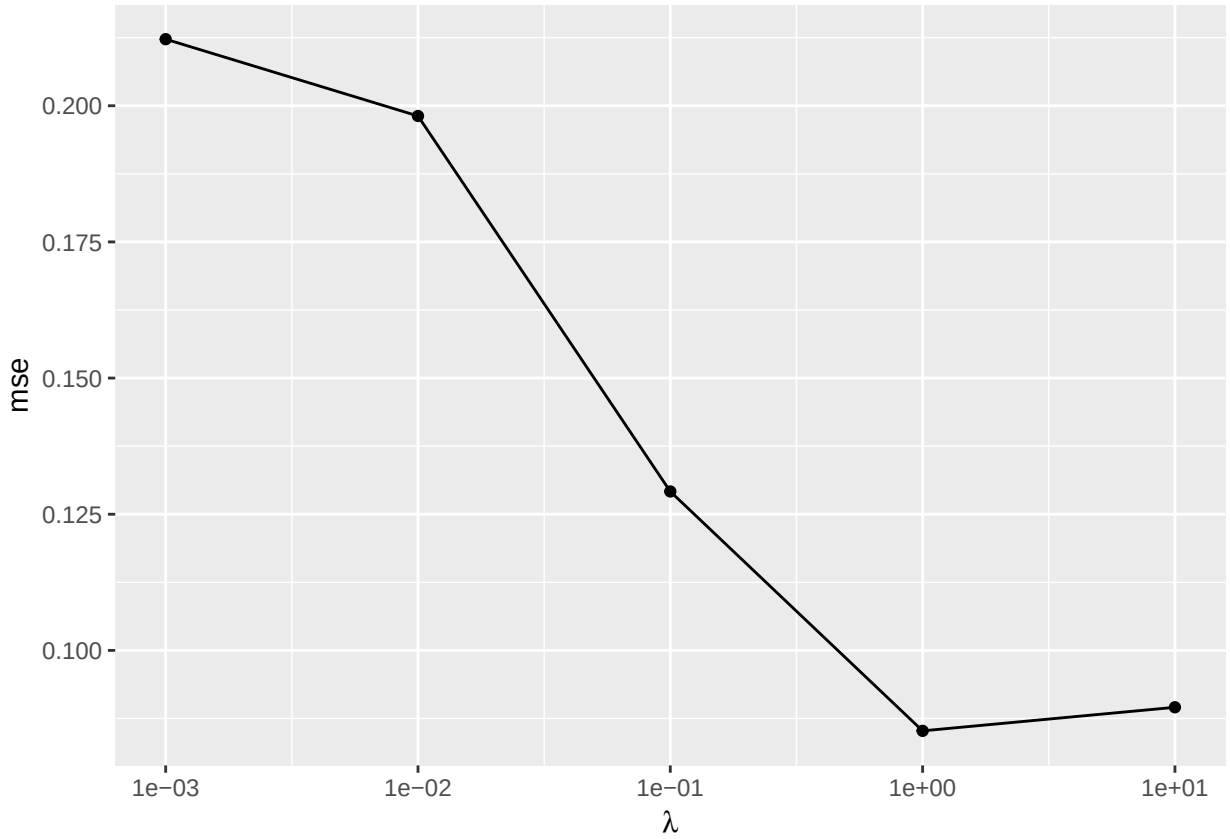
```
set.seed(37729)
B <- 100 #set the number of bootstrap samples to 100
b <- nrow(train) #bootstrap sample size, usually as large as the training data
msevec <- rep(0, 5)
msevec.bs <- rep(0, B)

lambda <- c(0.001, 0.01, 0.1, 1, 10)  #lambda vector we want to try out

library(glmnet)

for (i in 1:5) {
  #this is the tuning parameter loop
  paste("fitting with lambda = ", lambda[i])
  for (j in 1:B) {
  #this is the bootstrap loop
  samp <- sample(1:nrow(train), size = b, replace = TRUE) #sample *with* replacement
  mod <- glmnet( x = x.train[samp,],  #train on all but fold j
                 y = train$y[samp], lambda = lambda[i], alpha = 0, family = "gaussian", standardize = TR
  pred <- predict(mod, newx = x.train[-samp,]) #validate out of sample (out of bag)
  msevec.bs[j] <- mse(train$y[-samp], pred)
  }
  #take the average of the mse over bootstrap samples as the cross validation error
  msevec[i] <- mean(msevec.bs)
}

library(ggplot2)
ggplot() +  labs(x = expression(lambda), y = "mse") + scale_x_log10(breaks = lambda) +
  geom_line(aes(lambda, msevec)) + geom_point(aes(lambda,msevec))
```



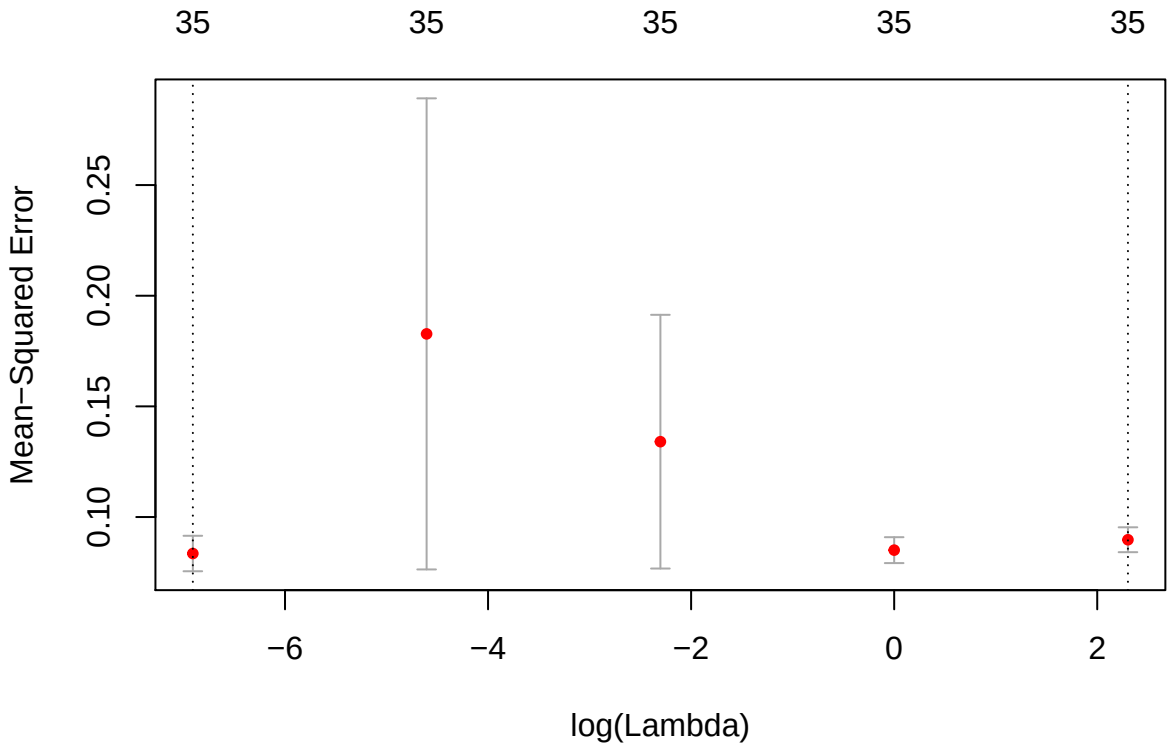

```
cat("Best lambda is ", lambda[which.min(msevec)], "with average mse = ", min(msevec))
```

```
## Best lambda is  1 with average mse =  0.0852
```

The bootstrap selects yet another value of λ than the previous methods of estimating generalization error.

In the above three implementations, for didactic purposes we have used pure R code to do the validation. However, many R libraries have cross validation or bootstrap built in. For instance, glmnet has its own (optimized) cross validation routine, as well as diagnostic plots, hiding all technical details.

```
set.seed(4728)
  mod <- cv.glmnet( x = x.train,
                    y = train$y, lambda = c(0.001, 0.01, 0.1, 1, 10),
                    alpha = 0, family = "gaussian", standardize = TRUE,
                    nfolds = 10)
plot(mod)
```

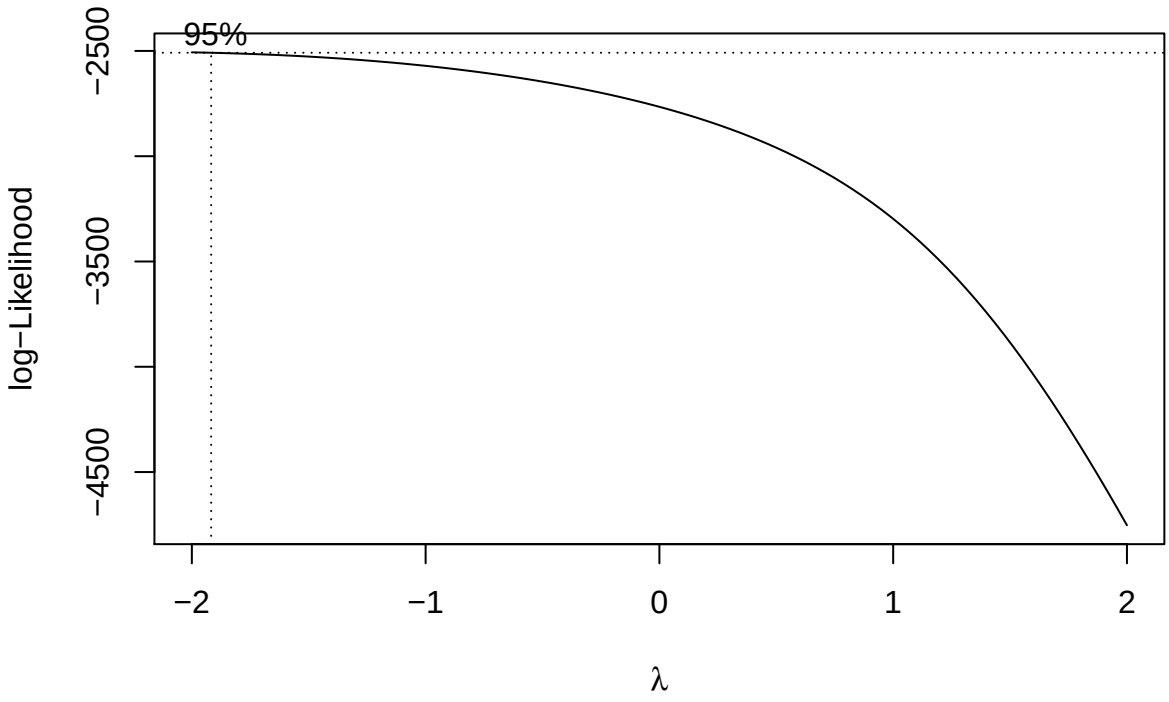



```
cat('Best lambda =',mod$lambda.min)
```

```
## Best lambda = 0.001
```

As a final note, for linear models, the Box-Cox family of transformations can also be used to estimate an optimal transformation (in terms of the maximum likelihood) for the dependent variable, using a single parameter λ (not to be confused with the L1-/L2-lambda!) in the form $\frac{(y^\lambda - 1)}{\lambda}$. A λ of 0 corresponds to The log transformation.

```
boxcox.profile <- MASS::boxcox(lm((SUMCASES / EXPOSURE * 365.25) + 1 ~
SEX + COUNTRYBIRTH + AGE + I(AGE^2) +  AGE1STCASE +  CRIMETYPE +
log(PREVCASES + 1), data = train))
```




```
optlambda <- boxcox.profile$x[which.max(boxcox.profile$y)]
print(paste('optimal lambda = ', optlambda))
```

```
## [1] "optimal lambda =  -2"
```

Box-Cox suggests using the transformation $y = \frac{1}{y^2}$.

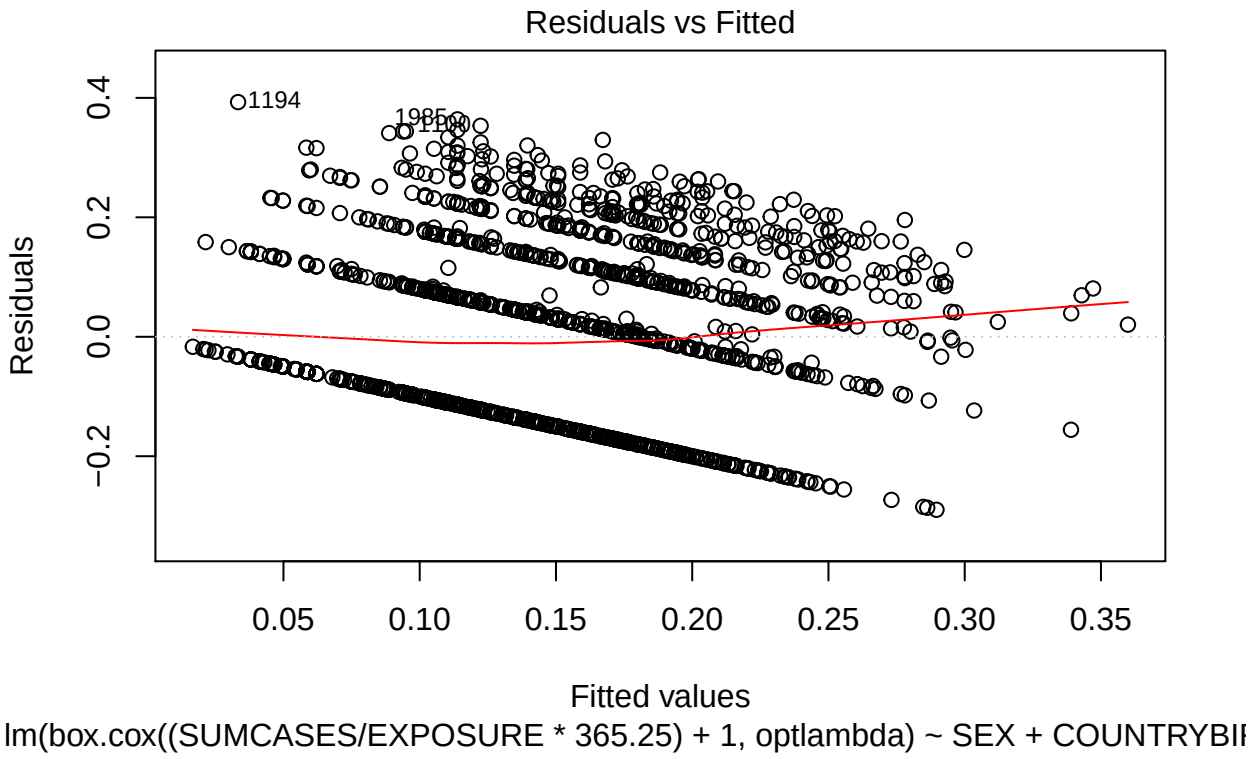
Now, we apply the model using this transformation and test its performance.

Conveniently, we can apply the transformation inside the formula specification.

Model fit can be assessed visually by looking at residual plots. The plot of the residuals versus the fitted values should display a straight line and evenly distributed residuals.

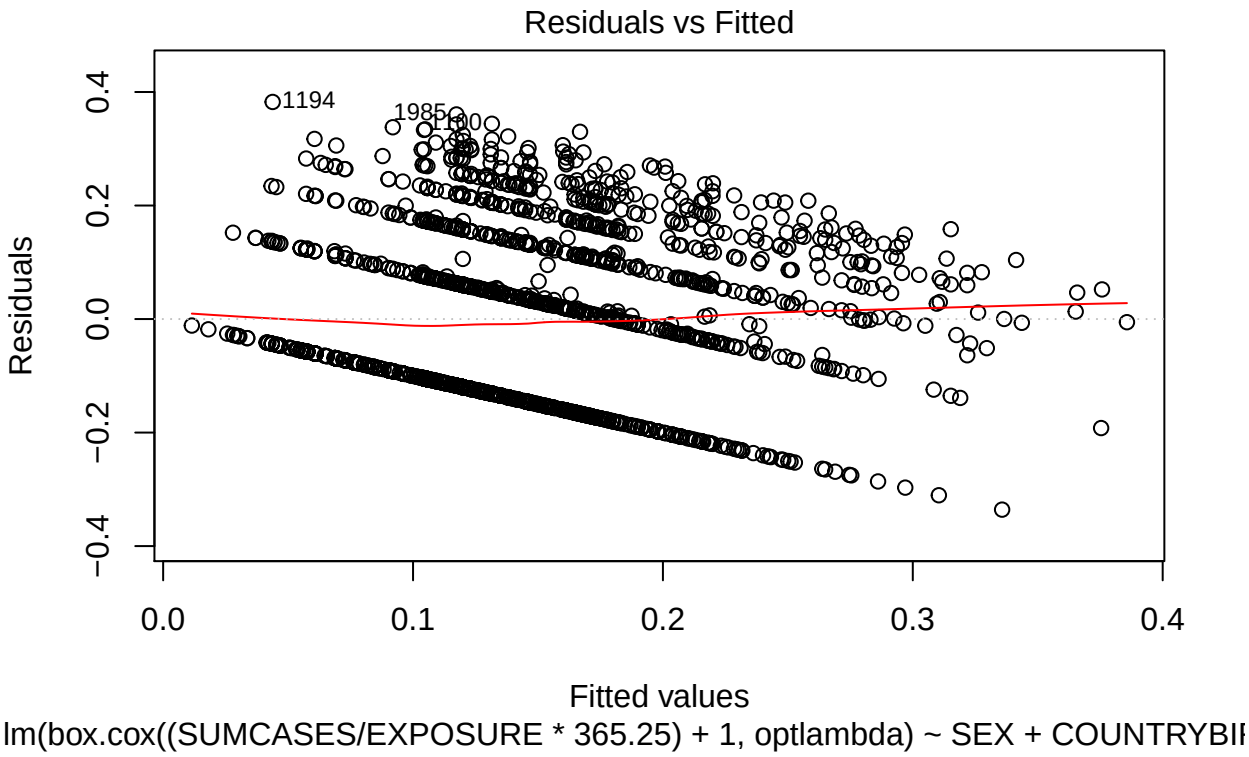
```
box.cox <- function(x, lambda){
  # applies the box-cox family of transformations
  # special values are:
  # -1 -> 1/x
  # 0 -> logarithm
  # 0.5 -> square root
  # 2 -> square
  if (lambda==0) {
    xprime <- log(x)
  } else {
  xprime <- (x^lambda - 1) / lambda
  }
}
#linear regression of (transformed) AGE
linear.model <- lm(box.cox((SUMCASES / EXPOSURE * 365.25) + 1, optlambda) ~
```

```
                    SEX + COUNTRYBIRTH + AGE + I(AGE^2) +  AGE1STCASE +  CRIMETYPE +
log(PREVCASES + 1), data = dat)
plot(linear.model, which = 1)
```



Unfortunately, a clear pattern can be discerned. Maybe a nonlinear model reveals deviations from linearity. Continuous variables can also be nonparametrically modeled non-linearly by using B-splines. Here we set 6 *df* B-splines.

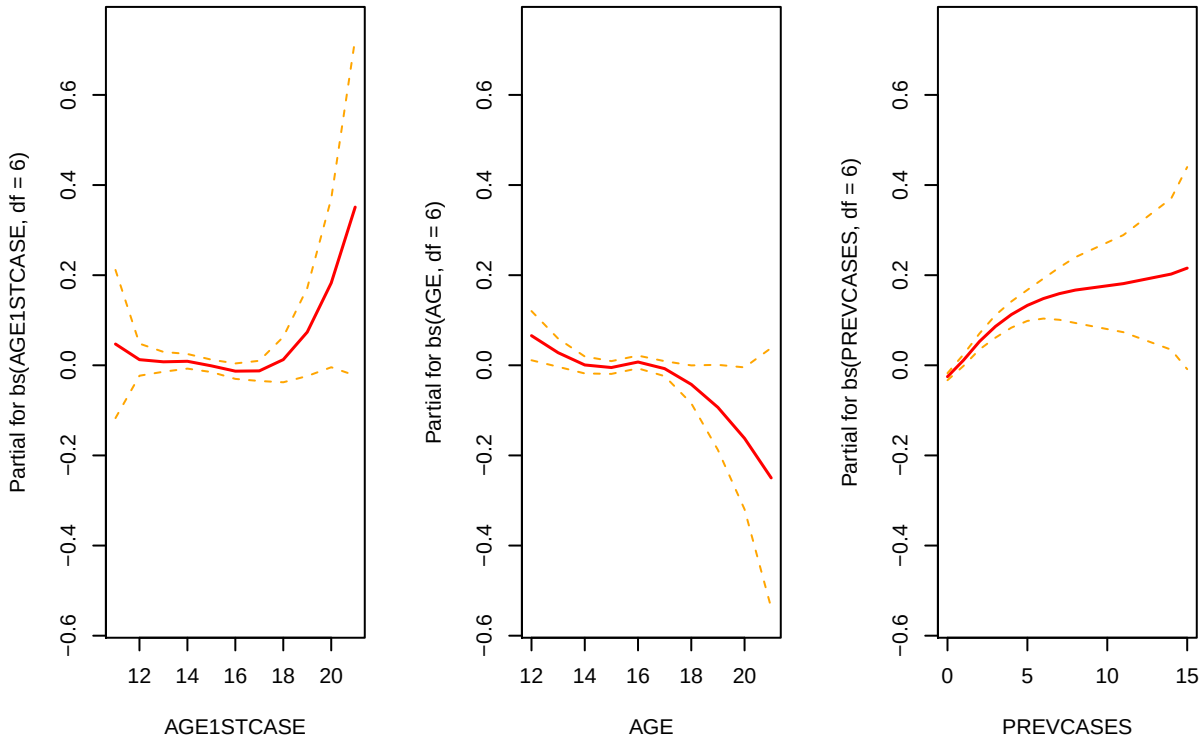
```
library(splines)
nonlinear.model <- lm(box.cox((SUMCASES / EXPOSURE * 365.25) + 1, optlambda) ~
                    SEX + COUNTRYBIRTH + bs(AGE, df = 6) +  bs(AGE1STCASE, df = 6) +  CRIMETYPE +
                    bs(PREVCASES, df = 6), data = dat)
plot(nonlinear.model, which = 1)
```



Now the plot of residuals vs fitted resemble straight line, although the variance of the residuals is still not constant over the scale of fitted values (heteroscedasticity).

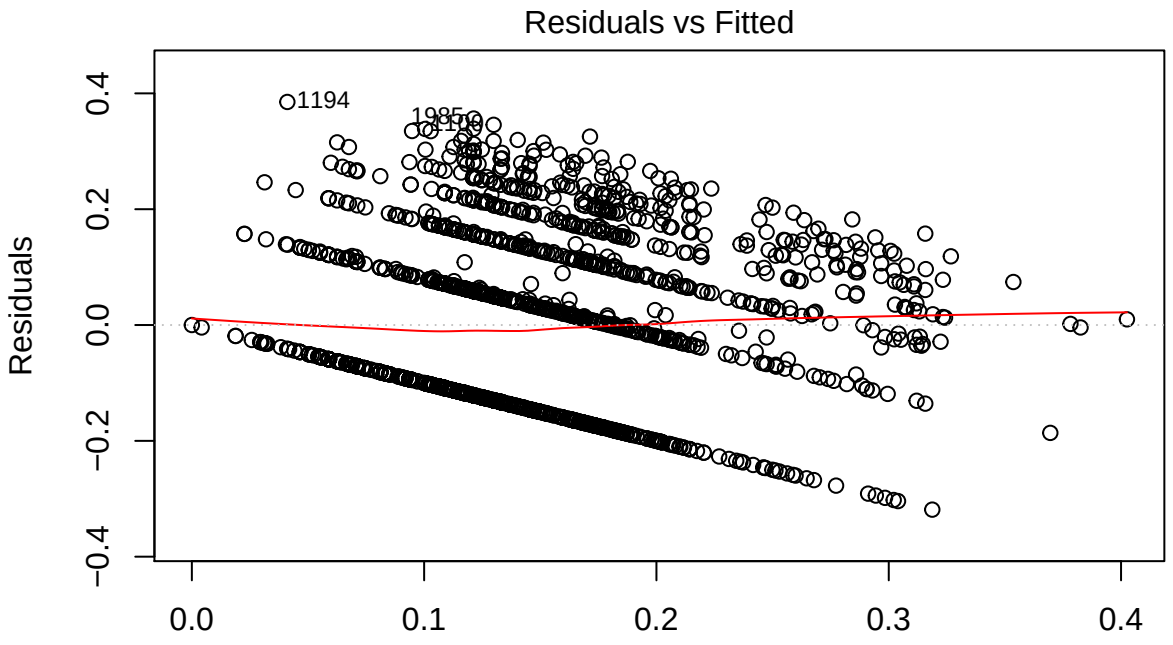
What function was estimated for these variables?

```
par(mfrow = c(1,3))
termplot(nonlinear.model, se = TRUE, terms = c("bs(AGE1STCASE, df = 6)","bs(AGE, df = 6)","bs(PREVCASES
```

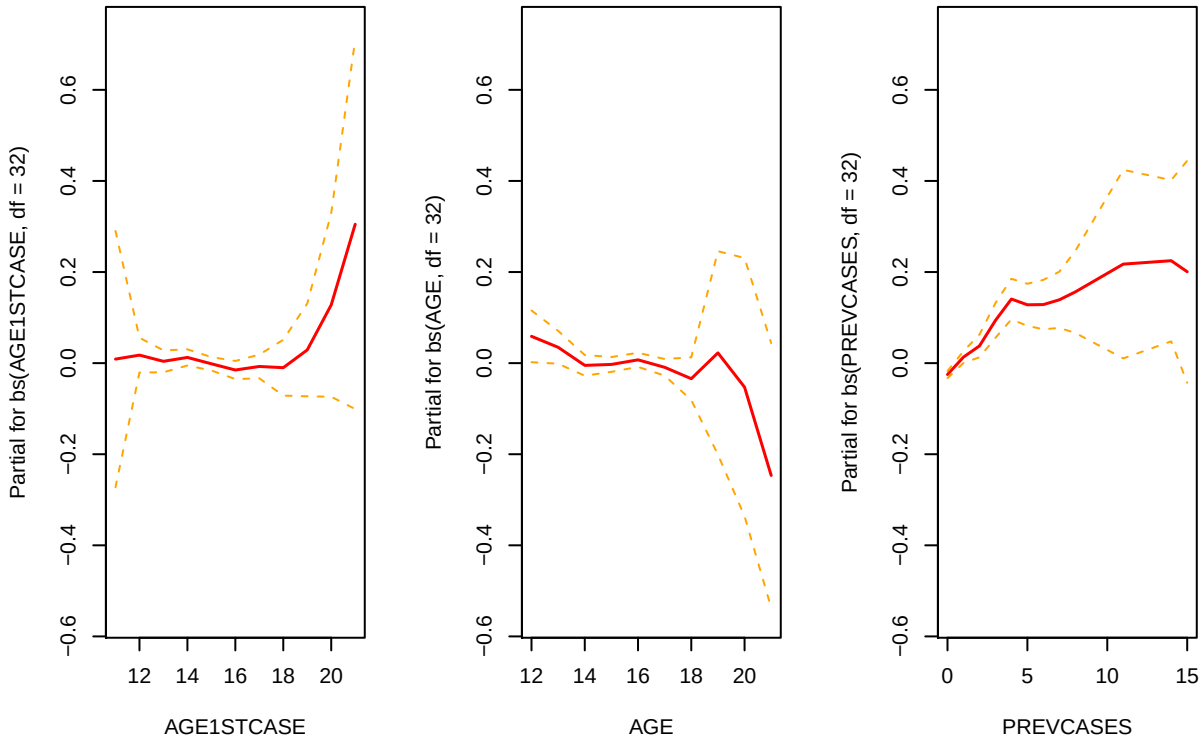


What happens if we allow more flexibility in modelling the continuous variables?

```
library(splines)
nonlinear.model2 <- lm(box.cox((SUMCASES / EXPOSURE * 365.25) + 1, optlambda) ~
                       SEX + COUNTRYBIRTH + bs(AGE, df = 32) +  bs(AGE1STCASE, df = 32) +  CRIMETYPE +
                       bs(PREVCASES, df = 32), data = dat)
plot(nonlinear.model2, which = 1)
```




```
par(mfrow = c(1,3))
termplot(nonlinear.model2, se = TRUE, terms = c("bs(AGE1STCASE, df = 32)","bs(AGE, df = 32)","bs(PREVCA
```



We can also visualize the relations of the previously fit models. How plausible are these relations? (note the scale of the y-axis)

```
par(mfrow=c(2,3))
coef(mod2) # coefficients
```

```
##                          (Intercept)                              SEXFemale
##                              0.90819                              -0.13278
##                  COUNTRYBIRTHMorocco           COUNTRYBIRTHDutch Antilles
##                              0.16599                              -0.04915
##                  COUNTRYBIRTHSurinam                  COUNTRYBIRTHTurkey
##                              0.07124                              -0.03836
##           COUNTRYBIRTHOther Western       COUNTRYBIRTHOther non-Western
##                             -0.10852                              -0.04916
##                                  AGE                              I(AGE^2)
##                             -0.06094                               0.00123
##                           AGE1STCASE                     CRIMETYPEsexual
##                             -0.00266                               0.07706
##     CRIMETYPEproperty with violence CRIMETYPEproperty without violence
##                              0.00232                               0.00748
##                CRIMETYPEpublic order                        CRIMETYPEdrugs
##                             -0.03518                               0.00691
##                    CRIMETYPEtraffic                         CRIMETYPEmisc
##                              0.01756                              -0.10337
##                  log(PREVCASES + 1)
##                              0.16311
```

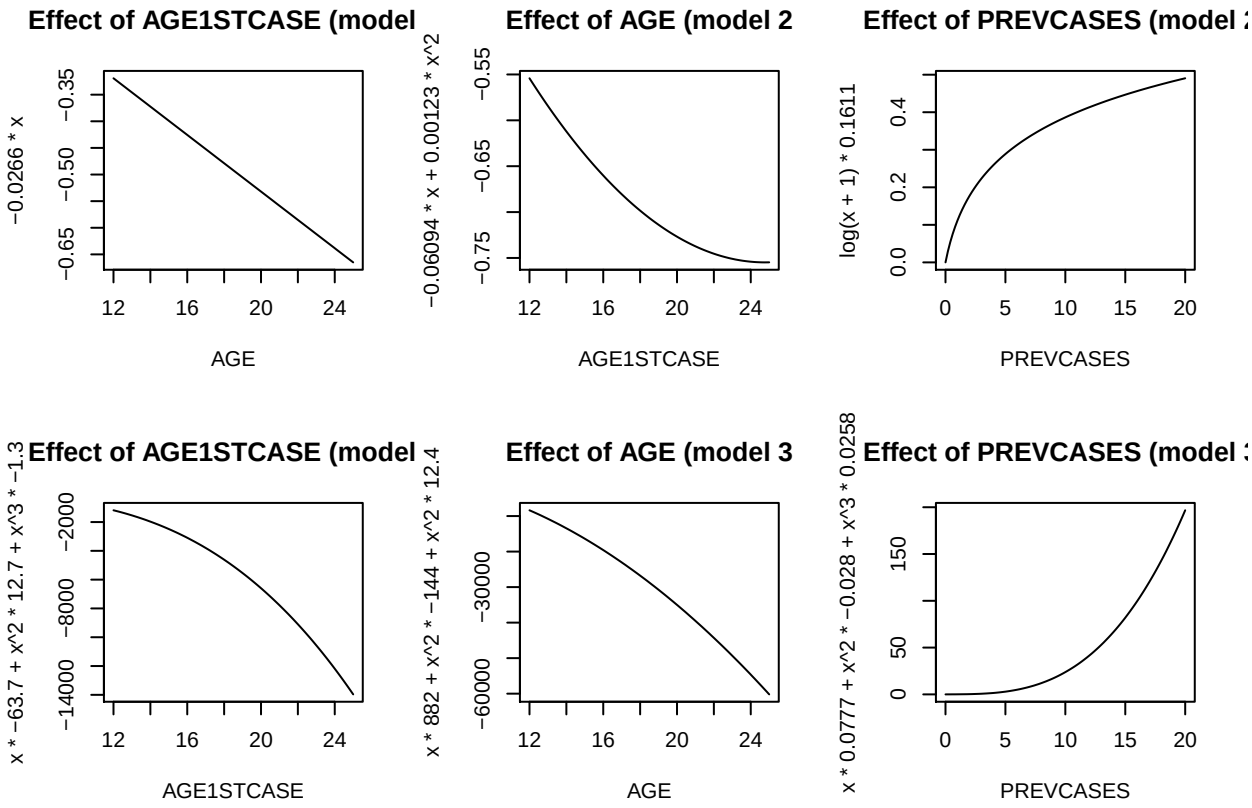
```r
curve(-0.0266*x, from = 12, to = 25, xlab = "AGE")
title('Effect of AGE1STCASE (model 2')
curve(-0.06094*x + 0.00123 * x ^2, from = 12, to = 25, xlab = "AGE1STCASE")
title('Effect of AGE (model 2')
curve(log(x + 1) * 0.1611, from = 0, to = 20, xlab = "PREVCASES") #coef(mod2[19]), from = c(0,20))
title('Effect of PREVCASES (model 2)')

coef(mod3) # coefficients
```

```
##                                 (Intercept)
##                                   -2.11e+03
##                                   SEXFemale
##                                   -1.27e-01
##                          COUNTRYBIRTHMorocco
##                                    1.71e-01
##                  COUNTRYBIRTHDutch Antilles
##                                   -4.92e-02
##                          COUNTRYBIRTHSurinam
##                                    6.18e-02
##                           COUNTRYBIRTHTurkey
##                                   -4.61e-02
##                   COUNTRYBIRTHOther Western
##                                   -1.14e-01
##               COUNTRYBIRTHOther non-Western
##                                   -4.86e-02
##                                         AGE
##                                    8.82e+02
##                                    I(AGE^2)
##                                   -1.44e+02
##                                    I(AGE^3)
##                                    1.24e+01
##                                  AGE1STCASE
##                                   -6.37e+01
##                              I(AGE1STCASE^2)
##                                    1.27e+01
##                              I(AGE1STCASE^3)
##                                   -1.30e+00
##                              CRIMETYPEsexual
##                                    8.02e-02
##             CRIMETYPEproperty with violence
##                                   -1.13e-03
##          CRIMETYPEproperty without violence
##                                    6.11e-03
##                        CRIMETYPEpublic order
##                                   -3.63e-02
##                               CRIMETYPEdrugs
##                                    8.25e-03
##                             CRIMETYPEtraffic
##                                    7.89e-03
##                                CRIMETYPEmisc
##                                   -1.01e-01
##                                   PREVCASES
##                                    7.77e-02
##                               I(PREVCASES^2)
```

```
##                                  -2.80e-02
##                               I(PREVCASES^3)
##                                     2.58e-02
##                                 AGE:I(AGE^2)
##                                          NA
##                                 AGE:I(AGE^3)
##                                    -5.98e-01
##                           I(AGE^2):I(AGE^3)
##                                     1.52e-02
##                   AGE1STCASE:I(AGE1STCASE^2)
##                                          NA
##                   AGE1STCASE:I(AGE1STCASE^3)
##                                     7.25e-02
##              I(AGE1STCASE^2):I(AGE1STCASE^3)
##                                    -2.10e-03
##                     PREVCASES:I(PREVCASES^2)
##                                          NA
##                     PREVCASES:I(PREVCASES^3)
##                                    -6.79e-03
##                I(PREVCASES^2):I(PREVCASES^3)
##                                     6.58e-04
##                        AGE:I(AGE^2):I(AGE^3)
##                                    -1.61e-04
## AGE1STCASE:I(AGE1STCASE^2):I(AGE1STCASE^3)
##                                     2.48e-05
##    PREVCASES:I(PREVCASES^2):I(PREVCASES^3)
##                                    -2.11e-05
```

```
curve(x * -6.37e+01 +x^2 * 1.27e+01 + x^3 * -1.30e+00, from = 12, to = 25, xlab = "AGE1STCASE")
title('Effect of AGE1STCASE (model 3')
curve(x * 8.82e+02 + x^2 * -1.44e+02 + x^2 * 1.24e+01 , from = 12, to = 25, xlab = "AGE")
title('Effect of AGE (model 3')
curve(x * 7.77e-02 + x^2 * -2.8e-02 + x^3 * 2.58e-02, from = 0, to = 20, xlab = "PREVCASES")
title('Effect of PREVCASES (model 3)')
```



Because of the many interactions in model 3, interpretation becomes much harder (watch the scale of the y-axis). This is because each effect is a *unique* effect *ceteris paribus* all other effects in the model.

regression trees

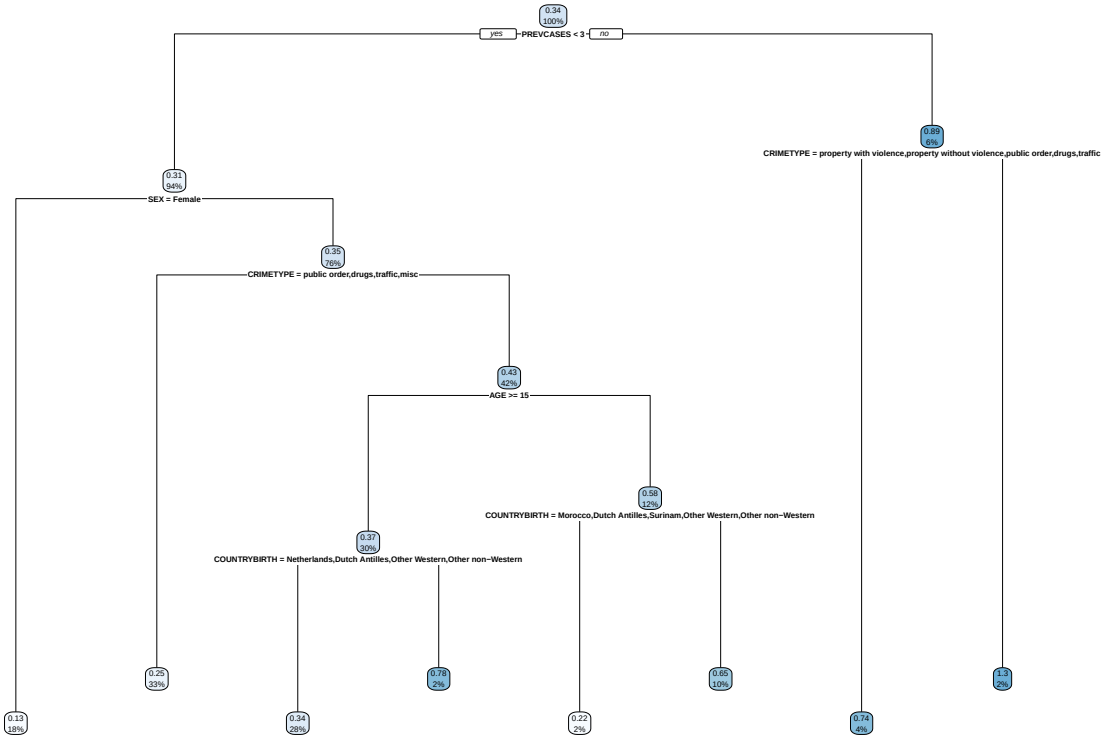
Now, instead of a statistical model, we try to create a regression tree for predicting the outcome. A regression tree splits the data recursively on a split point that generates the largest decreases in the MSE in the groups on both sides of the split. It is implemented in the function `rpart`, and needs a data frame and a formula as input. This technique is inherently nonlinear (invariant to order preserving transformations), we do not need to transform any variables.

For categorical variables with more than two categories, the algorithm applies a heuristic: the categories are sorted on the average y in the data and then splits are applied as if it is a continuous variable. This prevents that all possible combinations of splits need to be worked through, at the expense of some loss of accuracy.

```
library(rpart)
library(rpart.plot)
train2 <- train[1:400, ]
form <- SUMCASES / EXPOSURE * 365.25 ~ SEX + PREVCASES + AGE + AGE1STCASE + COUNTRYBIRTH + CRIMETYPE
tree <- rpart(form, data = train2, control = rpart.control( cp = 0.01))
yp.tree.train <- predict(tree)
yp.tree.val <- predict(tree, newdata = val)
tree.mse.train <- mse(train$SUMCASES / train$EXPOSURE * 365.25, yp.tree.train)
```

```
## Warning in obs - pred: longer object length is not a multiple of shorter
## object length
```

```
tree.mse.val <- mse(val$SUMCASES / val$EXPOSURE * 365.25, yp.tree.val)
rpart.plot(tree)
```




```
cat(sprintf("training error tree is %.4f", tree.mse.train),
    sprintf(", validation error tree is %.4f\n",tree.mse.val), sep = "")
```

```
## training error tree is 0.3430, validation error tree is 0.1698
```

```
tree
```

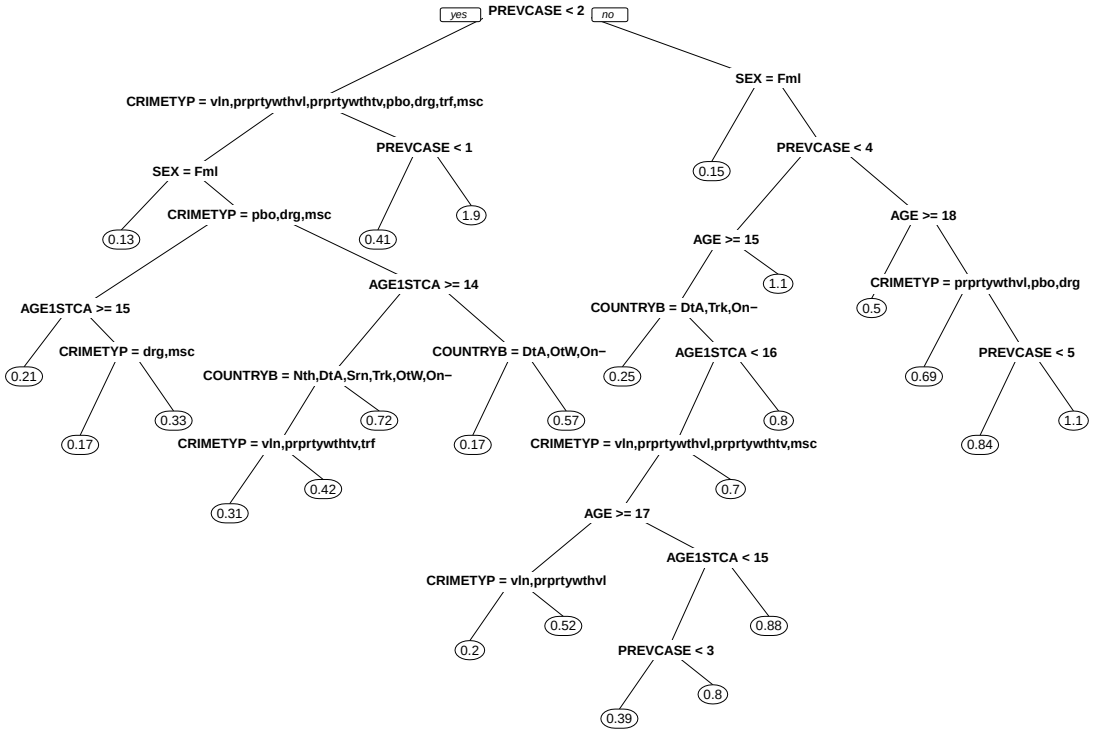
```
## n= 400
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 400 94.700 0.342
##    2) PREVCASES< 2.5 375 73.000 0.306
##      4) SEX=Female 73  2.780 0.127 *
##      5) SEX=Male 302 67.400 0.349
##       10) CRIMETYPE=public order,drugs,traffic,misc 133 15.600 0.248 *
##       11) CRIMETYPE=violence,sexual,property with violence,property without violence 169 49.300 0.429
##         22) AGE>=14.5 122 32.300 0.369
##           44) COUNTRYBIRTH=Netherlands,Dutch Antilles,Other Western,Other non-Western 113 25.400 0.33
##           45) COUNTRYBIRTH=Morocco,Surinam 9  5.250 0.782 *
##         23) AGE< 14.5 47 15.500 0.585
##           46) COUNTRYBIRTH=Morocco,Dutch Antilles,Surinam,Other Western,Other non-Western 7  0.429 0
##           47) COUNTRYBIRTH=Netherlands 40 13.900 0.650 *
```

```
##    3) PREVCASES>=2.5 25 13.700 0.889
##      6) CRIMETYPE=property with violence,property without violence,public order,drugs,traffic 18  2.
##      7) CRIMETYPE=violence,misc 7 10.000 1.280 *
```

The outcome is the number of penal cases per year. as the tree has 5 end nodes, only seven distinct values of predicted outcomes. About 66% of all observations falls into the third node. Therefore, we do not expect the model to do so well, and it indeed does not do well in terms of overall MSE. Note that a variable with many categories (i.e. COUNTRYBIRTH), is hard to discern in the visual tree.

What if we allow a more complex tree structure? For this, we decrease the `cp` parameter. This means, smaller increases in the overall R^2 will lead to a new split.

```
library(rpart)
form <- SUMCASES / EXPOSURE * 365.25 ~ SEX + PREVCASES + AGE + AGE1STCASE + COUNTRYBIRTH + CRIMETYPE
tree2 <- rpart(form, data = train, control = rpart.control(cp = 0.001))
yp.tree2.train <- predict(tree2)
yp.tree2.val <- predict(tree2, newdata = val)
tree2.mse.train <- mse(train$SUMCASES / train$EXPOSURE * 365.25, yp.tree2.train)
tree2.mse.val <- mse(val$SUMCASES / val$EXPOSURE * 365.25, yp.tree2.val)
prp(tree2)  #plotting function for more complex trees
```




```
cat(sprintf("training error tree is %.4f", tree2.mse.train),
    sprintf(", validation error tree is %.4f\n",tree2.mse.val), sep = "")
```

```
## training error tree is 0.2836, validation error tree is 0.1934
```

```
tree2
```

```
## n= 1216
```

```
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##    1) root 1216 414.000 0.344
##      2) PREVCASES< 1.5 1032 314.000 0.289
##        4) CRIMETYPE=violence,property with violence,property without violence,public order,drugs,tra
##          8) SEX=Female 194  10.300 0.131 *
##          9) SEX=Male 806 163.000 0.310
##           18) CRIMETYPE=public order,drugs,misc 333  43.900 0.240
##             36) AGE1STCASE>=14.5 204  22.700 0.206 *
##             37) AGE1STCASE< 14.5 129  20.600 0.292
##               74) CRIMETYPE=drugs,misc 31   1.880 0.171 *
##               75) CRIMETYPE=public order 98  18.100 0.331 *
##           19) CRIMETYPE=violence,property with violence,property without violence,traffic 473 117.00
##             38) AGE1STCASE>=13.5 396  91.500 0.328
##               76) COUNTRYBIRTH=Netherlands,Dutch Antilles,Surinam,Turkey,Other Western,Other non-Wes
##                152) CRIMETYPE=violence,property without violence,traffic 351  74.200 0.305 *
##                153) CRIMETYPE=property with violence 33   5.990 0.423 *
##               77) COUNTRYBIRTH=Morocco 12   8.980 0.725 *
##             39) AGE1STCASE< 13.5 77  22.700 0.523
##               78) COUNTRYBIRTH=Dutch Antilles,Other Western,Other non-Western 9   0.254 0.167 *
##               79) COUNTRYBIRTH=Netherlands,Morocco,Turkey 68  21.100 0.570 *
##        5) CRIMETYPE=sexual 32 129.000 0.732
##         10) PREVCASES< 0.5 25  10.300 0.409 *
##         11) PREVCASES>=0.5 7 107.000 1.890 *
##      3) PREVCASES>=1.5 184  79.400 0.652
##        6) SEX=Female 12   0.452 0.148 *
##        7) SEX=Male 172  75.700 0.688
##         14) PREVCASES< 3.5 121  52.700 0.616
##           28) AGE>=14.5 111  45.800 0.576
##             56) COUNTRYBIRTH=Dutch Antilles,Turkey,Other non-Western 9   0.260 0.253 *
##             57) COUNTRYBIRTH=Netherlands,Morocco,Surinam 102  44.500 0.604
##              114) AGE1STCASE< 15.5 89  28.700 0.575
##                228) CRIMETYPE=violence,property with violence,property without violence,misc 57  16.
##                  456) AGE>=16.5 23   2.080 0.380
##                    912) CRIMETYPE=violence,property with violence 10   0.356 0.203 *
##                    913) CRIMETYPE=property without violence,misc 13   1.170 0.516 *
##                  457) AGE< 16.5 34  14.000 0.588
##                    914) AGE1STCASE< 14.5 26   6.660 0.497
##                     1828) PREVCASES< 2.5 19   3.750 0.385 *
##                     1829) PREVCASES>=2.5 7   2.040 0.799 *
##                    915) AGE1STCASE>=14.5 8   6.450 0.883 *
##                229) CRIMETYPE=public order,drugs 32  11.200 0.702 *
##              115) AGE1STCASE>=15.5 13  15.200 0.802 *
##           29) AGE< 14.5 10   4.720 1.060 *
##         15) PREVCASES>=3.5 51  20.900 0.859
##           30) AGE>=17.5 7   1.010 0.502 *
##           31) AGE< 17.5 44  18.900 0.915
##             62) CRIMETYPE=property with violence,public order,drugs 11   3.170 0.690 *
##             63) CRIMETYPE=violence,sexual,property without violence,traffic,misc 33  15.000 0.991
##              126) PREVCASES< 4.5 17   6.900 0.844 *
##              127) PREVCASES>=4.5 16   7.320 1.150 *
```

Now, the tree has 25 end nodes (you can count this yourself using (`length(unique(tree2$where))`). Its MSE is better than the previous. If we allow even more splits, the model will eventually overfit. Try rerunning the model above with an even smaller complexity parameter.

Note that the `prp` tree plot truncates the labels for categorical variables, thereby increasing readability.

A random forest might improve the prediction error of this single tree. Folk wisdom tells us that random forests are relatively insensitive to different values of tuning parameters. We set `mtry` to 3, so only three random variables will be considered at each split point in each tree.

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
form <- SUMCASES / EXPOSURE * 365.25 ~ SEX + PREVCASES + AGE + AGE1STCASE + COUNTRYBIRTH + CRIMETYPE
set.seed(57729)
rf <- randomForest(form, data = train, mtry = 3, ntree = 500)
yp.rf.train <- predict(rf)
yp.rf.val <- predict(rf, newdata = val)
rf.mse.train <- mse(train$SUMCASES / train$EXPOSURE * 365.25, yp.rf.train)
rf.mse.val <- mse(val$SUMCASES / val$EXPOSURE * 365.25, yp.rf.val)
cat(sprintf("training error tree is %.4f", rf.mse.train),
    sprintf(", validation error tree is %.4f\n",rf.mse.val), sep = "")
```

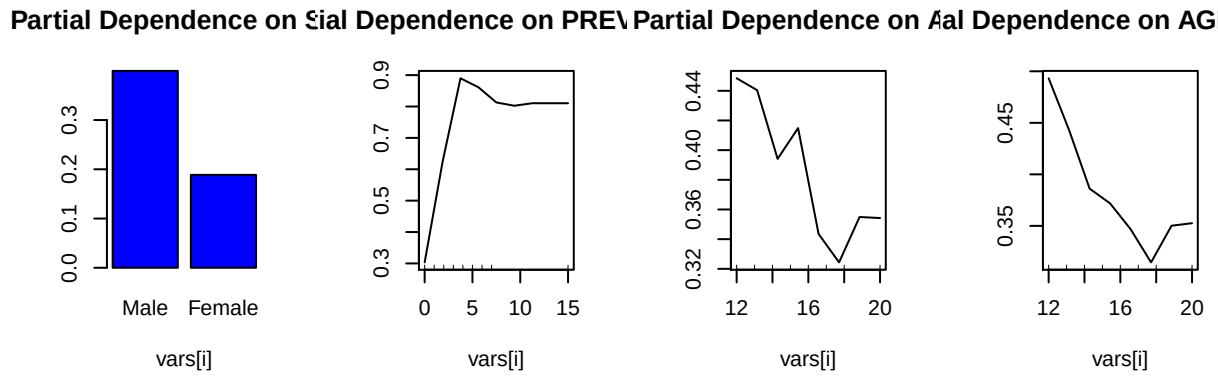
```
## training error tree is 0.3549, validation error tree is 0.1982
```

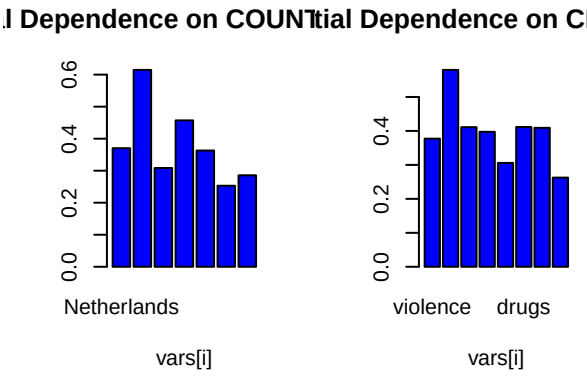
```
rf
```

```
##
## Call:
##  randomForest(formula = form, data = train, mtry = 3, ntree = 500)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.355
##                     % Var explained: -4.16
```

A feature of random forests (or boosting) is that we can visualize the effect of variables on the outcome by using partial dependence plots

```
vars <- all.vars(form[-1])
par(mfrow=c(2,4))
for (i in seq_along(vars)){
partialPlot(rf, pred.data = val, x.var = vars[i], main=paste("Partial Dependence on", vars[i]))
}
```





Again, when there is a lot of interactions, interpretation is hindered. As partial dependence plots show only the marginal effect of a variable, it can obscure actual interaction effects.

An additional statistic that can be requested on random forest models is the variable importance measure. For regression trees, it is calculated as the sum of the decrease in the sum of squares by splitting on the variable ('increase in the node purity').

```
varImpPlot(rf)
```


rf
CRIMETYPE
AGE1STCASE
PREVCASES
AGE
COUNTRYBIRTH
SEX
0
10
20
30
40
50
60
IncNodePurity