Using preferences to predict and explain human-decision making in game theory

J. van Buren

16 July 2018

Load packages

All packages that are used in this script are described and loaded in. The used functions of the packages are noted in brackets (). For more information see the corresponding help files.

- *QuantPsych*: used for extracting the standardized coefficients (lm.beta)
- *dplyr*: used for general data transformation (select, mutate, rename, inner_join)
- *glmnet*: used for conducting the lasso method (cv.glmnet, glmnet)
- *gam*: used for constructing a generalized additive model containing a smoothing spline (gam)
- *gbm*: used for conducting a gradient boosting machine (gbm)
- *caret*: used for fitting and training the gbm (trainControl, expand.grid, train)
- *car*: used for plotting the residuals (residualPlot, redidualsPlot)
- *bestglm*: used for finding the best subset selection of a logistic regresion (bestglm)

```
library(QuantPsyc)
library(dplyr)
library(glmnet)
library(gbm)
library(gam)
# library(caret): loaded in later to avoid conflicts
library(car)
library(bestglm)
```

thesisFunctions.R file

Other packages and functions are loaded in via "thesisFunctions.R" file. These packages are used in the created functions and described below. The used functions of the package are noted in brackets (). For more information see the corresponding help files.

- *psych*: used for descriptive statistics, a correlation test and for measuring Cronbach's alpha (describe, corr.test, alpha)
- *glmnet*: Used for conducting the lasso method and for plotting the shrinkage of the coefficients (glmnet)
- *corrplot*: used for a visualisation of the correlations (corrplot)
- *leaps*: used for conducting a best subset selection method of a linear regression model (regsubsets)
- *splines*: used for conducting a smoothing spline (smooth.spline)
- *ROCR*: used for plotting a ROC curve and accuracy/F-score vs threshold plot (prediction, performance)

Created functions loaded in via "thesisFunctions.R". For more information see the corresponding file.

- *plotBoxHist*: plots a standardized boxplot and a histogram in one figure. The histogram includes a mean and median line. Used in the function describeVariable.
- *plotBoxBar*: plots a standardized boxplot and a bar plot in one figure. Used in describeVariable.
- *plotBox*: plots a standardized bar plot. Used in describeVariable.

- *describeVariable*: describes a variable and returns the appropriate descriptive statistics. The function changes the description and plots based on whether the type of variable is a factor, discrete- or a continuous variable. In case of the latter, the parameter histogram = TRUE needs to be supplied.
- *plotCor*: plots a detailed correlation plot, If pvalues = FALSE, all correlations are coloured gradiently from -1 to +1, negative relations are coloured reddish and positive relationships are coloured blueish. If pvalues = TRUE, all significant relations are coloured gradiently, all other insignificant relations will have a white background. Returns the correlation table and if pvalues = TRUE also the p-values table.
- *cbAlpha*: calculates the Cronbach's alpha score, once for the normal input and once for the z-scaled variables of this input. Returns detailed scores and the correlations between the variables.
- *plotBoxes*: plots standardized boxplots for all preferences in one figure, split by a binary factor variable.
- *baseModelcv*: creates a base model and applies cross-validation. If a classification model is appropriate, then the majority class is taken as baseline and accuracy is measured. When a regression model is appropriate, RMSE is measured and the mean is used as a baseline. Returns the cross-validated score for every fold and the mean of these folds.
- *bestSubsetcv*: applies best subset selection and cross-validates the scores for every best subset corresponding with that number of variables. In case of a classification, logistic regression is conducted and all best models are supplied as a parameter. These are externally found via the bestglm function. In case of a regression model, linear regression is used and all best models are internally found using regsubsets and the R2 score. Returns a matrix with cross-validated scores for every fold with that number variables, for all number of variables. Also returns the standard deviation, the mean for every number of variables, the 'true' best number of variables and the corresponding score and standard deviation.
- *plotR2Subs*: plots the best R2 score for every number of variables and the variables corresponding with these scores in one figure. Returns the summary of regsubset.
- *plotModels*: plots the output scores of cross-validated models and places a red dot at the best performing model in the plot and returns these scores.
- *plotThreshold*: plots an Accuracy- and F-score- vs. threshold plot in one figure. If plotcontent = "ROC", a ROC curve is plotted and the AUC is returned.
- *plotShrinkage*: generates a glmnet model and then plots the shrinkage the coefficients. The red dotted line corresponds with the best lambda and a blue dotted line corresponds with the best lambda plus one standard error.
- *resultsLasso*: returns the results of the lassocv in a standardized way as in the other created cv functions. Returns the 'true' best lambda and the corresponding score and standard deviation and the best lambda plus one standard error. Takes a measure input, and adjusts the output in case of a general linear model.
- *plotBivariate*: plots a bivariate plot with a x variable and y variable and draws the general linear model or linear model best fit line. The plot and fit is adjusted if y == binary variable.
- *nonLinearcv*: cross-validation is applied on non-linear models. The non-linear models shoud be supplied. Returns the standardized output as in bestsubsetcv.
- *smoothSplinecv*: cross-validation is applied on smoothing splines for a range of supplied degrees of freedom from 2 until a supplied parameter of maxdegree. Returns the standardized output as in bestsubsetcv, but with the degrees of freedom included.

```
source("thesisFunctions.R")
```

Inspect workers

Load the data

Two datasets are loaded in.

- qualtricsNum: contains the survey data from Qualtrics in numerical values. Contains 317 entries and 61 variables (finished and unfinished surveys).
- amazon: contains all information of mturk workers. Contains 209 submitted entries, leaving 108 unfinished survey due to multiple reasons.

```
qualtrics <- read.csv2("data/qualtricsNum.csv")
dim(qualtrics)
```

```
## [1] 317  61
```

```
amazon <- read.csv2("data/amazon.csv", stringsAsFactors = FALSE)
nrow(amazon)
```

```
## [1] 209
```

Join datasets

In this section the datasets are prepared and joined together by the validation code, combining the validation code of Qualtrics with the inserted code by the workers on amazon. All incorrect matches are filtered. 204 workers remain and thus 5 workers have entered an invalid validation code.

```
amazon <- rename(amazon, valCode = Answer.surveycode)
qualtrics$valCode <- as.character(qualtrics$valCode)
qualtrics <- semi_join(qualtrics, amazon, by = "valCode")
dim(qualtrics)
```

```
## [1] 204  61
```

Reject invalid workers

Inspect the unmatched codes, these 5 entries are not matched and rejected.

```
amazon[!(amazon$valCode %in% qualtrics$valCode), c("WorkerId", "valCode")]
```

```
##           WorkerId
## 17   ATTNJ2IHS8MH8
## 20  A1AC6FP2BAJIOD
## 26  A1TML285XD4UZ7
## 202 A1UHNDB2EQ8TKO
## 207 A3PGUPNMOU5BPW
##                                                                valCode
## 17  27b53843f8a4a597bda1129b4277bbc460b6445f5cecceaf453db036136f
## 20                                                        A1AC6FP2BAJIOD
## 26                                                        A1TML285XD4UZ7
## 202                                                       A1UHNDB2EQ8TKO
## 207                                       Need some time to do this.
```

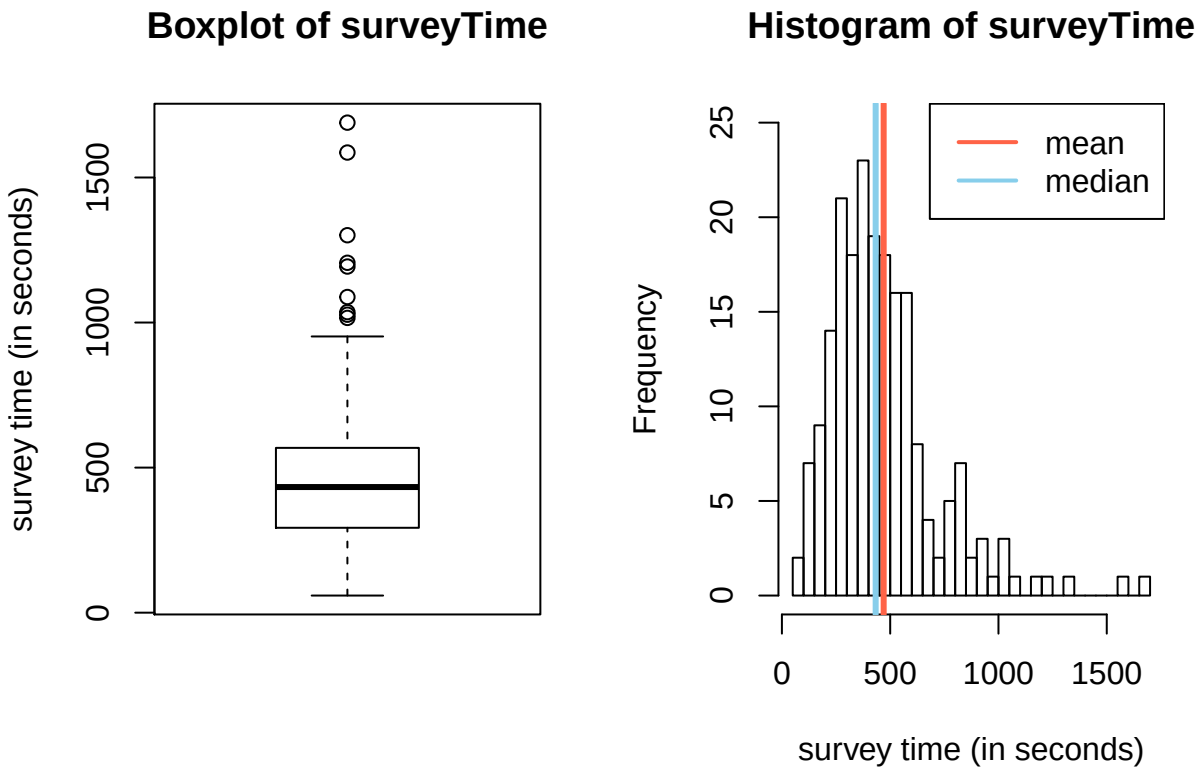
surveyTime

The surveyTime variable is inspected to see whether some respondents have highly unlikely fast responses. First the column name is renamed, hereafter surveyTime is inspected.

A mean of 470 seconds, median of 433 seconds and minimum of 59 seconds is found. Furthermore, the first quantile (25 %) is around 293 seconds, almost 5 minutes. The histogram and boxplot of the surveyTime variable are plotted.

However, even though some very fast responses have been observed, to avoid meddling with the data too much they are kept in for further analysis.

```
qualtrics <- rename(qualtrics, surveyTime = Duration..in.seconds.)
attach(qualtrics)
describeVariable(surveyTime, lab = "survey time (in seconds)", ylim = c(0,25), hist = TRUE)
```




```
## $describe
##    vars   n   mean     sd median trimmed    mad min  max range skew
## X1    1 204 469.87 255.12    433  439.09 206.82  59 1689  1630 1.58
##    kurtosis    se
## X1     3.91 17.86
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    59.0   292.8   433.0   469.9   566.0  1689.0
```

Pre-processing

Inspect the data

The columns are first inspected, there are a lot of metadata variables and attention questions that can be removed. Some names, as willingness_ and description_ are unnecessary complicated, and risk contains a lot of variables inherited from the filter questions to measure risk.

```
names(qualtrics)
```

```
##  [1] "StartDate"              "EndDate"
##  [3] "Status"                 "Progress"
##  [5] "surveyTime"             "Finished"
##  [7] "RecordedDate"           "ResponseId"
##  [9] "DistributionChannel"    "UserLanguage"
## [11] "consent"                "dilemma"
## [13] "chicken"                "offer"
## [15] "respond"                "public"
## [17] "attention1"             "risk1"
## [19] "willingness_negRec1"    "willingness_negRec2"
## [21] "willingness_altruism1"  "description_posRec1"
## [23] "description_negRec3"    "description_attention2"
## [25] "description_trust"      "risk160"
## [27] "risk80"                 "risk40"
## [29] "risk60"                 "risk70"
## [31] "risk50"                 "risk20"
## [33] "risk30"                 "risk10"
## [35] "risk120"                "risk100"
## [37] "risk90"                 "risk110"
## [39] "risk140"                "risk150"
## [41] "risk130"                "risk240"
## [43] "risk200"                "risk180"
## [45] "risk190"                "risk170"
## [47] "risk220"                "risk230"
## [49] "risk210"                "risk280"
## [51] "risk260"                "risk270"
## [53] "risk250"                "risk300"
## [55] "risk290"                "risk310"
## [57] "posRec2"                "altruism2"
## [59] "gender"                 "age"
## [61] "valCode"
```

Removing variables

Since the metadata columns are irrelevant, these are removed. Furthermore, valCode, the attention check questions and the variables gender, nationality and age are removed. After the removal, 45 columns are left.

```
qualtrics <- select(qualtrics, -(StartDate:consent), -contains("attention"),
 -(gender:valCode))
dim(qualtrics)
```

```
## [1] 204  45
```

Recoding variable names

Next, the variable names that contain willingness_ and description_ are recoded, because they inherited unintended names from Qualtrics

```
names(select(qualtrics, contains("willingness"), contains("description")))
```

```
## [1] "willingness_negRec1"   "willingness_negRec2"   "willingness_altruism1"
## [4] "description_posRec1"   "description_negRec3"   "description_trust"
```

```
names(qualtrics)[grepl("willingness*", names(qualtrics)) | grepl("description*", names(qualtrics))] <- c
```

Recoding risk

Now the staircase- and filter questions of risk are recoded.

```
names(select(qualtrics, contains("risk"), -risk1))
```

```
##  [1] "risk160" "risk80"  "risk40"  "risk60"  "risk70"  "risk50"  "risk20"
##  [8] "risk30"  "risk10"  "risk120" "risk100" "risk90"  "risk110" "risk140"
## [15] "risk150" "risk130" "risk240" "risk200" "risk180" "risk190" "risk170"
## [22] "risk220" "risk230" "risk210" "risk280" "risk260" "risk270" "risk250"
## [29] "risk300" "risk290" "risk310"
```

Only the risk variables that end on 10, 30, 50, 70 and 90 are used for determining the risk score, all others are removed.

```
qualtrics <- select(qualtrics, -c(contains("20"), contains("40"), contains("60"),
  contains("80"), contains("00")))
names(select(qualtrics, contains("risk"), -risk1))
```

```
##  [1] "risk70"  "risk50"  "risk30"  "risk10"  "risk90"  "risk110" "risk150"
##  [8] "risk130" "risk190" "risk170" "risk230" "risk210" "risk270" "risk250"
## [15] "risk290" "risk310"
```

Only the value that is not NA is counted, this is the particular risk score and all remaining risk variables are removed, except for risk1. The result is risk2, it contains the risk score and will be later inspected in more detail.

```
qualtrics <- mutate(qualtrics, risk2 = rowSums(select(qualtrics,
  contains("risk"), -risk1), na.rm = TRUE)) %>%
    select(-c(risk70:risk310))
qualtrics$risk2
```

```
##   [1]  8  4  1 13 15  1  8  4  6 17 15 10 10  4 17  7 31  7 29  7 20 11 29
##  [24] 10 13  2  8  5 20 32  2  4  4 17  7  3 10  7 19 20  7  1 12  4 15  8
##  [47]  3  8  4  5 10  2 15 10 18  8  9 15 15 11  4 17 14 15  8 10  6  8 15
##  [70] 20  4 24 10  6  9 13  2  7 20 17 10  7  8 25 25 31  3 10  9 10  4  2
##  [93]  4  3 14  4  8 10  7  9  5  1  5  4  6  9  8  2 12  4  4  1 18 21 13
## [116] 15  1  5 11  4  1 11 13  4 15  7 12  4 13  8  5  7 10  7  2  1  1 15
## [139] 10 32 20 10  5 18 12  4  4  7  2 31  4 16  1 14  4  4  1 10 10 10 21
## [162] 20 16 10  6 18  4 20  7 12  8 15  8  4  4  9 15 10 10 15 10  8  4 21
## [185]  3  4 10  4  4 12  3 10 32  2 25  1  1 15 12 18  6 15 32 10
```

Qualitative variables

Since dilemma and chicken are qualitative variables but imported as quantitative variables, they are transformed to factor variables. The levels of the variables are renamed to their corresponding categorical classes.

```
summary(qualtrics[c("dilemma", "chicken")])
```

```
##     dilemma          chicken
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000
##  Mean   :0.3235   Mean   :0.2745
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

```
qualtrics <- mutate_at(qualtrics, vars(dilemma:chicken), funs(as.factor))
levels(qualtrics$dilemma) <- c("cooperate", "defect")
levels(qualtrics$chicken) <- c("chicken", "dare")
summary(qualtrics[c("dilemma", "chicken")])
```

```
##       dilemma       chicken
##  cooperate:138   chicken:148
##  defect   : 66   dare   : 56
```

Re-ordering

Finally, all the remaining variables are re-ordered in a logical way.

```
names(qualtrics)
```

```
##  [1] "dilemma"   "chicken"   "offer"     "respond"   "public"
##  [6] "risk1"     "negRec1"   "negRec2"   "altruism1" "posRec1"
## [11] "negRec3"   "trust"     "posRec2"   "altruism2" "risk2"
```

```
qualtrics <- qualtrics[c(grep("public", names(qualtrics)), grep("offer", names(qualtrics)), grep("respo
```

General description

After all pre-processing, 15 variables and 204 observations are left, the variables contain five games, eight preferences and two risk variables.

```
names(qualtrics)
```

```
##  [1] "public"    "offer"     "respond"   "dilemma"   "chicken"
##  [6] "negRec1"   "negRec2"   "negRec3"   "posRec1"   "posRec2"
## [11] "altruism1" "altruism2" "trust"     "risk1"     "risk2"
```

```
dim(qualtrics)
```

```
## [1] 204  15
```

```
head(qualtrics)
```

```
##   public offer respond   dilemma chicken negRec1 negRec2 negRec3 posRec1
## 1      5     5       3    defect chicken       7       1       0      10
## 2      0     3       5    defect chicken       8       5       5       3
```

```
## 3      1     5       5 cooperate chicken       0       0       0      10
## 4      2     1       3 cooperate chicken       3       2       3       6
## 5      0     5       1 cooperate chicken       0       0       0      10
## 6      0     2       3    defect chicken       9      10       9      10
##   posRec2 altruism1 altruism2 trust risk1 risk2
## 1       2         8       200     8     6     8
## 2       1         3         0     5     7     4
## 3       3         9         0     3     0     1
## 4       2         6        25     5     3    13
## 5       1        10         0     4     6    15
## 6       1         6        10     5     4     1
```

```
str(qualtrics)
```

```
## 'data.frame':    204 obs. of  15 variables:
##  $ public   : int  5 0 1 2 0 0 4 10 5 1 ...
##  $ offer    : int  5 3 5 1 5 2 5 3 5 5 ...
##  $ respond  : int  3 5 5 3 1 3 5 3 2 1 ...
##  $ dilemma  : Factor w/ 2 levels "cooperate","defect": 2 2 1 1 1 2 2 2 1 1 ...
##  $ chicken  : Factor w/ 2 levels "chicken","dare": 1 1 1 1 1 1 1 1 1 1 ...
##  $ negRec1  : int  7 8 0 3 0 9 2 5 4 0 ...
##  $ negRec2  : int  1 5 0 2 0 10 0 6 4 0 ...
##  $ negRec3  : int  0 5 0 3 0 9 7 5 2 0 ...
##  $ posRec1  : int  10 3 10 6 10 10 9 3 7 10 ...
##  $ posRec2  : int  2 1 3 2 1 1 2 1 4 1 ...
##  $ altruism1: int  8 3 9 6 10 6 0 6 6 10 ...
##  $ altruism2: int  200 0 0 25 0 10 0 500 10 25 ...
##  $ trust    : int  8 5 3 5 4 5 2 7 5 10 ...
##  $ risk1    : int  6 7 0 3 6 4 0 7 2 9 ...
##  $ risk2    : num  8 4 1 13 15 1 8 4 6 17 ...
```

```
attach(qualtrics)
```

Univariate analysis

Next, univariate analysis of all variables will be performed. It consists of two section: the game theoretical games and the preferences and risk variables.

Game theory

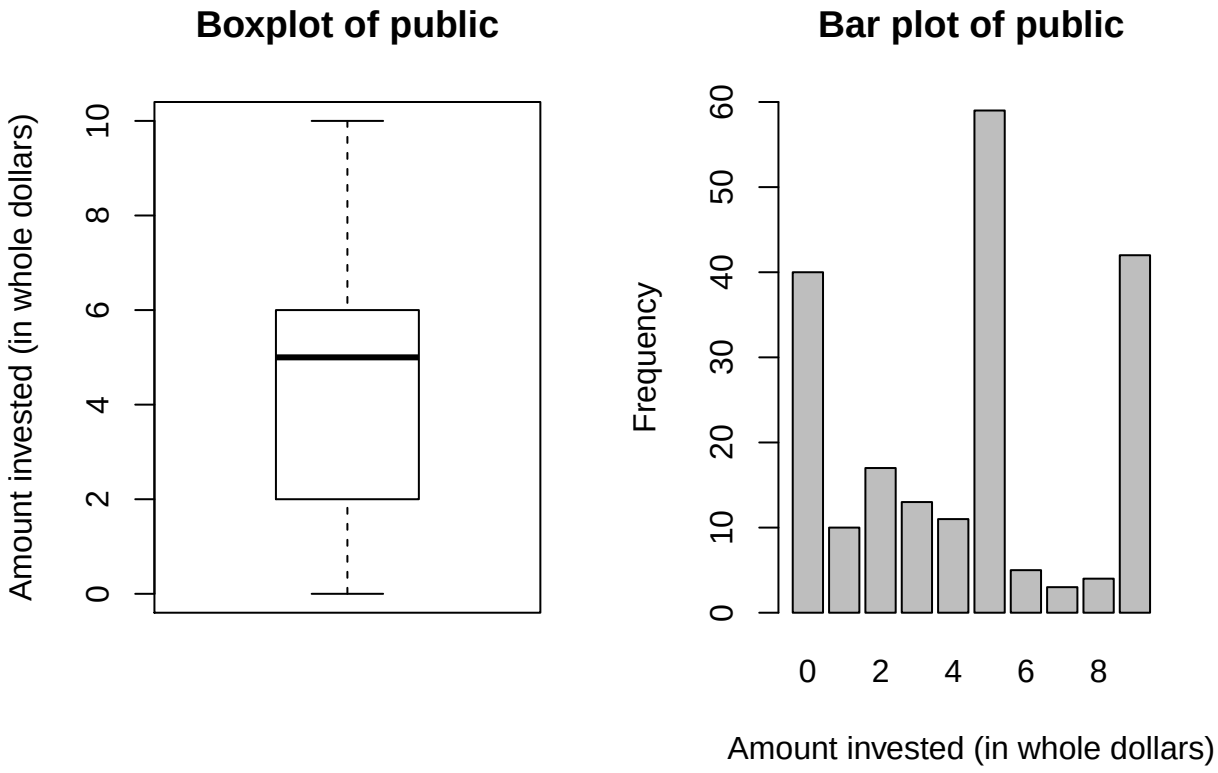
In the first section all games are inspected.

Public goods

Firstly, the amount invested in the public goods game is examined (in whole dollars). The mean is 4.53 and the median is 5 dollars, this is also the mode with 59 respondents (28.9%) investing 5 dollars. It has multiple peaks, one at 0 dollar, 5 dollars and at 10 dollars (all, half or nothing). The standard deviation is 3.46, which shows quite some variance, as is seen in the inter quartile range of the boxplot. The data is somewhat evenly distributed than the other two games, but still has a higher tendency in the lower numbers as confirmed by the interquartile range and the bar plot and does not follow a normal distribution caused by the multiple peaks. The excess kurtosis of -1.04 shows a mesokurtic character of the data.

The game theoretical solution would be to invest zero dollars, since you will share evenly in the pot no matter what amount you will invest in it. A player will thus maximize their investment by investing zero dollars in the pot, and still share evenly in the amount that other respondents have put into it. However, this is an inefficient solution, if everybody would cooperate, *all* respondents would get more money, but this is not a stable solution since as mentioned above, the best scenario for an individual is then that everybody invest 10 dollars and you invest zero dollars. All in all, the game theoretical solution of investing zero dollars is mostly not found, only 40 of 204 (19,6%) respondents do play this way. Most likely altruism, trust and positive reciprocity plays a role in the amount invested, where this relation is expected to be positive.

```
describeVariable(public, lab = "Amount invested (in whole dollars)", ylim = c(0,60))
```




```
## $describe
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 204 4.53 3.46      5    4.42 4.45   0  10    10 0.31    -1.04 0.24
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   5.000   4.534   6.000  10.000
##
## $table
##
##  0  1  2  3  4  5  6  7  8 10
## 40 10 17 13 11 59  5  3  4 42
##
## $prop.table
##
##     0     1     2     3     4     5     6     7     8    10
```

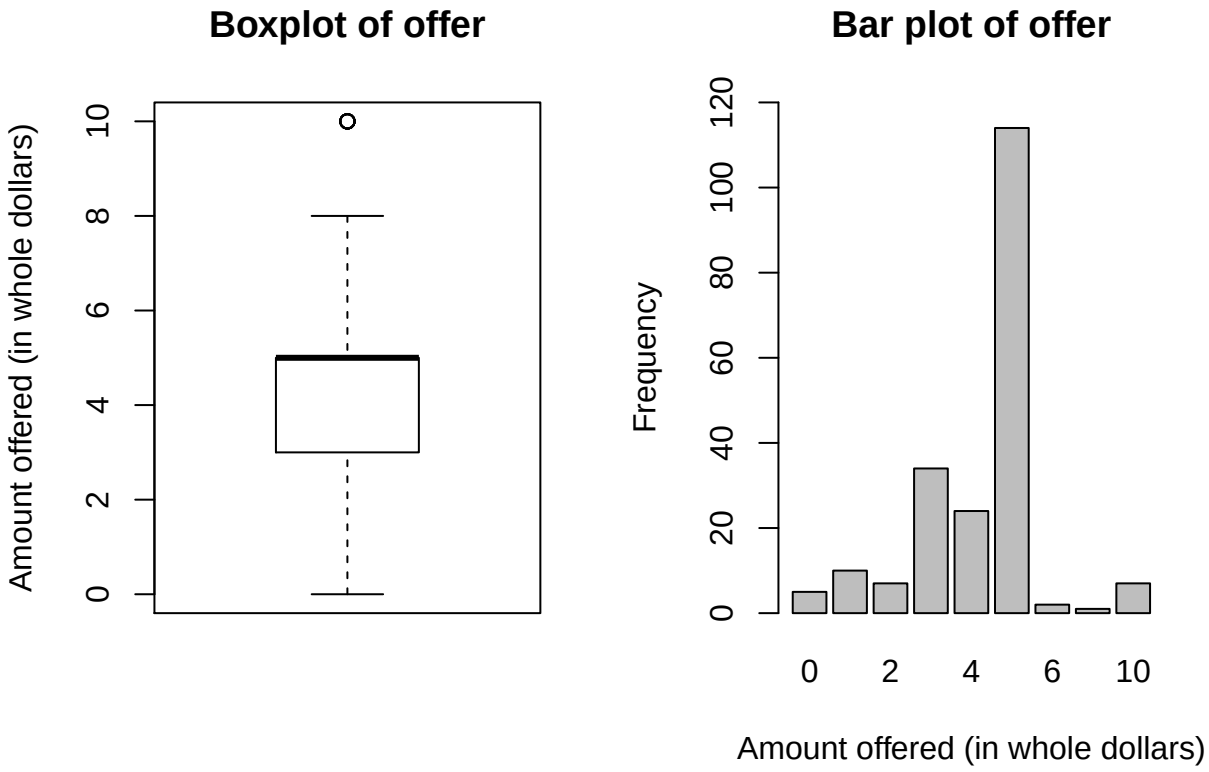
```
## 0.196 0.049 0.083 0.064 0.054 0.289 0.025 0.015 0.020 0.206
```

Ultimatum game: offer

The amount offered in the ultimatum game (in whole dollars) has a mean of 4.32 and a median of 5 dollars. This is also the mode with 114 (55.9%) respondents offering 5 dollars. The standard deviation is 1.72. Furthermore, as expected, the data is centredto the left, to the lower number (5 or smaller). This can also be seen in the bar plot and in the boxplot, such as that the inter quartile range is from 3 to 5, note the axis of the bar plot. The data is strongly leptokurtic, confirmed by a excess kurtosis of 3.03. There are also some outliers, namely seven offers of 10 dollars, but all offers greater than 5 dollar are somewhat odd.

The game theoretical solution is an offer of 1 dollar, which in turn a rational person would accept since every offer greater than zero is more than receiving nothing and should be accepted. However, theory states that negative reciprocity cause people to turn down offers that seem unfair. Most likely person adjust their offer on what they think people what people will accept, as can be seen in the distributions, and might offer lower if they are willing to take risk. It could also be possible that respondents might offer a 'friendly' amount, this relates to positive reciprocity or altruism, this relation is expected to be positive.

```
describeVariable(offer, lab = "Amount offered (in whole dollars)", ylim = c(0,120))
```




```
## $describe
##    vars   n mean   sd median trimmed mad min max range skew kurtosis   se
## X1    1 204 4.32 1.72      5     4.4   0   0  10    10 0.42     3.03 0.12
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.000   5.000   4.324   5.000  10.000
```

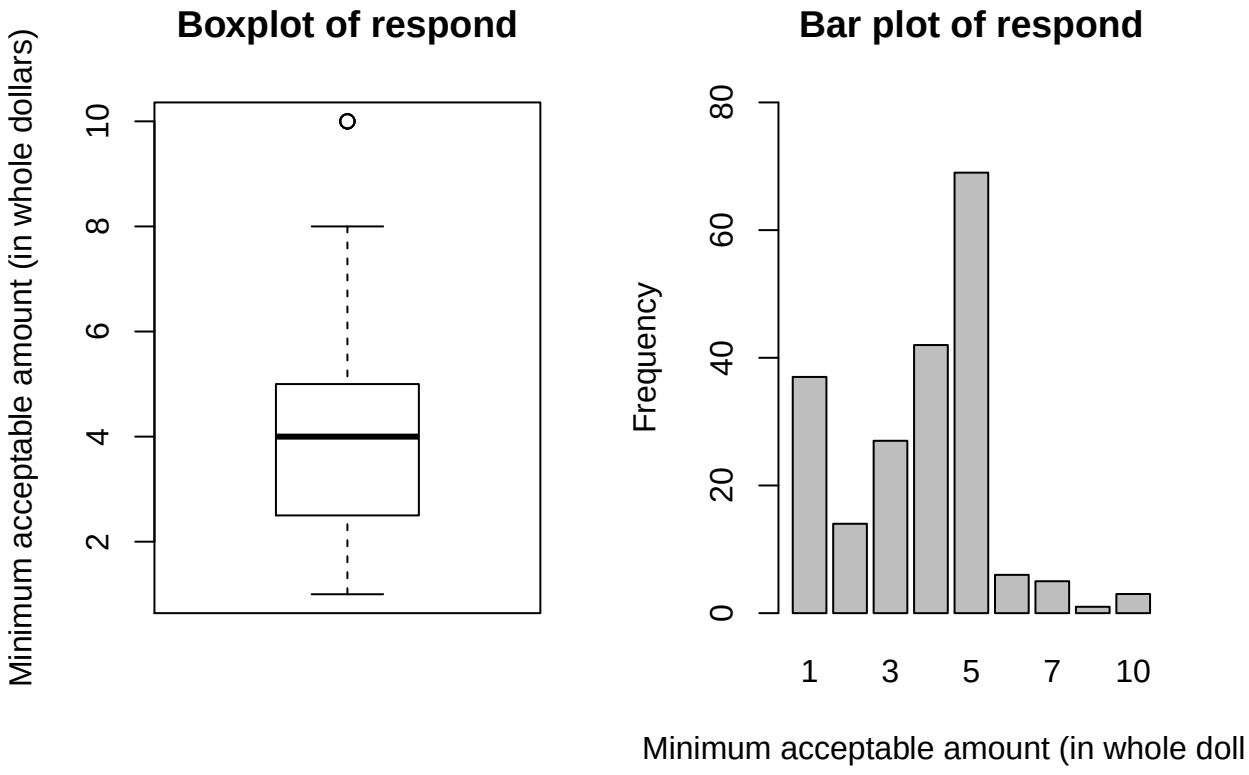
```
##
## $table
##
##   0   1   2   3   4   5   6   8  10
##   5  10   7  34  24 114   2   1   7
##
## $prop.table
##
##     0     1     2     3     4     5     6     8    10
## 0.025 0.049 0.034 0.167 0.118 0.559 0.010 0.005 0.034
```

Ultimatum game: response

The amount that is minimally acceptable in the ultimatum game (in whole dollars) has a mean of 3.76 and a median of 4 dollars, with an standard deviation of 1.81. The results are lower than in the ultimatum game offer, indicating some kind of inefficiency. Nonetheless, the mode is 5 dollars with 69 respondents (33.8%). Furthermore, the data is even more centredto the lower values than the offered amount, as can be seen in the histogram by the interquartile range and distance from the whisker to the 25% quantile, and also in the bar plot. Again, there are some outliers, namely three respondents found 10 dollar minimally acceptable and eleven respondents demanded more than 5 dollars. Five respondents accept a bid of 0 dollars. The excess kurtosis of 0.65 shows a slight leptokurtic character of the data.

As mentioned above, the rational solution would be to accept every offer greater than 0, however these results are not found. Theory states that this is caused by negative reciprocity, people will reject what they deem as unfair offers.

```
describeVariable(respond, lab = "Minimum acceptable amount (in whole dollars)", ylim = c(0,80))
```

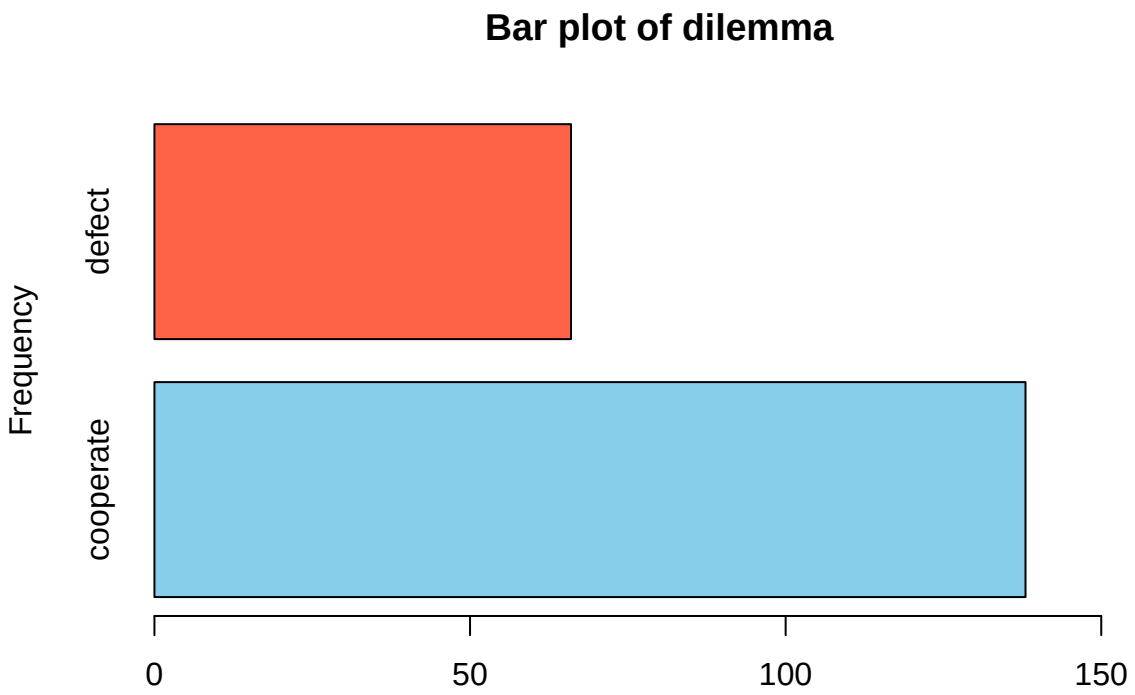



```
## $describe
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 204 3.76 1.81      4    3.74 1.48   1  10     9 0.25     0.65 0.13
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.750   4.000   3.765   5.000  10.000
##
## $table
##
##  1  2  3  4  5  6  7  8 10
## 37 14 27 42 69  6  5  1  3
##
## $prop.table
##
##     1     2     3     4     5     6     7     8    10
## 0.181 0.069 0.132 0.206 0.338 0.029 0.025 0.005 0.015
```

Prisoner's dilemma

The percentage of respondents that chose defect is 32%, corresponding with a total of 138 co-operators and 66 defectors. The game theoretical solution is defect, since in all scenario's defect will maximize payoffs and is thus a dominating strategy. This solution is in great contrast with the actual found results. Most likely trust, altruism and/or positive reciprocity plays a role in cooperating. The relation is expected to be a positive one.

```
describeVariable(dilemma)
```

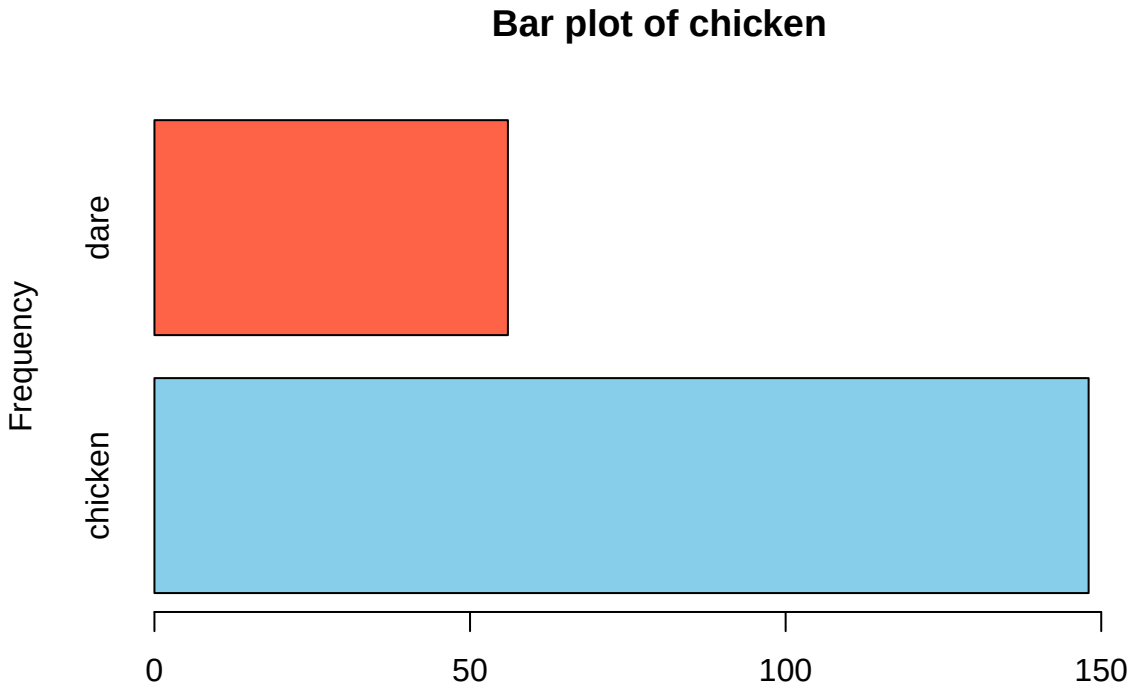



```
## $table
## x
## cooperate    defect
##       138        66
##
## $prop.table
## x
## cooperate    defect
##     0.676     0.324
```

Chicken game

Next the chicken game is studied. The percentage of dares is 27%, this corresponds 148 respondents playing chickens and 56 dare. There are slightly more co-operators/chickens in the chicken game than in the prisoner's dilemma. The game theoretical solution would be a percental combination of chicken and dare, in this case dare is not a dominating strategy. [CALCULATE] Most likely trust, altruism, positive reciprocity and risk plays a role in cooperating. The first 3 are expected to have a positive relationship, but the latter a negative relationship. The higher the risk, the more likely defect is chosen.

```
describeVariable(chicken)
```




```
## $table
## x
## chicken    dare
##     148      56
##
## $prop.table
## x
## chicken    dare
##   0.725   0.275
```

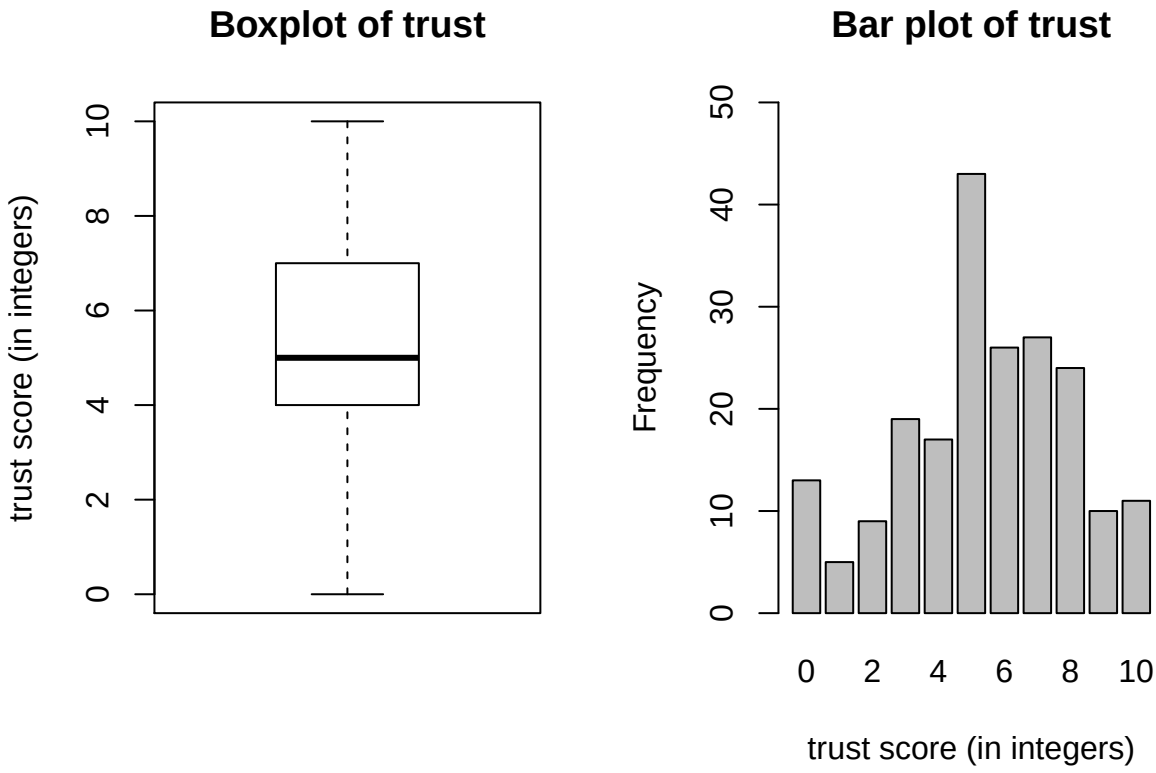
Preferences and risk

In this part the preferences and risks are inspected.

Trust

Firstly, the concept trust is inspected. It measures whether people assume others have good intentions. It has a mean of 5.39. The median, and the mode lie at 5. The standard deviation is 2.56 and the data is approximately symmetric, but has a tendency for higher scores. Furthermore the data has a slightly mesokurtic character, shown by the value of -0.42.

```
describeVariable(trust, lab = "trust score (in integers)", ylim = c(0,50))
```




```
## $describe
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 204 5.39 2.56      5    5.49 2.97   0  10    10 -0.3    -0.42 0.18
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.000   5.000   5.392   7.000  10.000
##
## $table
##
##  0  1  2  3  4  5  6  7  8  9 10
## 13  5  9 19 17 43 26 27 24 10 11
##
## $prop.table
##
##     0     1     2     3     4     5     6     7     8     9    10
## 0.064 0.025 0.044 0.093 0.083 0.211 0.127 0.132 0.118 0.049 0.054
```

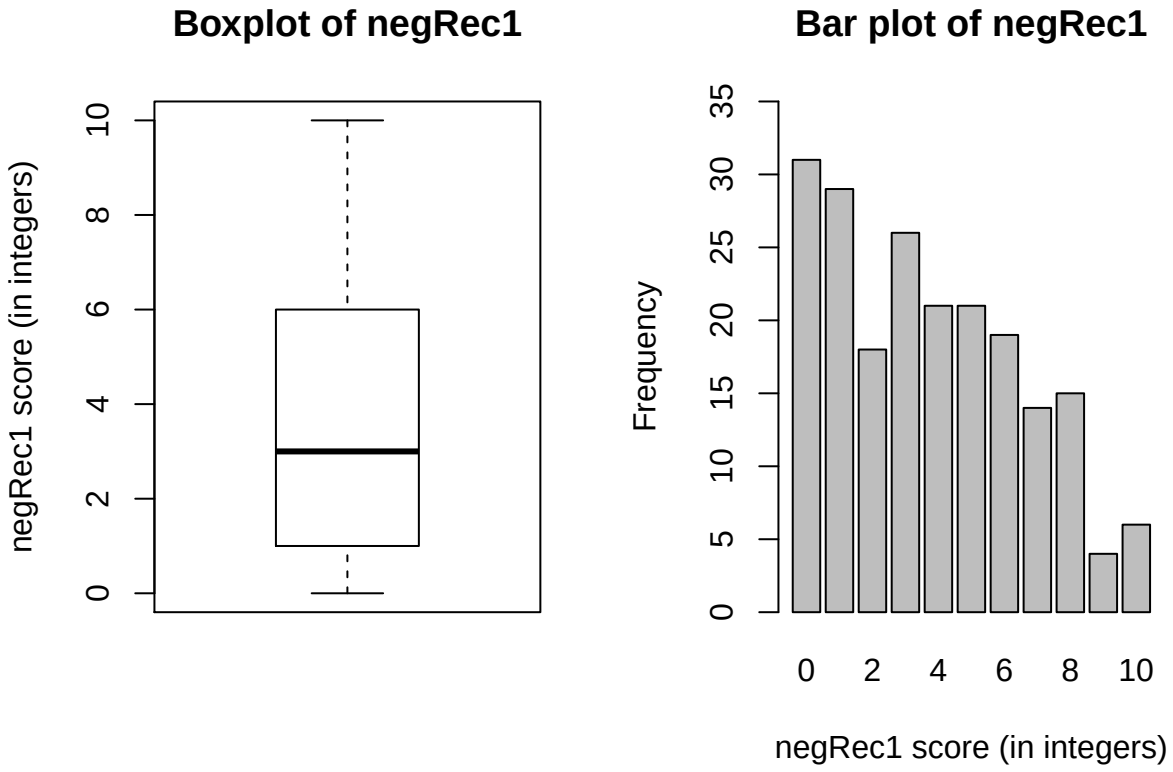
Negative reciprocity

Secondly, the concept negative reciprocity is inspected. In general it measures the tendency in which people are willing to punish others even at a cost for themselves.

negRec1

The first variable is negRec1, it measures how willing people are to punish someone who treats them unfairly, even at a cost for them self. It has a mean of 3.73, the median lies at 3 and the mode at 0. The standard deviation is 2.82 and the data is slightly right skewed with a higher tendency for lower scores, and has no symmetric characteristics. The data has a mesokurtic character, shown by the value of -0.89.

```
describeVariable(negRec1, lab = "negRec1 score (in integers)", ylim = c(0,35))
```

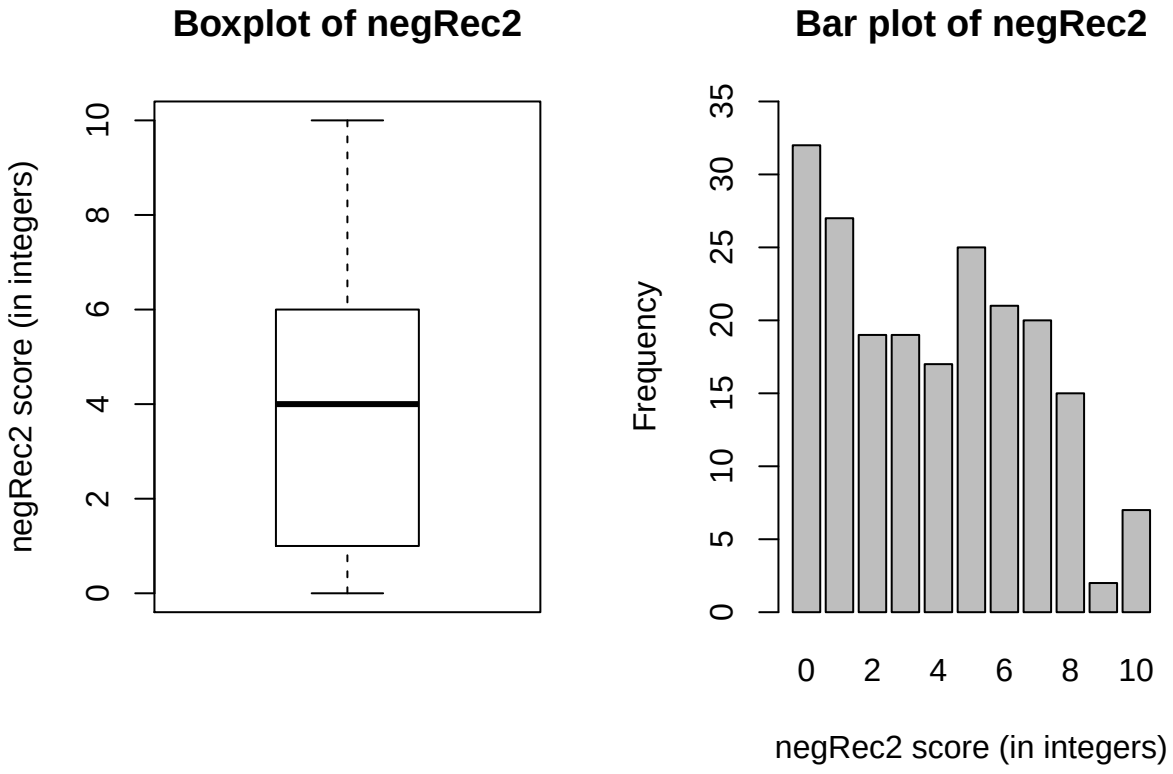



```
## $describe
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis  se
## X1    1 204 3.73 2.82      3    3.56 2.97   0  10    10 0.37    -0.89 0.2
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   3.000   3.725   6.000  10.000
##
## $table
##
##  0  1  2  3  4  5  6  7  8  9 10
## 31 29 18 26 21 21 19 14 15  4  6
##
## $prop.table
##
##     0     1     2     3     4     5     6     7     8     9    10
## 0.152 0.142 0.088 0.127 0.103 0.103 0.093 0.069 0.074 0.020 0.029
```

negRec2

negRec2 measures how willing people are to punish someone who treats *others* unfairly, even at a cost for them self. It has a mean of 3.87, the median lies at 4 and the mode at 0. The standard deviation is 2.87 and the data is slightly right skewed with a higher tendency for smaller scores, and has no symmetric characteristics. Furthermore the data has a mesokurtic character, shown by the value of -1.04.

```
describeVariable(negRec2, lab = "negRec2 score (in integers)", ylim = c(0,35))
```

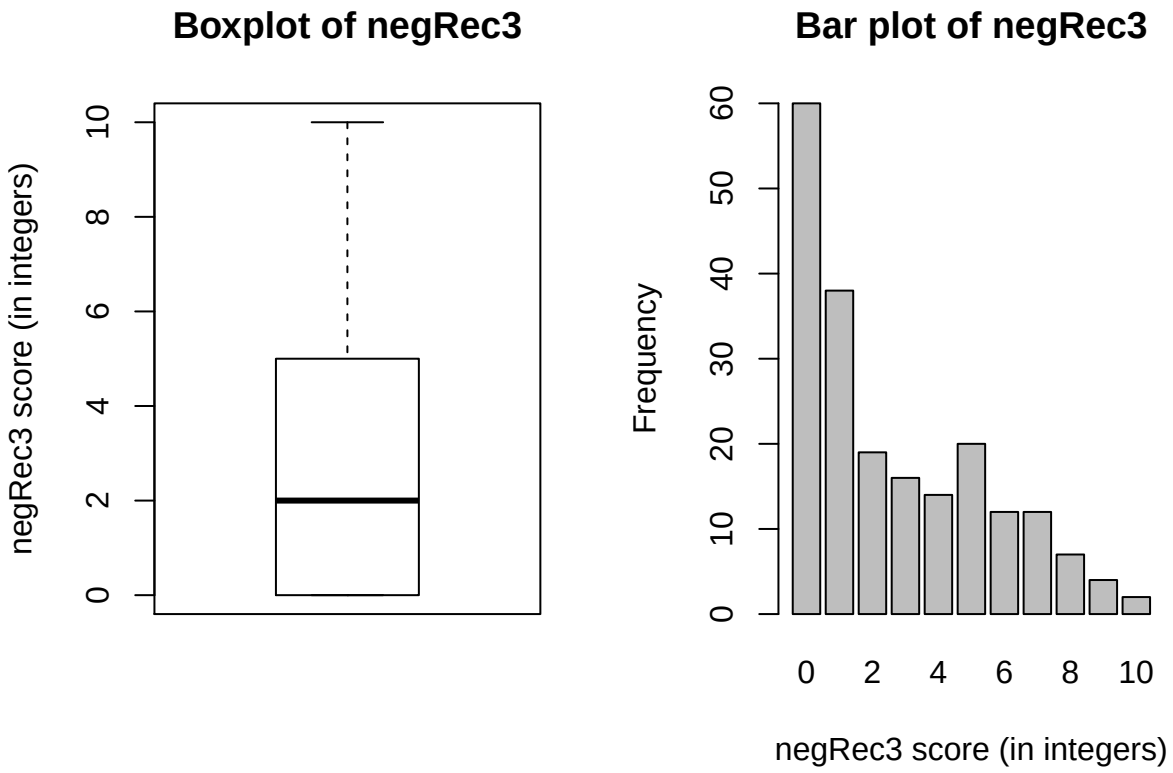



```
## $describe
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis  se
## X1    1 204 3.87 2.87      4    3.74 4.45   0  10    10 0.24    -1.04 0.2
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   4.000   3.868   6.000  10.000
##
## $table
##
##  0  1  2  3  4  5  6  7  8  9 10
## 32 27 19 19 17 25 21 20 15  2  7
##
## $prop.table
##
##     0     1     2     3     4     5     6     7     8     9    10
## 0.157 0.132 0.093 0.093 0.083 0.123 0.103 0.098 0.074 0.010 0.034
```

negRec3

negRec3 measures people's tendency to take revenge if threaten unfairly, even at a cost for them self. It has a mean of 2.69, the median lies at 2 and the mode at 0. The standard deviation is 2.72 and the data is right skewed with a strong tendency for smaller scores, confirmed by the skewness value of 0.77 and has no symmetric characteristics at all. Furthermore the data has a slight mesokurtic character, shown by the value of -0.55.

```
describeVariable(negRec3, lab = "negRec3 score (in integers)", ylim = c(0,60))
```




```
## $describe
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 204 2.69 2.72      2    2.36 2.97   0  10    10 0.77    -0.55 0.19
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   2.000   2.686   5.000  10.000
##
## $table
##
##  0  1  2  3  4  5  6  7  8  9 10
## 60 38 19 16 14 20 12 12  7  4  2
##
## $prop.table
##
##     0     1     2     3     4     5     6     7     8     9    10
## 0.294 0.186 0.093 0.078 0.069 0.098 0.059 0.059 0.034 0.020 0.010
```

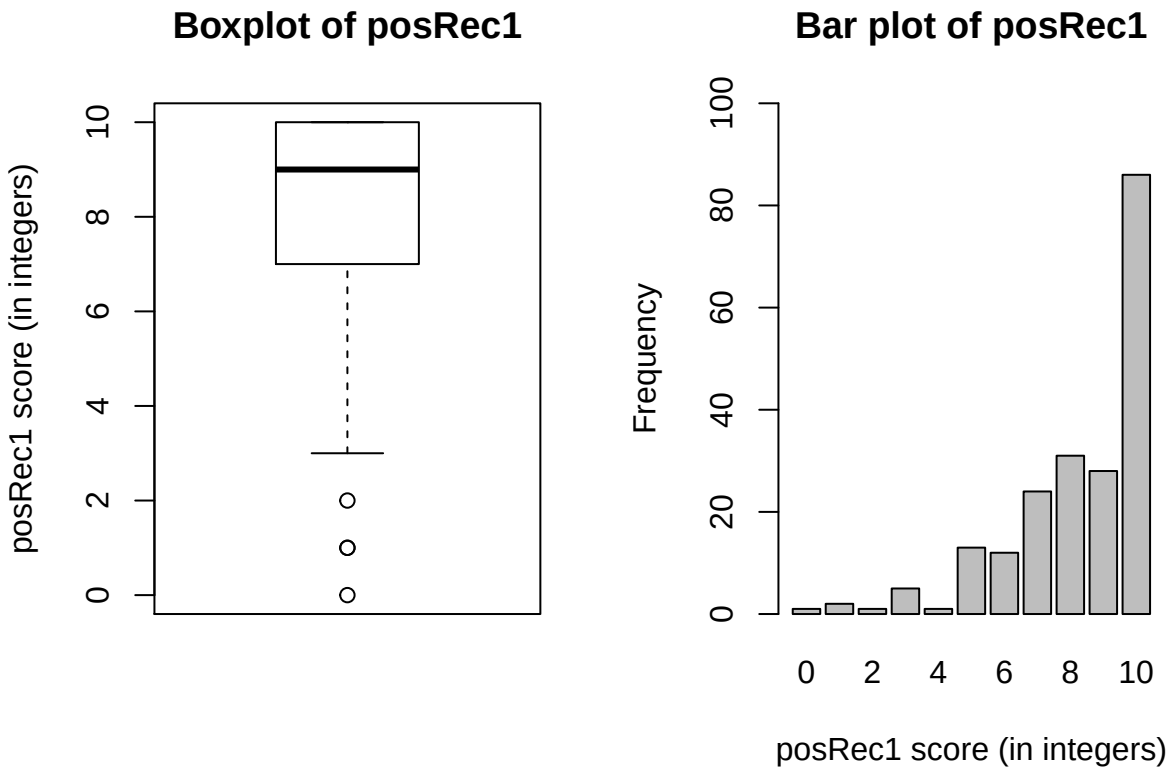
Positive reciprocity

Now, the concept positive reciprocity is inspected. In general it measures the tendency in which people are willing treat others well, who treat them well, even at a cost for themselves.

posRec1

posRec1 measures people's willingness to return a favour. It has a mean of 8.27, the median lies at 9 and the mode at 10. The standard deviation is 2.09 and the data is strongly left skewed with a tendency for higher scores, confirmed by a skewness of -1.4 and has no symmetric characteristics. It has excess kurtosis of 1.82 indicating a leptokurtic character of the data.

```
describeVariable(posRec1, lab = "posRec1 score (in integers)", ylim = c(0,100))
```




```
## $describe
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 204 8.27 2.09      9    8.63 1.48   0  10    10 -1.4     1.82 0.15
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   7.000   9.000   8.275  10.000  10.000
##
## $table
##
##  0  1  2  3  4  5  6  7  8  9 10
##  1  2  1  5  1 13 12 24 31 28 86
```

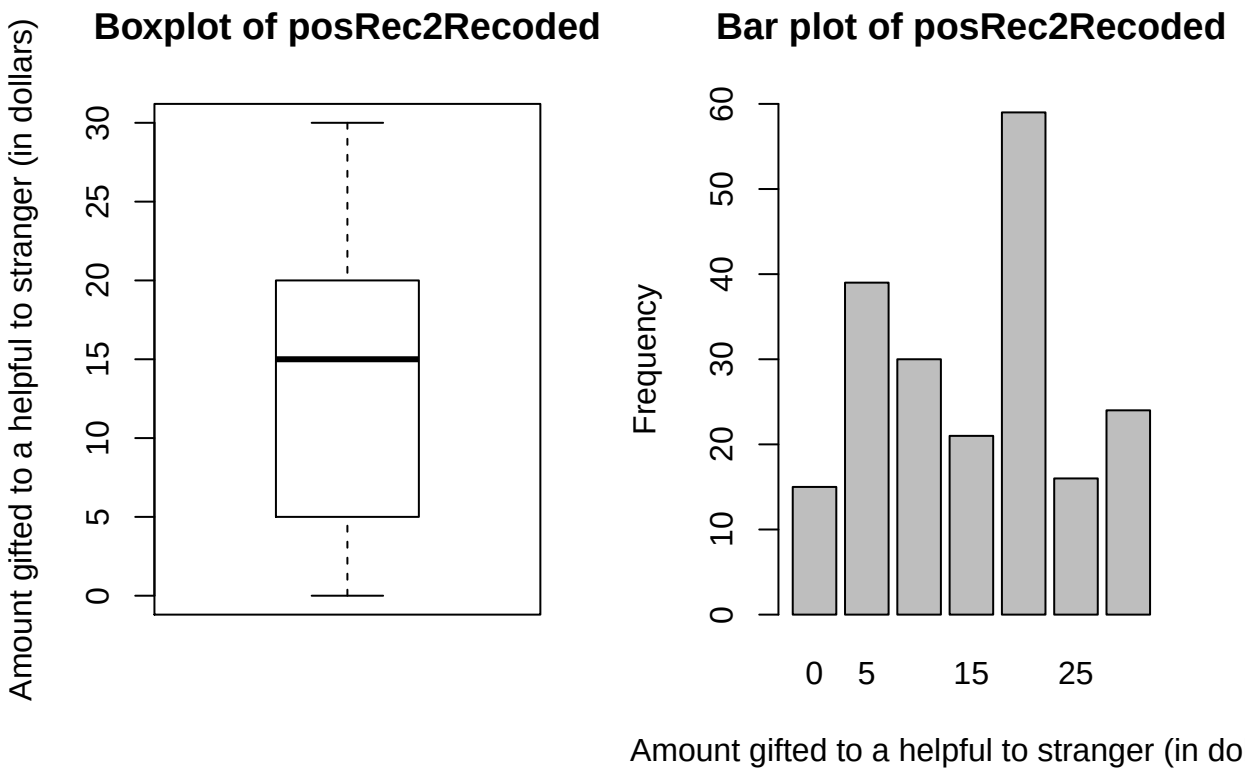
```
##
## $prop.table
##
##     0     1     2     3     4     5     6     7     8     9    10
## 0.005 0.010 0.005 0.025 0.005 0.064 0.059 0.118 0.152 0.137 0.422
```

posRec2

posRec2 measures if people are willing to give a gift worth of 5, 10, 15, 20, 25 or 30 dollars, or give no present to a stranger in return for a costly helpful deed (costing the stranger 20 dollars). It has been coded from steps from 0 to 6. For the description it will be re-coded to the original values of 0 to 30, by multiplying the variable with 5.

It has a mean of 15.25, the median lies at 15 and the mode at 20 dollars. The standard deviation is 9.01 and the data is somewhat symmetric, but has a mesokurtic character shown by the value of -1.09 and multiple peaks at 5 and 20.

```
posRec2Recoded <- posRec2 * 5
describeVariable(posRec2Recoded, lab = "Amount gifted to a helpful to stranger (in dollars)", ylim = c(0
```




```
## $describe
##    vars   n  mean   sd median trimmed  mad min max range  skew kurtosis
## X1    1 204 15.25 9.01     15   15.15 7.41   0  30    30 -0.01    -1.09
##      se
## X1 0.63
##
## $summary
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    5.00   15.00   15.25   20.00   30.00
##
## $table
##
##  0  5 10 15 20 25 30
## 15 39 30 21 59 16 24
##
## $prop.table
##
##     0     5    10    15    20    25    30
## 0.074 0.191 0.147 0.103 0.289 0.078 0.118
```

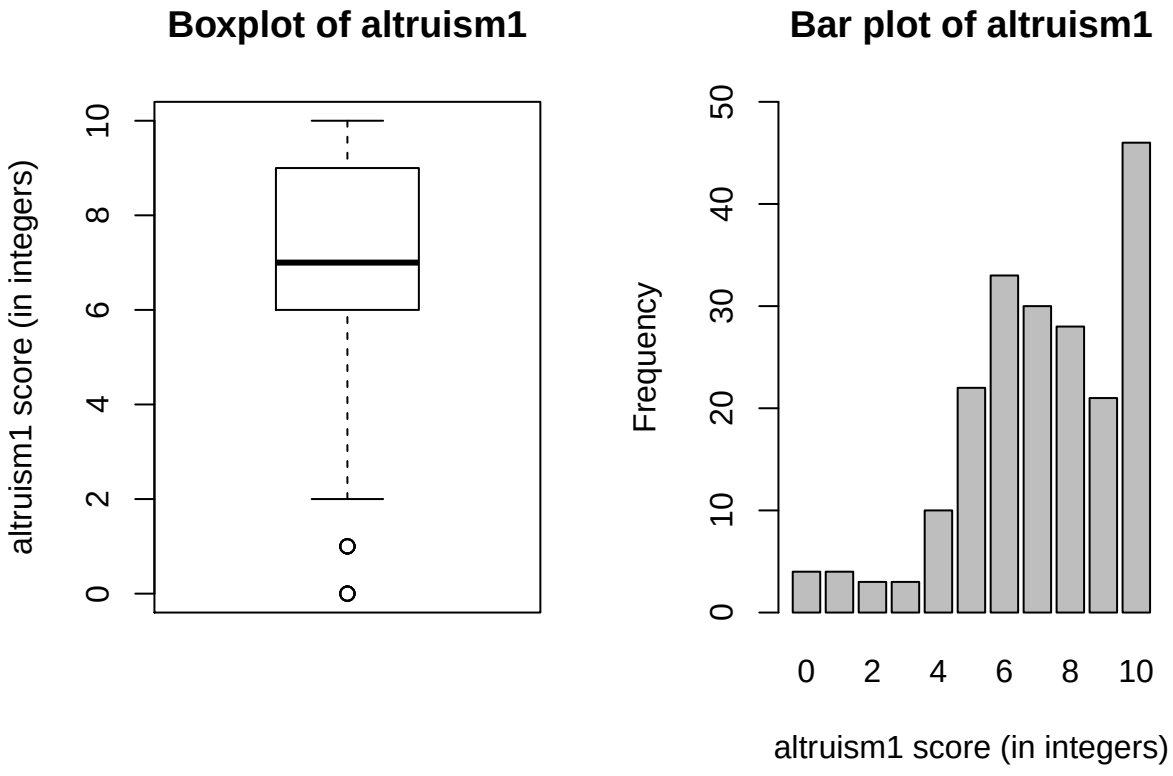
Altruism

Now, the concept altruism is inspected. In general it measures the care and concern for the happiness and well-being of other people.

altruism1

altruism1 measures if people are willing to give to good causes without expecting anything in return. It has a mean of 7.11, the median lies at 7 and the mode at 6. The standard deviation is 2.43 and the data is moderately left-skewed shown by the value of -0.76, with a tendency for higher scores. It has no symmetric characteristics and has an approximately platykurtic character shown by the value of 0.32.

```
describeVariable(altruism1, lab = "altruism1 score (in integers)", ylim = c(0,50))
```

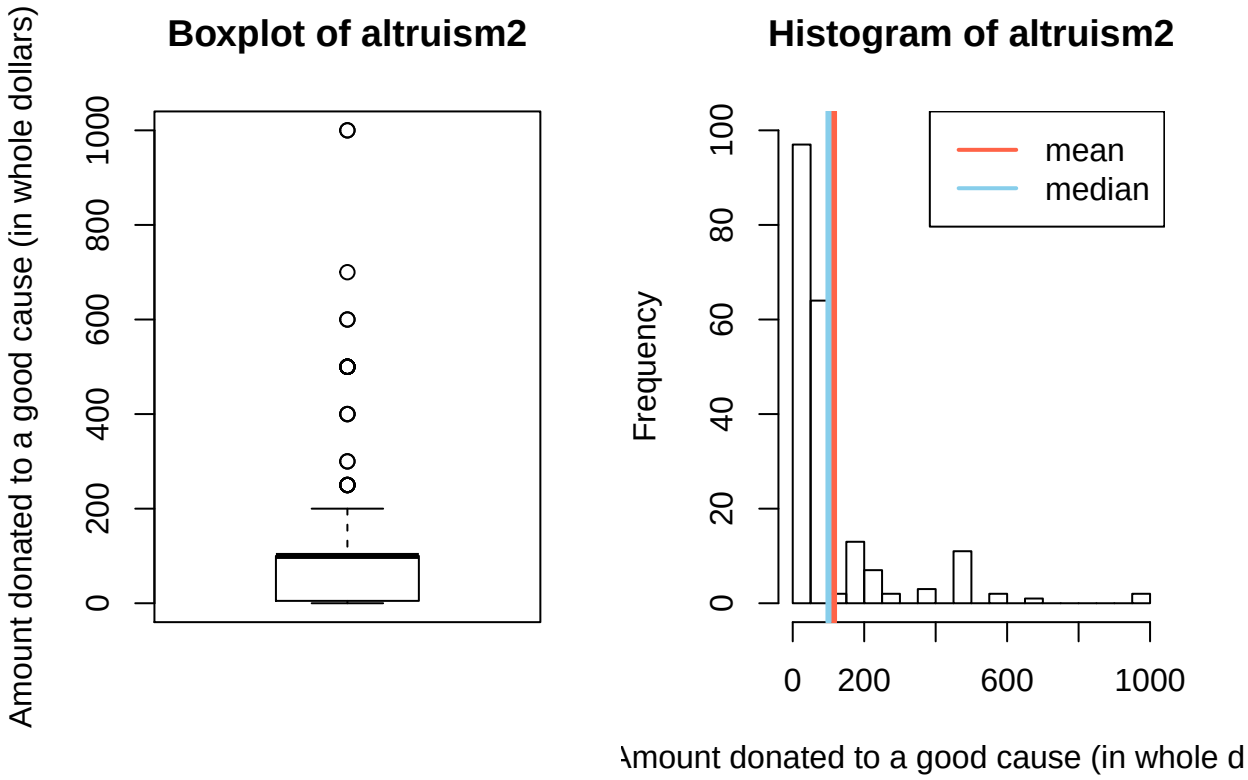


```
## $describe
##    vars   n mean   sd median trimmed  mad min max range  skew kurtosis
## X1    1 204 7.11 2.43      7    7.36 2.97   0  10    10 -0.76     0.32
##      se
## X1 0.17
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   6.000   7.000   7.108   9.000  10.000
##
## $table
##
##  0  1  2  3  4  5  6  7  8  9 10
##  4  4  3  3 10 22 33 30 28 21 46
##
## $prop.table
##
##     0     1     2     3     4     5     6     7     8     9    10
## 0.020 0.020 0.015 0.015 0.049 0.108 0.162 0.147 0.137 0.103 0.225
```

altruism2

altruism2 measures how much people are willing to donate to a good cause if they unexpectedly received 1000 dollars. It has a mean of 115.62, the median lies at 100. The standard deviation is 166, this means that it has quite a large variance, this is caused by outliers. It has quite some outliers offering more than 250 dollars. The data is very strongly right-skewed confirmed by the skewness value of 2.62, with a tendency for lower values and has very a strong leptokurtic character shown by the value of 8.22.

```
describeVariable(altruism2, lab = "Amount donated to a good cause (in whole dollars)", ylim = c(0,100),
```




```
## $describe
##    vars   n   mean     sd median trimmed    mad min  max range skew
## X1    1 204 115.62 165.76    100   77.36 133.43   0 1000  1000 2.62
##    kurtosis    se
## X1     8.22 11.61
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     5.0   100.0   115.6   100.0  1000.0
```

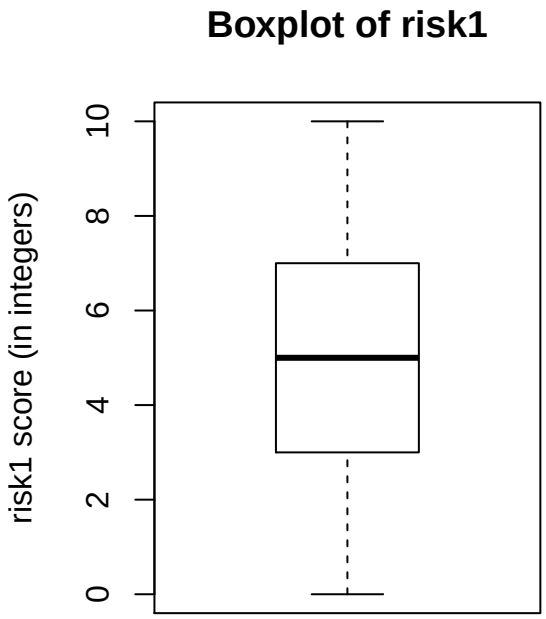
Risk

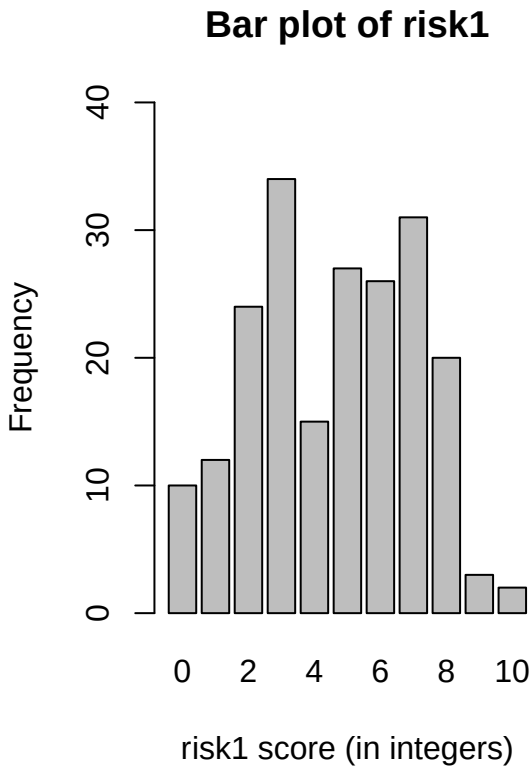
Lastly, the concept risk is inspected. It has to be noted that this is not a social preference. In general it measures the willingness to take risks under uncertainty.

risk1

risk1 measures willingness to take risks. It has a mean of 4.59, the median lies at 5 and the mode at 3. The standard deviation is 2.43 and the data is approximately symmetric, but has a mesokurtic character shown by the value of -0.99.

```
describeVariable(risk1, lab = "risk1 score (in integers)", ylim = c(0,40))
```





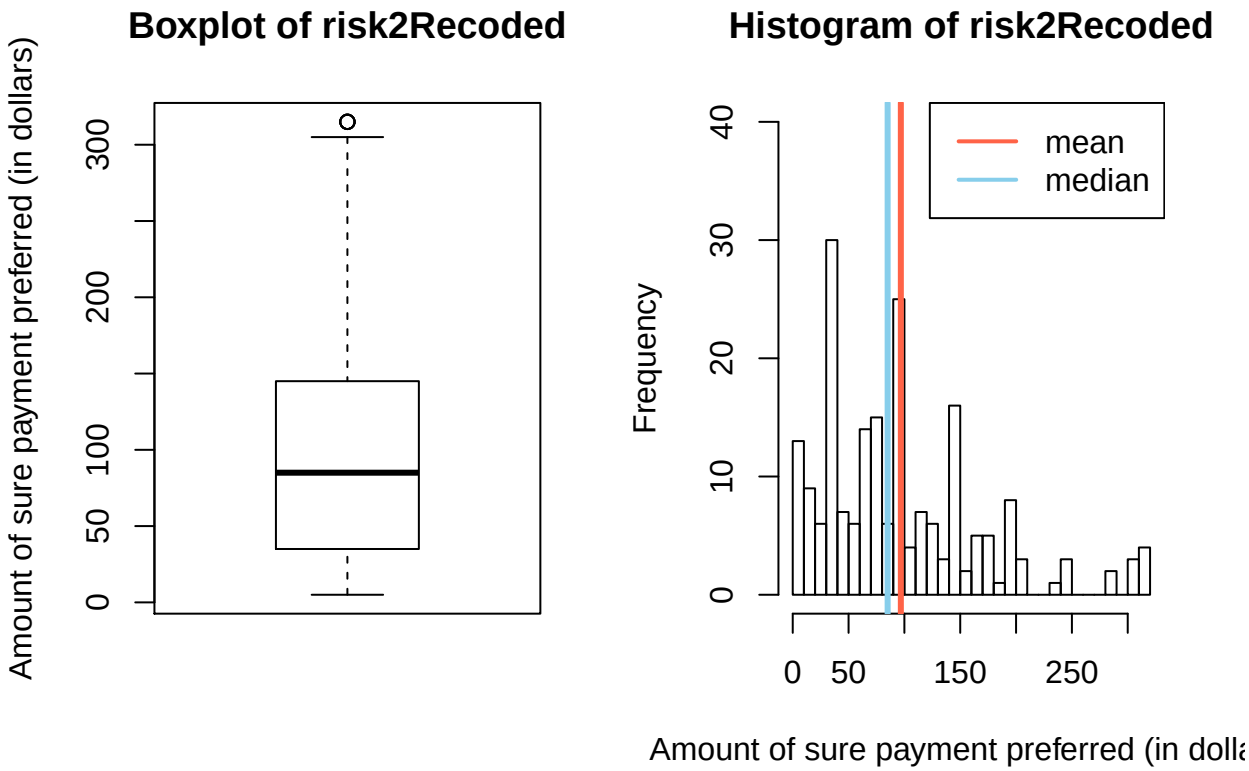

```
## $describe
##    vars   n mean   sd median trimmed  mad min max range  skew kurtosis
## X1    1 204 4.59 2.43      5    4.63 2.97   0  10    10 -0.07    -0.99
##      se
## X1 0.17
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.000   5.000   4.593   7.000  10.000
##
## $table
##
##  0  1  2  3  4  5  6  7  8  9 10
## 10 12 24 34 15 27 26 31 20  3  2
##
## $prop.table
##
##     0     1     2     3     4     5     6     7     8     9    10
## 0.049 0.059 0.118 0.167 0.074 0.132 0.127 0.152 0.098 0.015 0.010
```

risk2

risk2 uses a staircase type of questions, whether people prefer a 50/50 chance of receiving 300 dollars or nothing, or a certain amount of dollars as a sure payment. It has been coded to a scale from 0 to 32. For the description it is recoded to the original values of 5 to 315 dollars. It has a mean of 97 dollars, the median lies at 85.

Respondents show a strong tendency for risk-aversive behaviour. The standard deviation is 7.27 and the data is strongly right skewed shown by the value of 1.11 and has multiple peaks at approximately 40 dollars, 100 dollars, 150 dollars and 200 dollars. The kurtosis of 0.98 show a mesokurtic character of the data.

```
risk2Recoded <- risk2 * 10 - 5
describeVariable(risk2Recoded, lab = "Amount of sure payment preferred (in dollars)", ylim = c(0,40), hi
```




```
## $describe
##    vars   n  mean    sd median trimmed   mad min max range skew kurtosis
## X1    1 204 96.81 72.69     85   88.23 74.13   5 315   310 1.11     0.98
##      se
## X1 5.09
##
## $summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   35.00   85.00   96.81  145.00  315.00
```

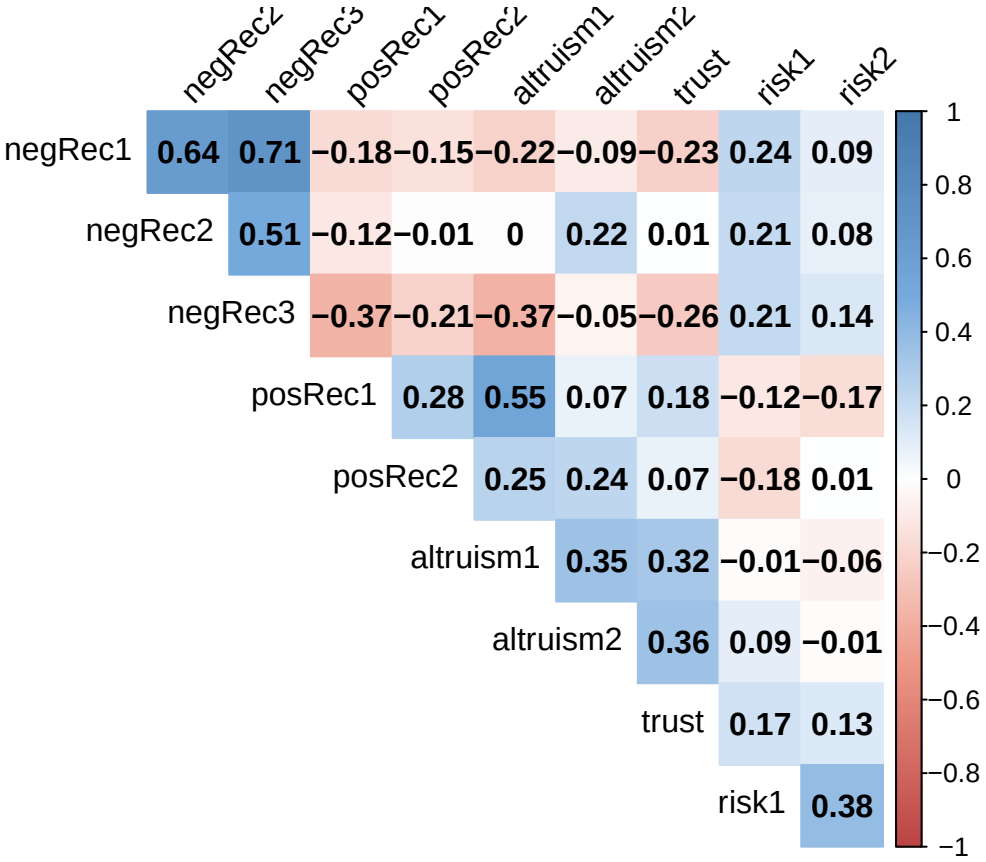
Multi-indicator concepts

As the correlation plot shows negRec has very high correlations in-between the negRec variables, just as altruism and risk. High correlations between and in-between preferences are found and this (multi-)collinearity could create a problem in linear regression. When collinearity is present it can be difficult to determine how each one variable is separately associated with the response variable. This reduces the accuracy of the estimates of the coefficients and causes the standard error to grow and thus could fail to reject valid null-hypothesis.

When possible, the variables are joined together. Another important reason is that these multi-indicator

concepts are not caught by one measure, but by the combination of measures that captures the underlying concepts. So it is preferred to combine the underlying concepts to provide a more internal reliable and full measure of the concept. To measure if the variables can be joined together Cronbach's Alpha is used.

```
preferences <- select(qualtrics, c(negRec1:risk2))
plotCor(preferences)
```




```
## $cor
##           negRec1 negRec2 negRec3 posRec1 posRec2 altruism1 altruism2
## negRec1      1.00    0.64    0.71   -0.18   -0.15     -0.22     -0.09
## negRec2      0.64    1.00    0.51   -0.12   -0.01      0.00      0.22
## negRec3      0.71    0.51    1.00   -0.37   -0.21     -0.37     -0.05
## posRec1     -0.18   -0.12   -0.37    1.00    0.28      0.55      0.07
## posRec2     -0.15   -0.01   -0.21    0.28    1.00      0.25      0.24
## altruism1   -0.22    0.00   -0.37    0.55    0.25      1.00      0.35
## altruism2   -0.09    0.22   -0.05    0.07    0.24      0.35      1.00
## trust       -0.23    0.01   -0.26    0.18    0.07      0.32      0.36
## risk1        0.24    0.21    0.21   -0.12   -0.18     -0.01      0.09
## risk2        0.09    0.08    0.14   -0.17    0.01     -0.06     -0.01
##           trust risk1 risk2
## negRec1   -0.23  0.24  0.09
## negRec2    0.01  0.21  0.08
## negRec3   -0.26  0.21  0.14
## posRec1    0.18 -0.12 -0.17
## posRec2    0.07 -0.18  0.01
## altruism1  0.32 -0.01 -0.06
## altruism2  0.36  0.09 -0.01
```

```
## trust      1.00  0.17  0.13
## risk1      0.17  1.00  0.38
## risk2      0.13  0.38  1.00
```

negRec

First the correlations and Cronbach's alpha of negRec is measured for the unscaled and scaled variables. The results show a Cronbach's Alpha of 0.81 for the unscaled negRec variables, which is higher than the threshold of 0.7. The scaled variables do not improve the Cronbach's Alpha, so the unscaled variables of negRec are joined together.

```
cbAlpha(negRec1, negRec2, negRec3)
```

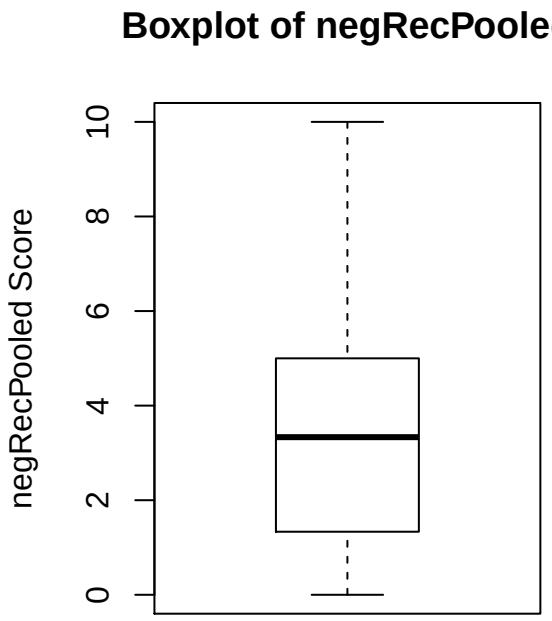
```
## $cor
##           negRec1   negRec2   negRec3
## negRec1 1.0000000 0.6372277 0.7113473
## negRec2 0.6372277 1.0000000 0.5118749
## negRec3 0.7113473 0.5118749 1.0000000
##
## $alpha
##
## Reliability analysis
## Call: alpha(x = data.frame(...))
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase mean  sd
##       0.81      0.81    0.76      0.59 4.3 0.023  3.4 2.4
##
##  lower alpha upper     95% confidence boundaries
## 0.76 0.81 0.86
##
##  Reliability if an item is dropped:
##         raw_alpha std.alpha G6(smc) average_r S/N alpha se
## negRec1      0.64      0.64    0.47      0.47 1.8    0.050
## negRec2      0.80      0.80    0.67      0.67 4.1    0.028
## negRec3      0.76      0.76    0.62      0.62 3.3    0.033
##
##  Item statistics
##           n raw.r std.r r.cor r.drop mean  sd
## negRec1 204  0.90  0.90  0.84   0.75  3.7 2.8
## negRec2 204  0.82  0.82  0.67   0.60  3.9 2.9
## negRec3 204  0.83  0.84  0.72   0.64  2.7 2.7
##
## $scaledAlpha
##
## Reliability analysis
## Call: alpha(x = scale(data.frame(...)))
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase    mean   sd
##       0.81      0.81    0.76      0.59 4.3 0.023 2.5e-17 0.85
##
##  lower alpha upper     95% confidence boundaries
## 0.77 0.81 0.86
##
##  Reliability if an item is dropped:
```

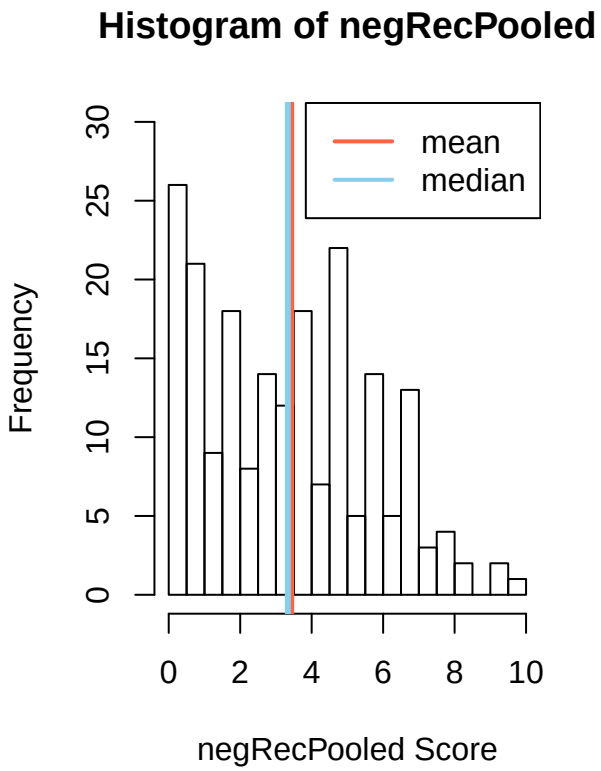
```
##         raw_alpha std.alpha G6(smc) average_r S/N alpha se
## negRec1      0.64      0.64    0.47      0.47 1.8    0.050
## negRec2      0.80      0.80    0.67      0.67 4.1    0.028
## negRec3      0.76      0.76    0.62      0.62 3.3    0.033
##
##  Item statistics
##           n raw.r std.r r.cor r.drop     mean sd
## negRec1 204  0.90  0.90  0.84   0.75 -2.5e-17  1
## negRec2 204  0.82  0.82  0.67   0.60 -1.7e-17  1
## negRec3 204  0.84  0.84  0.72   0.64  9.2e-17  1
```

negRecPooled

The joined variable is called negRecPooled. It has a mean of 3.43, the median lies at 3.33. the standard deviation is 2.39. negRecPooled shows a somewhat more symmetric character than the uncombined variables of negRec. Nonetheless, the data is somewhat right skewed with a higher tendency for lower scores. The data has a mesokurtic character, shown by the value of -0.77.

```
qualtrics$negRecPooled <- cbAlpha(negRec1, negRec2, negRec3)$alpha$scores
attach(qualtrics)
describeVariable(negRecPooled, lab = "negRecPooled Score", ylim = c(0,30), hist = TRUE)
```






```
## $describe
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 204 3.43 2.39   3.33    3.32 2.97   0  10    10 0.33    -0.77 0.17
##
## $summary
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.333   3.333   3.426   5.000  10.000
```

Re-order

Afterwards the former negRec variables are deleted and the order is adjusted.

```
qualtrics <- select(qualtrics, -c(negRec1:negRec3))
indexNegPooled <- grep("negRecPooled", names(qualtrics))
qualtrics <- qualtrics[c(1:5, indexNegPooled, 6:(indexNegPooled-1))]
names(qualtrics)
```

```
##  [1] "public"       "offer"        "respond"      "dilemma"
##  [5] "chicken"      "negRecPooled" "posRec1"      "posRec2"
##  [9] "altruism1"    "altruism2"    "trust"        "risk1"
## [13] "risk2"
```

posRec

Next positive reciprocity is inspected. A low Cronbach's alpha is found of 0.47, even after scaling the variables it barely improved. Due the lack of internal reliability it is decided to keep the variables apart.

```
cbAlpha(posRec1, posRec2)
```

```
## $cor
##           posRec1   posRec2
## posRec1 1.0000000 0.2844932
## posRec2 0.2844932 1.0000000
##
## $alpha
##
## Reliability analysis
## Call: alpha(x = data.frame(...))
##
##   raw_alpha std.alpha G6(smc) average_r  S/N   ase mean  sd
##       0.47      0.48    0.31      0.31 0.91 0.073  5.7 1.6
##
##  lower alpha upper     95% confidence boundaries
## 0.33 0.47 0.62
##
##  Reliability if an item is dropped:
##         raw_alpha std.alpha G6(smc) average_r S/N alpha se
## posRec1      0.31      0.31   0.098      0.31  NA       NA
## posRec2      0.31      0.31   0.098      0.31  NA       NA
##
##  Item statistics
##           n raw.r std.r r.cor r.drop mean  sd
## posRec1 204  0.84  0.81  0.45   0.31  8.3 2.1
## posRec2 204  0.78  0.81  0.45   0.31  3.0 1.8
##
## $scaledAlpha
##
## Reliability analysis
## Call: alpha(x = scale(data.frame(...)))
##
```

```
##   raw_alpha std.alpha G6(smc) average_r  S/N   ase     mean   sd
##       0.48      0.48    0.31      0.31 0.91 0.073 -1.3e-16 0.81
##
##  lower alpha upper     95% confidence boundaries
## 0.33 0.48 0.62
##
##  Reliability if an item is dropped:
##         raw_alpha std.alpha G6(smc) average_r S/N alpha se
## posRec1      0.31      0.31   0.098      0.31  NA       NA
## posRec2      0.31      0.31   0.098      0.31  NA       NA
##
##  Item statistics
##           n raw.r std.r r.cor r.drop     mean sd
## posRec1 204  0.81  0.81  0.45   0.31 -3.8e-16  1
## posRec2 204  0.81  0.81  0.45   0.31  1.2e-16  1
```

Altruism

After this altruism is inspected and a very low Cronbach's alpha is found of 0.01, after scaling the variables the Cronbach's alpha increases to 0.34. Since outliers of altruism2 were found and to prevent these from influencing to much altruism2 is log transformed, this significantly improves the Cronbach's alpha to 0.58, but is still to low to join the variables together. Due the lack of internal reliability it is decided to keep the variables apart.

```
cbAlpha(altruism1, altruism2)
```

```
## $cor
##           altruism1 altruism2
## altruism1 1.0000000 0.3524847
## altruism2 0.3524847 1.0000000
##
## $alpha
##
## Reliability analysis
## Call: alpha(x = data.frame(...))
##
##   raw_alpha std.alpha G6(smc) average_r  S/N   ase mean sd
##      0.012      0.34    0.21      0.21 0.53 0.004   61 83
##
##  lower alpha upper     95% confidence boundaries
## 0 0.01 0.02
##
##  Reliability if an item is dropped:
##           raw_alpha std.alpha G6(smc) average_r S/N alpha se
## altruism1      0.21      0.21   0.043      0.21  NA       NA
## altruism2      0.21      0.21   0.043      0.21  NA       NA
##
##  Item statistics
##             n raw.r std.r r.cor r.drop  mean    sd
## altruism1 204  0.22  0.78  0.35   0.21   7.1   2.4
## altruism2 204  1.00  0.78  0.35   0.21 115.6 165.8
##
## $scaledAlpha
##
```

```
## Reliability analysis
## Call: alpha(x = scale(data.frame(...)))
##
##   raw_alpha std.alpha G6(smc) average_r  S/N   ase    mean   sd
##       0.34      0.34    0.21      0.21 0.53 0.092 7.8e-17 0.78
##
##  lower alpha upper     95% confidence boundaries
## 0.16 0.34 0.52
##
##  Reliability if an item is dropped:
##           raw_alpha std.alpha G6(smc) average_r S/N alpha se
## altruism1      0.21      0.21   0.043      0.21  NA       NA
## altruism2      0.21      0.21   0.043      0.21  NA       NA
##
##  Item statistics
##             n raw.r std.r r.cor r.drop     mean sd
## altruism1 204  0.78  0.78  0.35   0.21  1.8e-16  1
## altruism2 204  0.78  0.78  0.35   0.21 -1.4e-17  1
```

```
altruism2Log <- log(ifelse(altruism2 == 0, 1, altruism2))
cbAlpha(altruism1, altruism2Log)
```

```
## $cor
##              altruism1 altruism2Log
## altruism1    1.0000000    0.3524847
## altruism2Log 0.3524847    1.0000000
##
## $alpha
##
## Reliability analysis
## Call: alpha(x = data.frame(...))
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase mean  sd
##       0.57      0.58     0.4       0.4 1.4 0.059  5.3 1.9
##
##  lower alpha upper     95% confidence boundaries
## 0.46 0.57 0.69
##
##  Reliability if an item is dropped:
##              raw_alpha std.alpha G6(smc) average_r S/N alpha se
## altruism1          0.4       0.4    0.16       0.4  NA       NA
## altruism2Log       0.4       0.4    0.16       0.4  NA       NA
##
##  Item statistics
##                n raw.r std.r r.cor r.drop mean  sd
## altruism1    204  0.86  0.84  0.53    0.4  7.1 2.4
## altruism2Log 204  0.81  0.84  0.53    0.4  3.4 2.1
##
## $scaledAlpha
##
## Reliability analysis
## Call: alpha(x = scale(data.frame(...)))
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase    mean   sd
##       0.58      0.58     0.4       0.4 1.4 0.059 8.4e-17 0.84
```

```
##
##  lower alpha upper     95% confidence boundaries
## 0.46 0.58 0.69
##
##  Reliability if an item is dropped:
##              raw_alpha std.alpha G6(smc) average_r S/N alpha se
## altruism1          0.4       0.4    0.16       0.4  NA       NA
## altruism2Log       0.4       0.4    0.16       0.4  NA       NA
##
##  Item statistics
##                n raw.r std.r r.cor r.drop     mean sd
## altruism1    204  0.84  0.84  0.53    0.4  1.8e-16  1
## altruism2Log 204  0.84  0.84  0.53    0.4 -2.1e-17  1
```

Risk

Next risk is inspected. However a low Cronbach's alpha of 0.37 is found, the scaled variables increases the Cronbach's alpha to 0.57, Due the lack of internal reliability it is decided to keep the variables apart.

```
cbAlpha(risk1, risk2)
```

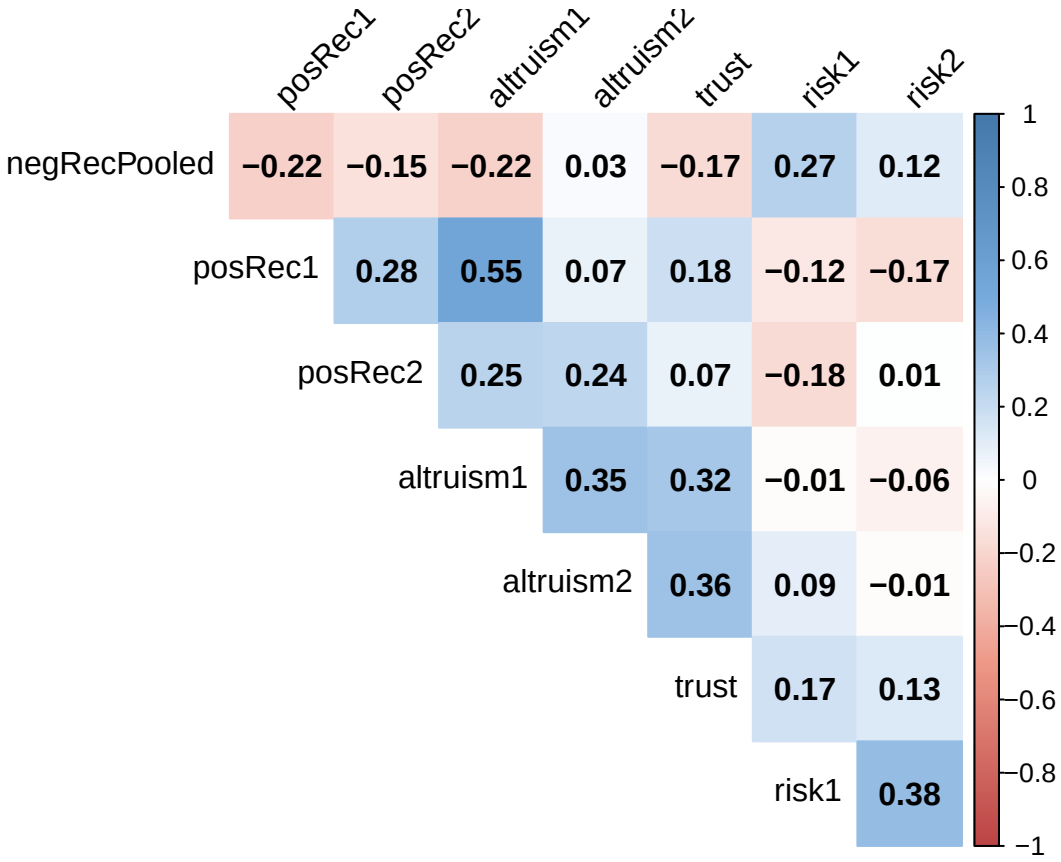
```
## $cor
##           risk1     risk2
## risk1 1.0000000 0.3847468
## risk2 0.3847468 1.0000000
##
## $alpha
##
## Reliability analysis
## Call: alpha(x = data.frame(...))
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase mean  sd
##       0.37      0.54    0.37      0.37 1.2 0.051  7.4 4.2
##
##  lower alpha upper     95% confidence boundaries
## 0.27 0.37 0.47
##
##  Reliability if an item is dropped:
##       raw_alpha std.alpha G6(smc) average_r S/N alpha se
## risk1      0.37      0.37    0.14      0.37  NA       NA
## risk2      0.37      0.37    0.14      0.37  NA       NA
##
##  Item statistics
##         n raw.r std.r r.cor r.drop mean  sd
## risk1 204  0.61  0.83  0.51   0.37  4.6 2.4
## risk2 204  0.96  0.83  0.51   0.37 10.2 7.3
##
## $scaledAlpha
##
## Reliability analysis
## Call: alpha(x = scale(data.frame(...)))
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase  mean   sd
##       0.54      0.54    0.37      0.37 1.2 0.064 6e-17 0.83
```

```
##
##  lower alpha upper     95% confidence boundaries
## 0.42 0.54 0.67
##
##  Reliability if an item is dropped:
##       raw_alpha std.alpha G6(smc) average_r S/N alpha se
## risk1      0.37      0.37    0.14      0.37  NA       NA
## risk2      0.37      0.37    0.14      0.37  NA       NA
##
##  Item statistics
##         n raw.r std.r r.cor r.drop     mean sd
## risk1 204  0.83  0.83  0.51   0.37  1.5e-16  1
## risk2 204  0.83  0.83  0.51   0.37 -2.4e-17  1
```

Joined correlations

After joining the variables, correlations are still present but the highest correlations of negRec are solved. Nonetheless, collinearity might still be an issue.

```
preferences <- model.matrix(~ ., data = qualtrics)[,-c(1:6)]
plotCor(preferences)
```




```
## $cor
##              negRecPooled posRec1 posRec2 altruism1 altruism2 trust risk1
## negRecPooled         1.00   -0.22   -0.15     -0.22      0.03 -0.17  0.27
## posRec1             -0.22    1.00    0.28      0.55      0.07  0.18 -0.12
```

```
## posRec2             -0.15    0.28    1.00      0.25      0.24  0.07 -0.18
## altruism1           -0.22    0.55    0.25      1.00      0.35  0.32 -0.01
## altruism2            0.03    0.07    0.24      0.35      1.00  0.36  0.09
## trust               -0.17    0.18    0.07      0.32      0.36  1.00  0.17
## risk1                0.27   -0.12   -0.18     -0.01      0.09  0.17  1.00
## risk2                0.12   -0.17    0.01     -0.06     -0.01  0.13  0.38
##              risk2
## negRecPooled  0.12
## posRec1      -0.17
## posRec2       0.01
## altruism1    -0.06
## altruism2    -0.01
## trust         0.13
## risk1         0.38
## risk2         1.00
```

Model building

Test set

A test set is created to avoid data snooping and overfitting, and to have an extra test set to measure final performance. The data is split in 6 folds. 1 separate test fold and 5 remaining trainings fold, which is used to perform cross-validation. The test set consists of 34 observations.

```
set.seed(256)
test <- sample(1:nrow(qualtrics), size = nrow(qualtrics) / 6)
testQualtrics <- qualtrics[test,]
dim(testQualtrics)
```

```
## [1] 34 13
```

Train set

The training set is used for cross-validation. However, for some basic exploration it is sometimes too cumbersome and in that case all trainings data is used. The train set consists of 170 observations. Three datasets are created one with all variables included, one with only the preferences and risk variables and a z-scaled dataset of all preferences and risk.

```
trainQualtrics <- qualtrics[-test,]
dim(trainQualtrics)
```

```
## [1] 170  13
```

```
trainPreferences <- preferences[-test,]
dim(trainPreferences)
```

```
## [1] 170   8
```

```
trainPreferencesScaled <- data.frame(scale(trainPreferences))
trainPreferencesScaled <- model.matrix(~ ., data = trainPreferencesScaled)
```

Cross-validation

5-fold cross-validation is used for training the data, to get more valid results and to prevent overfitting. The foldid's used are used in all models whenever possible to make sure models are comporable with each other,

but this is not possible in the boosting method. The final result is 5 validation folds of 34 observations and a separate sixth test set of 34 observations, the test set.

```
set.seed(128)
foldid <- sample(rep(1:5, length = nrow(trainQualtrics)))
table(foldid)
```

```
## foldid
##  1  2  3  4  5
## 34 34 34 34 34
```

Public goods

First a subset of trainQualtrics is taken with only public good as dependent variable and the preferences, all the other games are excluded.

```
trainPublic <- select(trainQualtrics, -c(offer, respond, dilemma, chicken))
attach(trainPublic)
```

Bivariate analysis

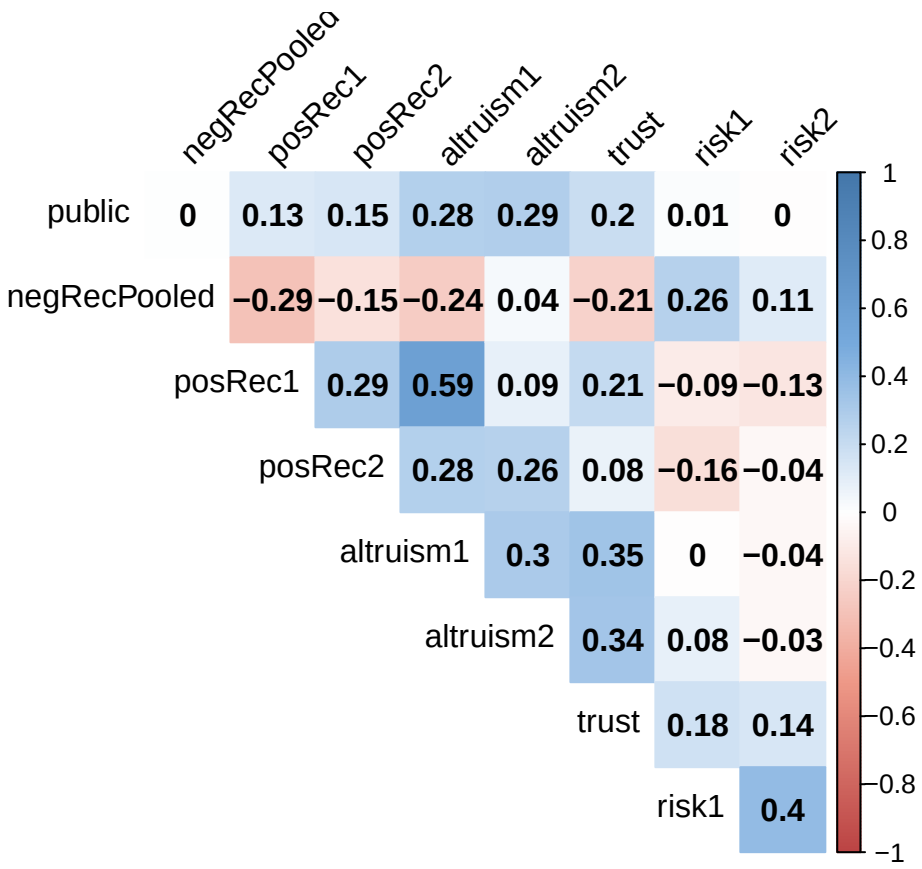
It must be noted that these are just bivariate relationships and only contain a subset of all the data and do not account for any relations accounted by confounding variables because of the collinearity between the predictors.

Spearman's rang correlation is chosen because most of these variables are asymmetric and mesokurtic or leptokurtic and Spearman's rang correlation is a non-parametric method, which makes no assumptions based on normality and is less influenced by outliers.

The following (absolute) correlations of greater than 0.10 are found:

- Positive relationship with posRec1 (0.13).
- Positive relationship with posRec2 (0.15).
- Positive relationship with altruism1 (0.28).
- Positive relationship with altruism2 (0.29).
- Positive relationship with trust (0.2).

```
plotCor(trainPublic)
```




```
## $cor
##              public negRecPooled posRec1 posRec2 altruism1 altruism2 trust
## public         1.00         0.00    0.13    0.15      0.28      0.29  0.20
## negRecPooled   0.00         1.00   -0.29   -0.15     -0.24      0.04 -0.21
## posRec1        0.13        -0.29    1.00    0.29      0.59      0.09  0.21
## posRec2        0.15        -0.15    0.29    1.00      0.28      0.26  0.08
## altruism1      0.28        -0.24    0.59    0.28      1.00      0.30  0.35
## altruism2      0.29         0.04    0.09    0.26      0.30      1.00  0.34
## trust          0.20        -0.21    0.21    0.08      0.35      0.34  1.00
## risk1          0.01         0.26   -0.09   -0.16      0.00      0.08  0.18
## risk2          0.00         0.11   -0.13   -0.04     -0.04     -0.03  0.14
##              risk1 risk2
## public        0.01  0.00
## negRecPooled  0.26  0.11
## posRec1      -0.09 -0.13
## posRec2      -0.16 -0.04
## altruism1     0.00 -0.04
## altruism2     0.08 -0.03
## trust         0.18  0.14
## risk1         1.00  0.40
## risk2         0.40  1.00
```

Base model

A base model is conducted using the basemodelcv function, it calculates the mean of the four training folds and then uses this mean as prediction for the remaining validation fold, repeated for all folds. A cross-validated MSE score of 11.36, a standard deviation of 2.57 and a RMSE of 3.37.

```
baseline <- baseModelcv(trainPublic, foldid)
baseline
```

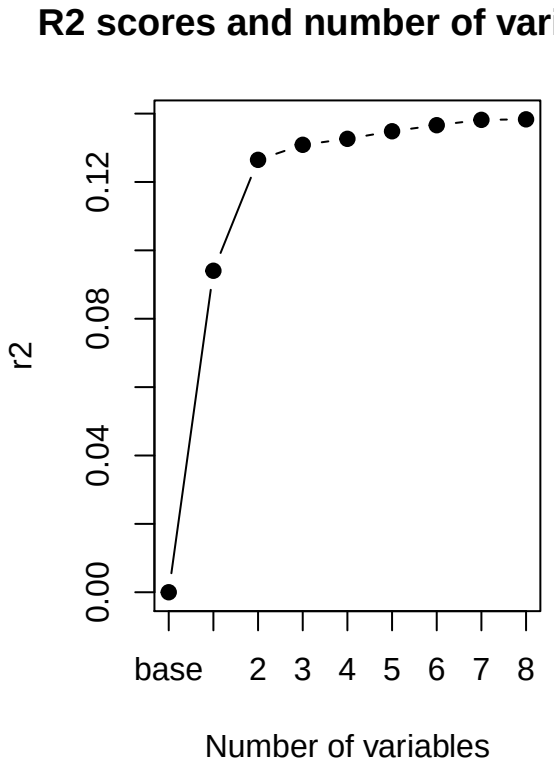
```
## $cv
## [1] 10.899870 10.887760  8.147978 11.584991 15.304985
##
## $meancv
## [1] 11.36512
##
## $sdcv
## [1] 2.567353
##
## $rmse
## [1] 3.371219
```

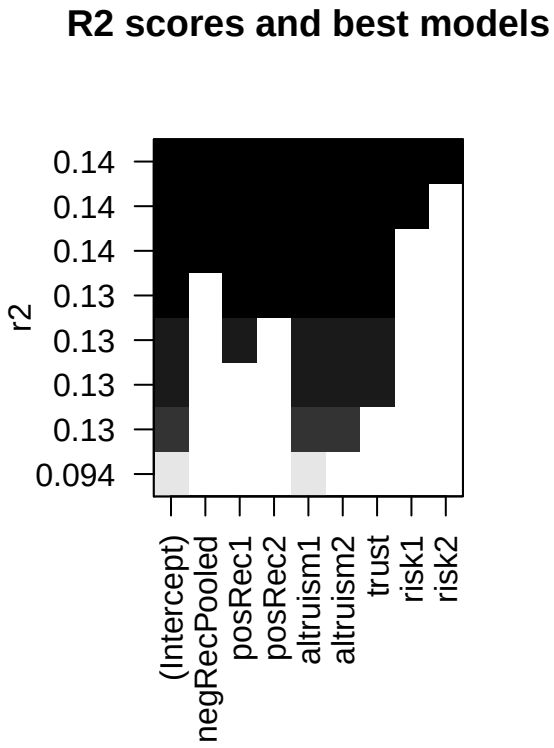
```
mseBase <- baseline$meancv
```

Best subset selection

A best subset selection is performed on all trainings data and then the R2 score is used to chose the best model for every number of variables. As the plot shows altruism1 explains a little more than 9% of all the variance in the model. Adding a second predictor altrusim2 explains approximately 13% of all variance. After those two variables, little increase is gained.

```
bestPublic <- regsubsets(public ~ ., data = trainPublic)
plotR2Subs(bestPublic)
```






```
## Subset selection object
## Call: regsubsets.formula(public ~ ., data = trainPublic)
## 8 Variables  (and intercept)
##              Forced in Forced out
## negRecPooled     FALSE      FALSE
## posRec1          FALSE      FALSE
## posRec2          FALSE      FALSE
## altruism1        FALSE      FALSE
## altruism2        FALSE      FALSE
## trust            FALSE      FALSE
## risk1            FALSE      FALSE
## risk2            FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          negRecPooled posRec1 posRec2 altruism1 altruism2 trust risk1
## 1  ( 1 ) " "          " "     " "     "*"       " "       " "   " "
## 2  ( 1 ) " "          " "     " "     "*"       "*"       " "   " "
## 3  ( 1 ) " "          " "     " "     "*"       "*"       "*"   " "
## 4  ( 1 ) " "          "*"     " "     "*"       "*"       "*"   " "
## 5  ( 1 ) " "          "*"     "*"     "*"       "*"       "*"   " "
## 6  ( 1 ) "*"          "*"     "*"     "*"       "*"       "*"   " "
## 7  ( 1 ) "*"          "*"     "*"     "*"       "*"       "*"   "*"
## 8  ( 1 ) "*"          "*"     "*"     "*"       "*"       "*"   "*"
##          risk2
## 1  ( 1 ) " "
## 2  ( 1 ) " "
```

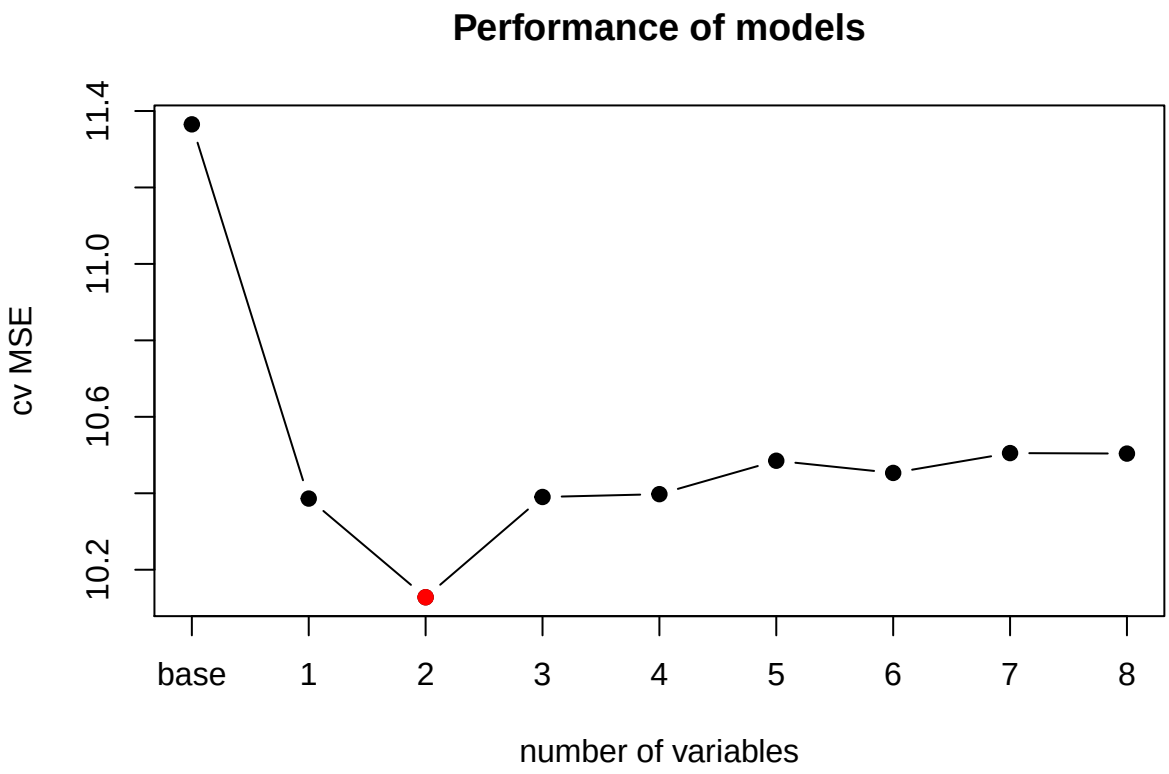
```
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) " "
## 7  ( 1 ) " "
## 8  ( 1 ) "*"
```

A best subset selection cross-validation is conducted using the bestsubsetcv function. It applies the regsubsets function and trains the best model for that specific number of variables on four folds and uses that model to predict the remaining validation fold, repeated for all folds. The best performing model has two variables, as expected reviewing the R2 score above. These number of variables correspond with a MSE of 10.12, a standard deviation of 2.36 and a RMSE of 3.18.

```
bestSubs <- bestSubsetcv(trainPublic, foldid)
bestSubs
```

```
## $cv
##              1         2         3         4         5         6         7
## [1,]  9.541425  8.946933  9.041989  8.973390  9.016817  8.927112  8.864252
## [2,]  9.744439  9.708939  9.807149  9.863480  9.998496  9.970131  9.925181
## [3,]  7.079632  7.027789  6.911953  6.953092  6.946279  6.891176  6.891841
## [4,] 12.035254 12.306052 13.144826 13.147174 13.305978 13.236130 13.403697
## [5,] 13.529985 12.650530 13.045633 13.051358 13.158048 13.241821 13.440922
##              8
## [1,]  8.893743
## [2,]  9.927013
## [3,]  6.909851
## [4,] 13.402465
## [5,] 13.386168
##
## $meancv
##        1        2        3        4        5        6        7        8
## 10.38615 10.12805 10.39031 10.39770 10.48512 10.45327 10.50518 10.50385
##
## $sdcv
##        1        2        3        4        5        6        7        8
## 2.483025 2.360557 2.687709 2.682391 2.739429 2.773600 2.876858 2.852707
##
## $bestnmdl
## [1] 2
##
## $bestnmean
## [1] 10.12805
##
## $bestnsd
## [1] 2.360557
##
## $bestnrmse
## [1] 3.18246
```

```
mseBestSub <- bestSubs$bestnmean
plotModels(bestSubs$meancv, baseline)
```




```
##     base        1        2        3        4        5        6        7
## 11.36512 10.38615 10.12805 10.39031 10.39770 10.48512 10.45327 10.50518
##        8
## 10.50385
```

The variables in the best model are altruism1 and altruism2, which confirms the findings of the R2 plot. The corresponding coefficients are 0.39 for altruism1 and 0.0038 for altruism2. This means that if the altruism1 score increases with one, the amount invested in the public funds increases on average with 0.39. For every dollar gifted to a good cause in altruism2, the amount invested in the public fund increases on average with 0.0038. The model as a whole is significant, furthermore altruism1 is very significant (<0.001) and altruism2 is moderately significant and has a p-value of 0.0137. Since altruism2 is on a different scale the standardized coefficients are calculated to compare the coefficients, in that case altruism1 has a coefficient of 0.28 and altruism2 of 0.18.

```
bestSubMdl <- public ~ altruism1 + altruism2
bestSubFit <- lm(bestSubMdl, data = trainPublic)
summary(bestSubFit)
```

```
##
## Call:
## lm(formula = bestSubMdl, data = trainPublic)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.2445 -2.3115 -0.7068  2.6113  6.6265
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.432449   0.770900   1.858 0.064908 .
## altruism1   0.389341   0.102954   3.782 0.000217 ***
## altruism2   0.003837   0.001541   2.490 0.013737 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.172 on 167 degrees of freedom
## Multiple R-squared:  0.1265, Adjusted R-squared:  0.116
## F-statistic: 12.09 on 2 and 167 DF,  p-value: 1.249e-05
```

```
lm.beta(bestSubFit)
```

```
## altruism1 altruism2
## 0.2771497 0.1825146
```

Lasso

The lasso model is performed, it uses cross-validation to find the best lambda. The best performing model has a lambda of 0.19 and a corresponding MSE of 10.27 and a RMSE of 3.18, which is worse than the best subset selection method. However, the standard deviation of 1.15 is a lot lower compared to the previous models and is thus more stable. As the plot shows, the optimal lambda includes four variables, however the lambda plus 1 standard error is corresponds with the base model, this shows that the difference isnt't very substantial.

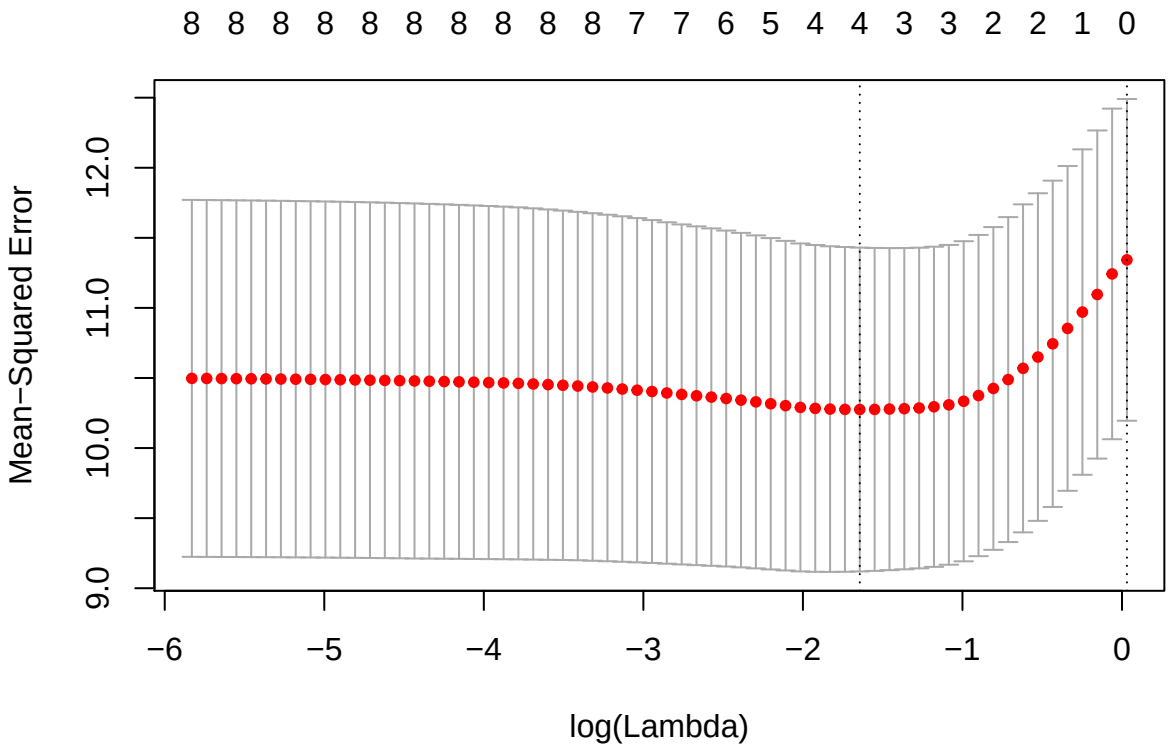
```
lassocv <- cv.glmnet(trainPreferences, public, alpha = 1, foldid = foldid)
resultsLasso(lassocv)
```

```
## $glmnet.fit
##
## Call:  glmnet(x = trainPreferences, y = public, alpha = 1)
##
##       Df    %Dev   Lambda
##  [1,]  0 0.00000 1.031000
##  [2,]  1 0.01597 0.939900
##  [3,]  1 0.02922 0.856400
##  [4,]  1 0.04022 0.780300
##  [5,]  2 0.04956 0.711000
##  [6,]  2 0.06262 0.647800
##  [7,]  2 0.07346 0.590300
##  [8,]  2 0.08246 0.537800
##  [9,]  2 0.08993 0.490000
## [10,]  2 0.09614 0.446500
## [11,]  2 0.10130 0.406800
## [12,]  3 0.10630 0.370700
## [13,]  3 0.11050 0.337800
## [14,]  3 0.11390 0.307800
## [15,]  3 0.11680 0.280400
## [16,]  3 0.11920 0.255500
## [17,]  3 0.12120 0.232800
## [18,]  3 0.12280 0.212100
## [19,]  4 0.12430 0.193300
## [20,]  4 0.12560 0.176100
## [21,]  4 0.12680 0.160500
## [22,]  4 0.12770 0.146200
```

```
## [23,]  4 0.12850 0.133200
## [24,]  4 0.12920 0.121400
## [25,]  5 0.13010 0.110600
## [26,]  5 0.13080 0.100800
## [27,]  5 0.13150 0.091830
## [28,]  6 0.13210 0.083670
## [29,]  7 0.13300 0.076230
## [30,]  7 0.13390 0.069460
## [31,]  7 0.13460 0.063290
## [32,]  7 0.13520 0.057670
## [33,]  7 0.13570 0.052550
## [34,]  7 0.13610 0.047880
## [35,]  7 0.13650 0.043620
## [36,]  7 0.13680 0.039750
## [37,]  8 0.13700 0.036220
## [38,]  8 0.13720 0.033000
## [39,]  8 0.13740 0.030070
## [40,]  8 0.13760 0.027400
## [41,]  8 0.13770 0.024960
## [42,]  8 0.13780 0.022750
## [43,]  8 0.13790 0.020730
## [44,]  8 0.13800 0.018880
## [45,]  8 0.13800 0.017210
## [46,]  8 0.13810 0.015680
## [47,]  8 0.13810 0.014290
## [48,]  8 0.13820 0.013020
## [49,]  8 0.13820 0.011860
## [50,]  8 0.13820 0.010810
## [51,]  8 0.13820 0.009846
## [52,]  8 0.13820 0.008971
## [53,]  8 0.13830 0.008174
## [54,]  8 0.13830 0.007448
## [55,]  8 0.13830 0.006787
## [56,]  8 0.13830 0.006184
## [57,]  8 0.13830 0.005634
## [58,]  8 0.13830 0.005134
## [59,]  8 0.13830 0.004678
## [60,]  8 0.13830 0.004262
## [61,]  8 0.13830 0.003884
## [62,]  8 0.13830 0.003539
## [63,]  8 0.13830 0.003224
## [64,]  8 0.13830 0.002938
## [65,]  8 0.13830 0.002677
## [66,]  8 0.13830 0.002439
##
## $bestlambda
## [1] 0.193283
##
## $bestnmean
## [1] 10.27502
##
## $bestnsd
## [1] 1.155274
##
```

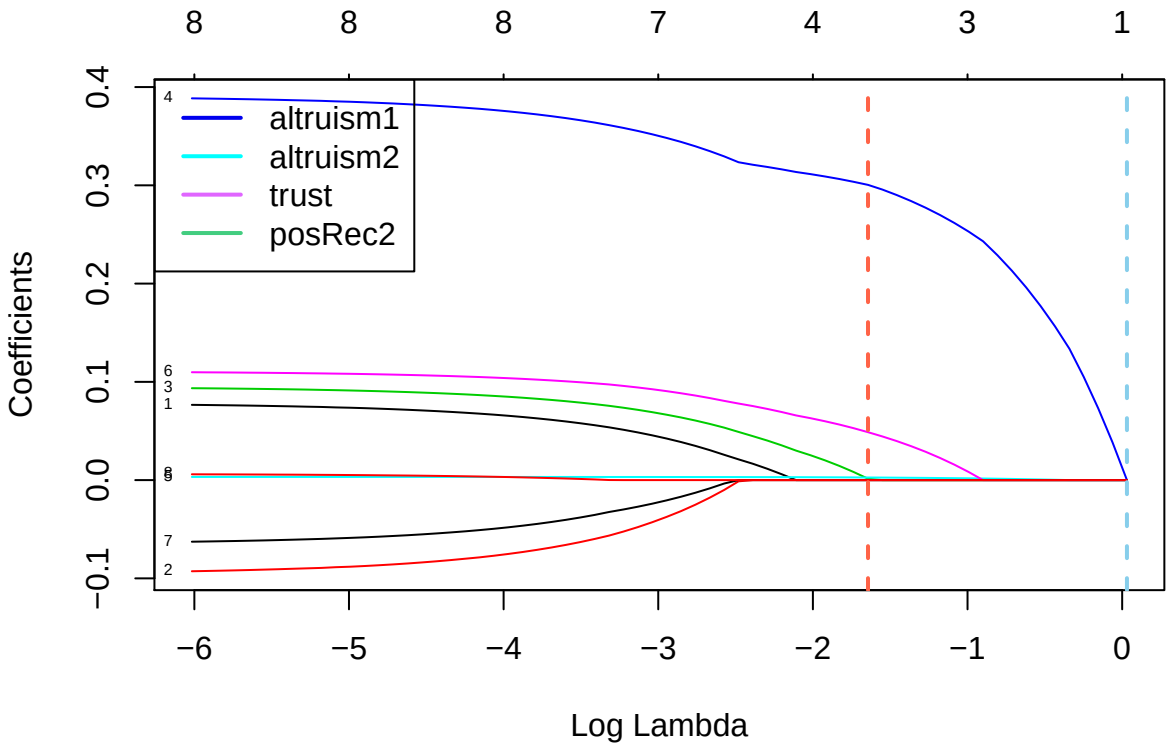
```
## $bestnrmse
## [1] 3.205467
##
## $lambda1se
## [1] 1.031493
```

```
mseLasso <- min(lassocv$cvm)
plot(lassocv)
```



The shrinkage plot shows the shrank coefficients for all variables, the red dotted line corresponds with the best lambda and the blue line corresponds with the lambda plus 1 standard error. The plot shows that the best lambda altruism1, altruism2, trust and posRec2. Where the first two were also found via best subset selection method. The coefficient that corresponds with this penalty factor can be seen underneath. posRec2 is just barely included.

```
plotShrinkage(trainPreferences, public, lassocv)
legend('topleft', legend = c("altruism1", "altruism2", "trust", "posRec2"),
  lwd = 2, col= c("blue2", "cyan", "mediumorchid1", "seagreen3"))
```

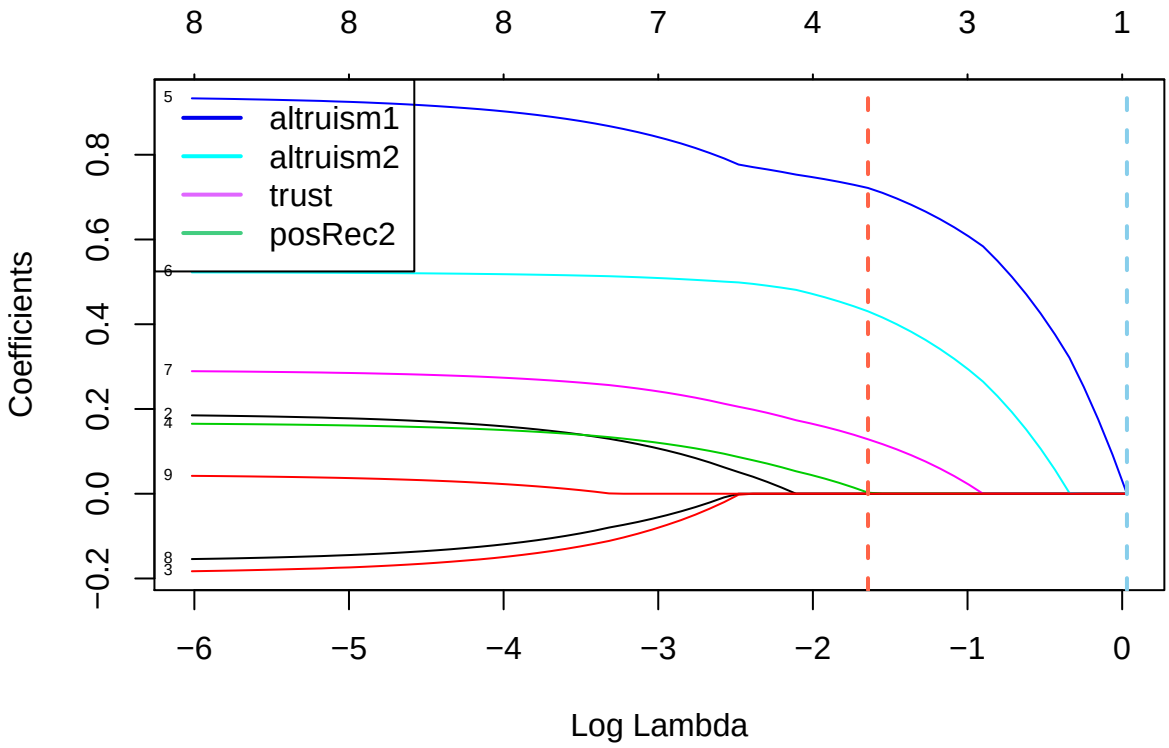



```
predict(lassocv, type = "coefficients", s = lassocv$lambda.min)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  1.930623219
## negRecPooled .
## posRec1      .
## posRec2      0.001430255
## altruism1    0.300467463
## altruism2    0.002682396
## trust        0.048786926
## risk1        .
## risk2        .
```

This next shrinkage plot shows the same coefficients but then on a scaled dataset, now the coefficients can be compared. Altruism1 and altruism2 have the highest coefficients, which corresponds with the findings of the best subset selection method. This is followed by trust, which also had high bivariate correlation, and posRec2 is barely included.

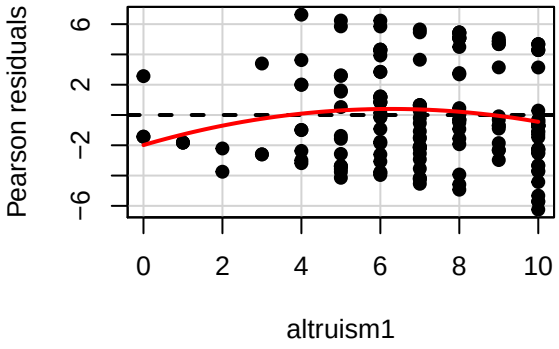
```
plotShrinkage(trainPreferencesScaled, public, lassocv)
legend('topleft', legend = c("altruism1", "altruism2", "trust", "posRec2"),
  lwd = 2, col= c("blue2", "cyan", "mediumorchid1", "seagreen3"))
```

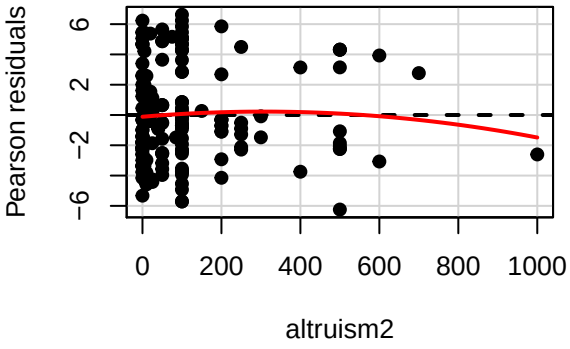


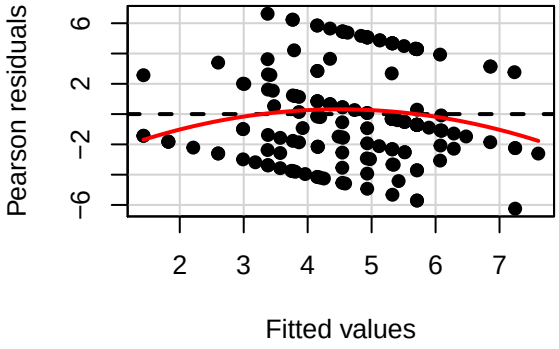
Non-linearity

Non-linear relations for the best subset model are examined. The residual plot shows some non-linearity mostly prevalent in altruism1, this is confirmed by the p-value of adding second degree term for altruism1 of 0.055, this is on the border of significance.

```
residualPlots((bestSubFit),pch=19)
```



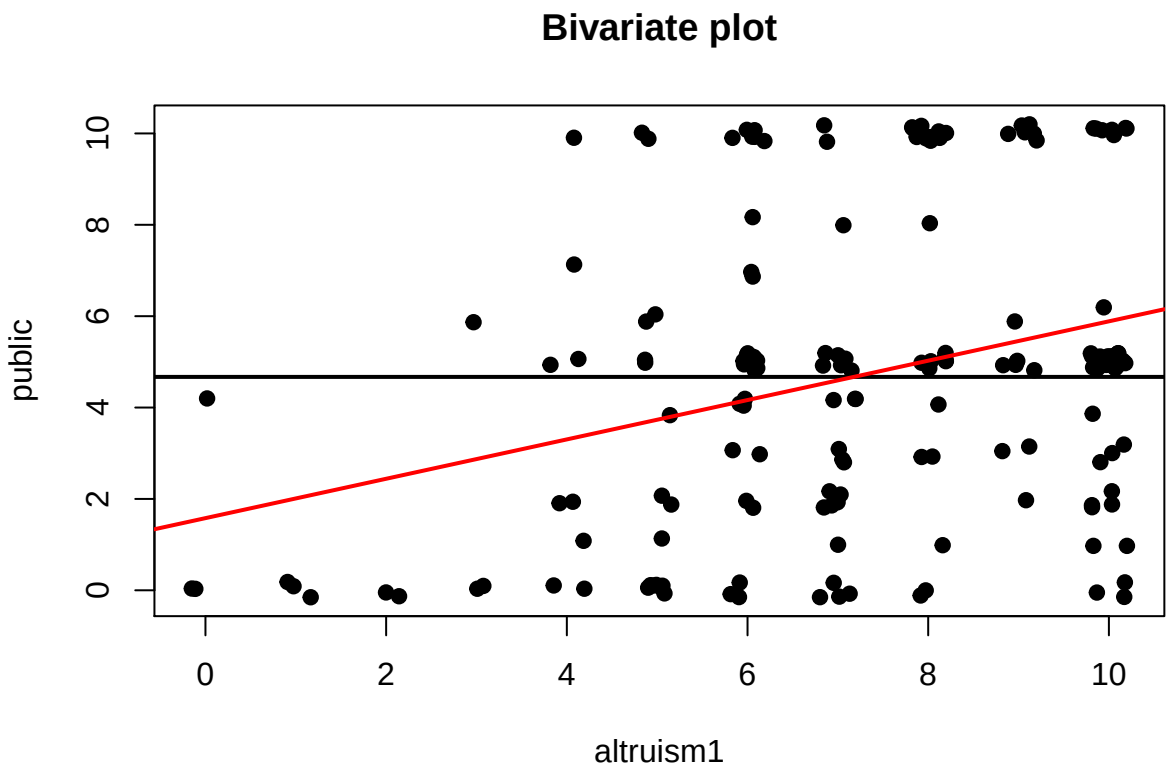





```
##            Test stat Pr(>|t|)
## altruism1     -1.935    0.055
## altruism2     -0.669    0.504
## Tukey test    -2.249    0.025
```

As mentioned before adding another second degree term would be almost significant and thus might improve cross-validated MSE and better fit the true relationship. Inspecting the scatterplot of public and altruism1 seems to confirm this suspicion. However, the plot is an simplified approximation of the true relationship because it only is based on one variable.

```
plotBivariate(altruism1, public)
```



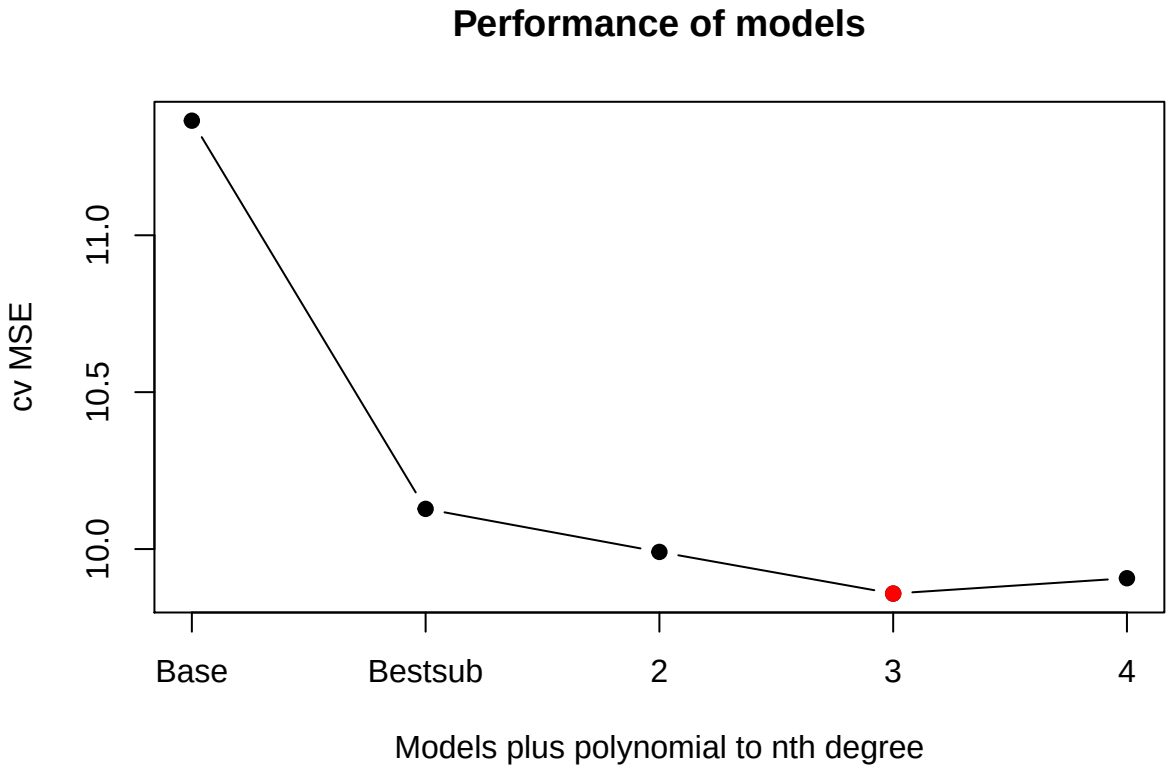
Multiple models to the 4th degree polynomial term of altruism1 are created. First a cross-validated non-linear model is applied to test performance. The best performing model contains a polynomial of altruism to the third degree and altruism2. It has an MSE of 9.85 and a RMSE of 3.14. The standard deviation is 2.25 which is even lower than then the original best subset selection method. So, even though the model is more complex the results are more stable, but less stable than lasso.

```
mdl2 <- public ~ poly(altruism1, 2) + altruism2
mdl3 <- public ~ poly(altruism1, 3) + altruism2
mdl4 <- public ~ poly(altruism1, 4) + altruism2
models <- c(bestSubMdl, mdl2, mdl3, mdl4)

polynomial <- nonLinearcv(trainPublic, foldid, models)
polynomial
```

```
## $cv
##              1         2         3         4
## [1,]  8.946933  9.251000  9.004945  8.989257
## [2,]  9.708939  9.356797  9.174472  9.143206
## [3,]  7.027789  6.736003  6.911291  6.878363
## [4,] 12.306052 12.275782 11.754639 11.708331
## [5,] 12.650530 12.335053 12.445107 12.815973
##
## $meancv
##         1         2         3         4
## 10.128049  9.990927  9.858091  9.907026
##
## $sdcv
```

```
##        1        2        3        4
## 2.360557 2.359006 2.245460 2.361419
##
## $bestnmdl
## [1] 3
##
## $bestnmean
## [1] 9.858091
##
## $bestnsd
## [1] 2.24546
##
## $bestnrmse
## [1] 3.13976
```

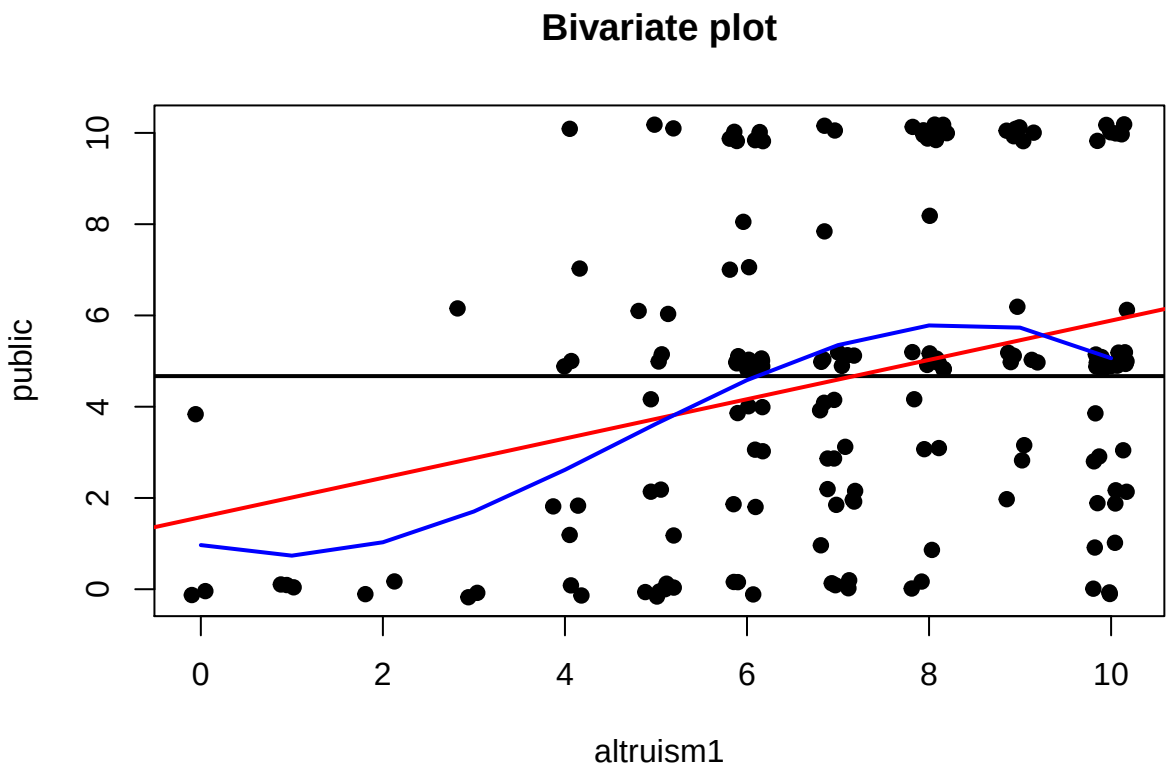
```
msePoly <- polynomial$bestnmean

plotModels(polynomial$meancv, baseline, xlab = "Models plus polynomial to nth degree", axislabels = c("
```




```
##      base         1         2         3         4
## 11.365117 10.128049  9.990927  9.858091  9.907026
```

As the bivariate plot shows, a polynomial to the third degree seems to better fit the data.

```
poly3Altruism1 <- lm(public ~ poly(altruism1, 3), data = trainPublic)
preds <- predict(poly3Altruism1, newdata = list(altruism1 = 0:max(altruism1)))
plotBivariate(altruism1, public)
lines(0:max(altruism1), preds, col = "blue", lwd = 2)
```



Boosting

The final model is a non-parametric model that tackles the residuals using an ensemble method called boosting, it gradually builds trees on the residuals. library caret is loaded in and will not conflict with the previous used packages. First the parameters are set to train the model.

```
library(caret)

fitControl <- trainControl(method = "cv", number = 5)
gbmGrid <- expand.grid(interaction.depth = c(1, 2, 4, 8), n.trees = (1:10)*200, n.minobsinnode = 10, sh
```

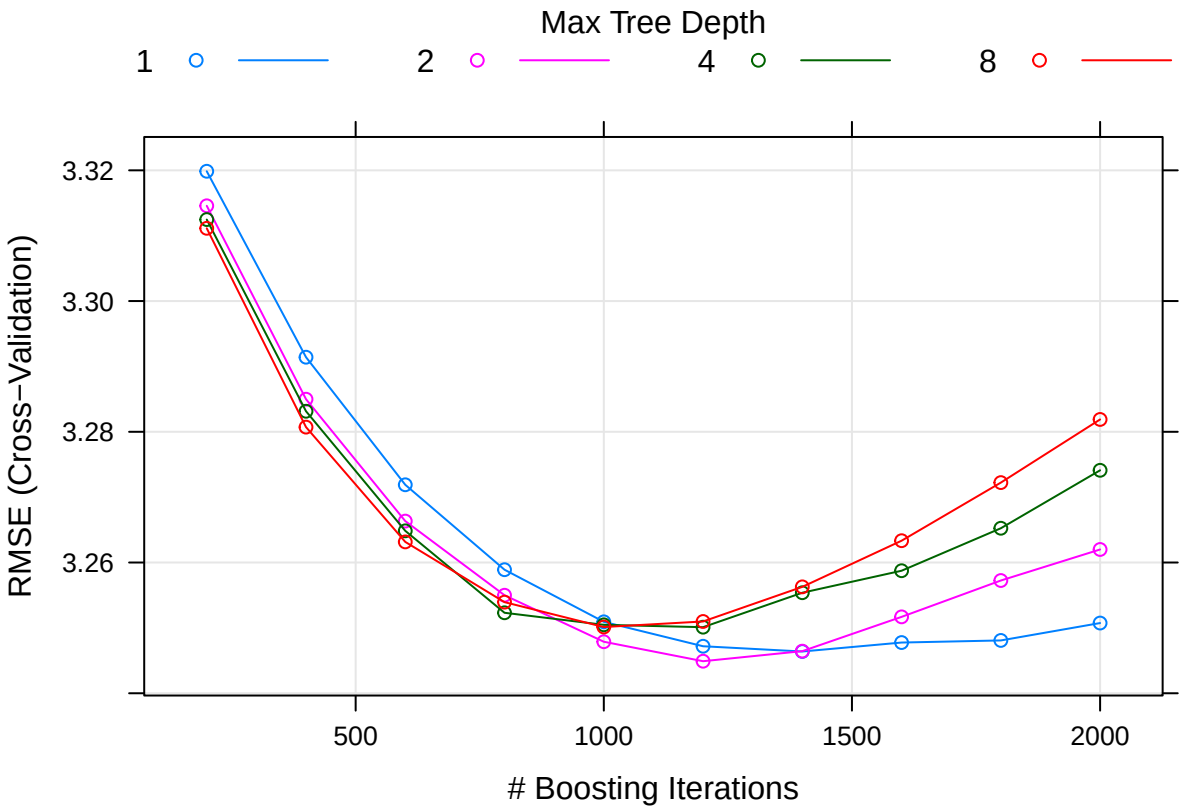
The model is trained. A MSE of 10.53, and a RMSE of 3.24 is found, which is higher than found in other simpler models. The optimal settings are 1200 trees, an interaction.depth of 2 and shrinkage factor of 0.001.

```
set.seed(64)
gbmfit <- train(public ~ ., data = trainPublic, method = "gbm", verbose = FALSE,
  trControl = fitControl, tuneGrid = gbmGrid, distribution = "gaussian")
gbmfit
```

```
## Stochastic Gradient Boosting
##
## 170 samples
##   8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
```

```
## Summary of sample sizes: 136, 136, 135, 136, 137
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared    MAE
##   1                   200     3.319857  0.08666023  2.626628
##   1                   400     3.291407  0.08687573  2.591609
##   1                   600     3.271894  0.08666377  2.582054
##   1                   800     3.258898  0.08516587  2.588160
##   1                  1000     3.250930  0.08292338  2.595987
##   1                  1200     3.247193  0.08049680  2.604366
##   1                  1400     3.246386  0.07807123  2.613351
##   1                  1600     3.247757  0.07543726  2.622359
##   1                  1800     3.248081  0.07418841  2.629240
##   1                  2000     3.250741  0.07220168  2.636237
##   2                   200     3.314571  0.08347371  2.624328
##   2                   400     3.284984  0.08196602  2.596111
##   2                   600     3.266337  0.08013164  2.591217
##   2                   800     3.254994  0.07847344  2.598256
##   2                  1000     3.247881  0.07735756  2.609699
##   2                  1200     3.244901  0.07616313  2.620682
##   2                  1400     3.246439  0.07429076  2.630085
##   2                  1600     3.251687  0.07146662  2.641521
##   2                  1800     3.257248  0.06889395  2.651646
##   2                  2000     3.261995  0.06744263  2.660318
##   4                   200     3.312467  0.08174825  2.623733
##   4                   400     3.283141  0.07687209  2.605173
##   4                   600     3.264848  0.07575776  2.607223
##   4                   800     3.252312  0.07655113  2.615428
##   4                  1000     3.250467  0.07377210  2.630175
##   4                  1200     3.250108  0.07277180  2.640177
##   4                  1400     3.255380  0.07043427  2.651534
##   4                  1600     3.258744  0.06997884  2.659719
##   4                  1800     3.265241  0.06843363  2.670916
##   4                  2000     3.274110  0.06618212  2.681782
##   8                   200     3.311131  0.08003663  2.622965
##   8                   400     3.280712  0.07782168  2.602335
##   8                   600     3.263156  0.07598029  2.605025
##   8                   800     3.253938  0.07454846  2.614972
##   8                  1000     3.250134  0.07323228  2.628438
##   8                  1200     3.250980  0.07201016  2.640687
##   8                  1400     3.256278  0.06963739  2.652553
##   8                  1600     3.263336  0.06741482  2.663313
##   8                  1800     3.272229  0.06484331  2.674951
##   8                  2000     3.281888  0.06253799  2.686154
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.001
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 1200,
##  interaction.depth = 2, shrinkage = 0.001 and n.minobsinnode = 10.
```

```
plot(gbmfit)
```




```
gbmfit$results[which.min(gbmfit$results$RMSE),1:6]
```

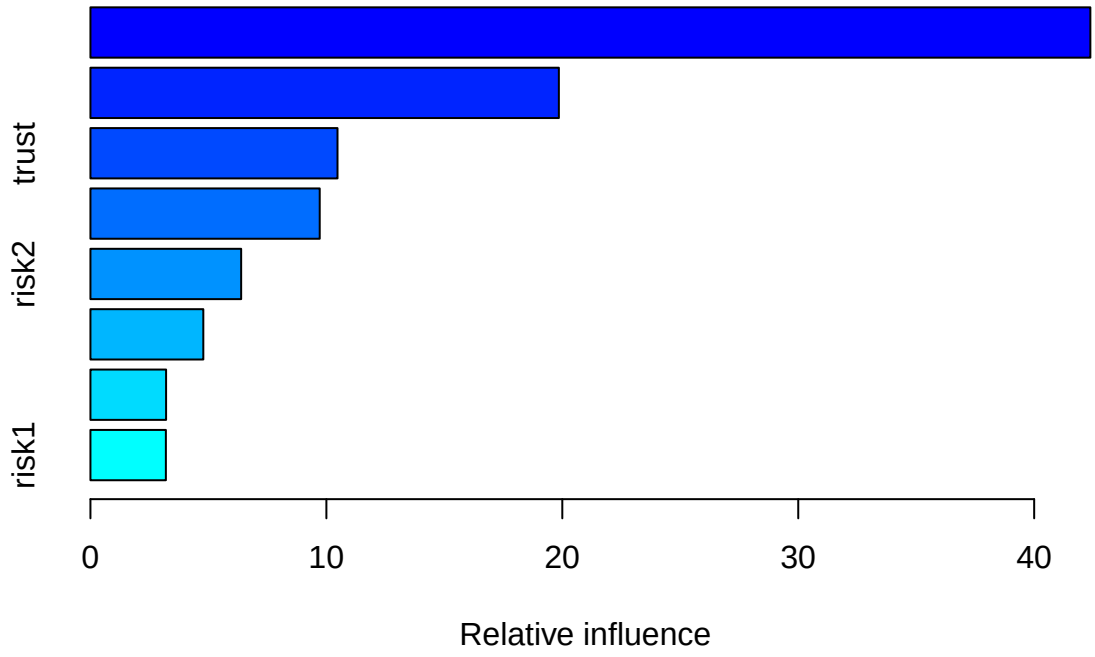
```
##    shrinkage interaction.depth n.minobsinnode n.trees     RMSE   Rsquared
## 16     0.001                 2             10    1200 3.244901 0.07616313
```

```
mseBoost <- min(gbmfit$results$RMSE)^2
mseBoost
```

```
## [1] 10.52938
```

The most important variables are the same found in all other methods: altruism1, followed by altruism2 and trust. The relative influence of altruism1, altruism2 and trust is 43%, 20% and 10% respectively. These 3 variables have a relative importance of approximately 83% and the results correspond with findings of the previous models.

```
set.seed(32)
gbmBestFit <- gbm(public ~ ., data = trainPublic, distribution = "gaussian", n.trees= 1200, interaction
summary(gbmBestFit)
```




```
##                        var   rel.inf
## altruism1       altruism1 42.381067
## altruism2       altruism2 19.853807
## trust               trust 10.471299
## negRecPooled negRecPooled  9.717780
## risk2               risk2  6.388298
## posRec2           posRec2  4.784133
## posRec1           posRec1  3.206574
## risk1               risk1  3.197041
```

Final models

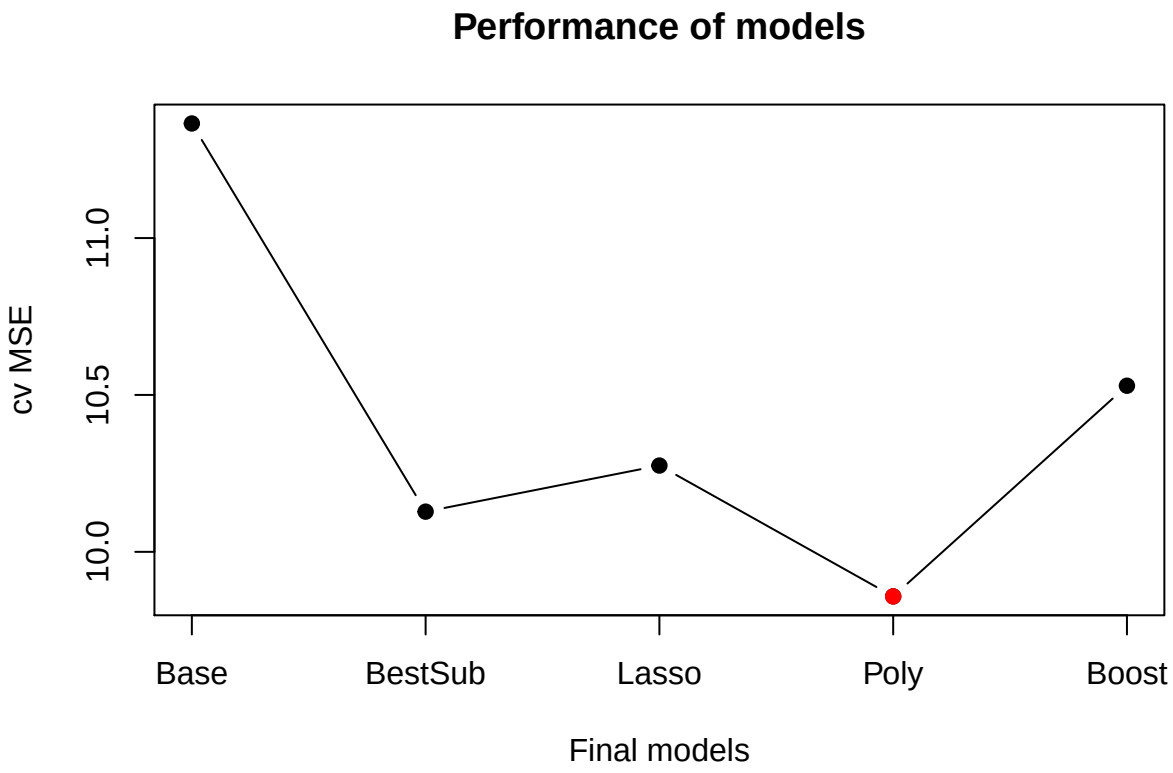
The final model are inspected. As can be seen, the best performing model is the combined model. The best subset selection improves with more than 12% compared to the base model. Adding a polynomial or a square root further decreases MSE. Lasso performs slightly worse. The combined model of square root and polynomial with trust added to the model performs best with a MSE of 9.83. This is a decrease of MSE of 13%, however the increase compared to the more simple polynomial containing altruism1 to the third degree and altruism2 is very small. Therefore the simpler polynomial model is chosen.

```
finalModels <- data.frame(MSE = c(mseBase, mseBestSub, mseLasso, msePoly, mseBoost), row.names = c("Base
finalModels
```

```
##               MSE
## Base    11.365117
## BestSub 10.128049
## Lasso   10.275017
```

```
## Poly     9.858091
## Boost   10.529381
```

```
plotModels(finalModels$MSE, xlab = "Final models", axislabels = rownames(finalModels))
```




```
## [1] 11.365117 10.128049 10.275017  9.858091 10.529381
```

Final prediction

Now the final predictions on test set are performed. The model is first trained on all training data. The base model scores a MSE of 15.15 on the test set and a RMSE of 3.89. The best model scores quite lower MSE with a 12.43 and a RMSE of 3.53. On the test set the final models performs 18% better than the base model, when measured in MSE.

```
ytrue <- qualtrics[test, "public"]
baseMse <- mean((ytrue - mean(public))^2)
baseMse
```

```
## [1] 15.14692
```

```
sqrt(baseMse)
```

```
## [1] 3.891904
```

```
finalModel <- public ~ poly(altruism1, 3) + sqrt(altruism2)
finalFit <- lm(finalModel, data = trainPublic)
yhat <- predict(finalFit, qualtrics[test,])
finalMse <- mean((ytrue - yhat)^2)
```

```
finalMse
```

```
## [1] 12.4259
```

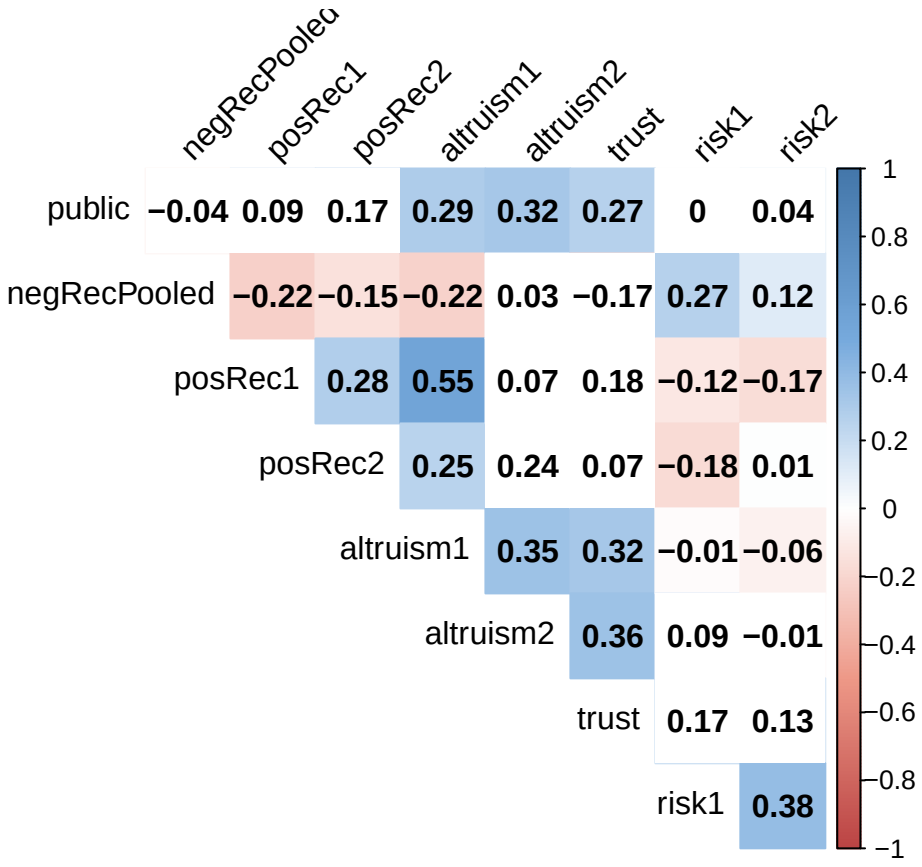
```
sqrt(finalMse)
```

```
## [1] 3.525039
```

Significance testing

Models are created to check for significance from less complex to more complex models on all data, thus including train and test set. Starting with bivariate analysis, using a non-directional Spearman's rang correlations test with a p-value threshold of 0.10. This shows a significant relation for altruism1 (0.05), altruism2 (0.01) and trust (0.02), all but trust were also included in the final model. These variables were also part of the lasso model and the first in best subset selection method, and all were found to be the most important using the boosting method.

```
qualtricsPublic <- select(qualtrics, -c(offer, respond, dilemma, chicken))
attach(qualtricsPublic)
plotCor(qualtricsPublic, pvalues = TRUE)
```




```
## $cor
##              public negRecPooled posRec1 posRec2 altruism1 altruism2 trust
## public         1.00        -0.04    0.09    0.17      0.29      0.32  0.27
## negRecPooled  -0.04         1.00   -0.22   -0.15     -0.22      0.03 -0.17
## posRec1        0.09        -0.22    1.00    0.28      0.55      0.07  0.18
## posRec2        0.17        -0.15    0.28    1.00      0.25      0.24  0.07
```

```
## altruism1      0.29        -0.22    0.55    0.25      1.00      0.35  0.32
## altruism2      0.32         0.03    0.07    0.24      0.35      1.00  0.36
## trust          0.27        -0.17    0.18    0.07      0.32      0.36  1.00
## risk1          0.00         0.27   -0.12   -0.18     -0.01      0.09  0.17
## risk2          0.04         0.12   -0.17    0.01     -0.06     -0.01  0.13
##              risk1 risk2
## public        0.00  0.04
## negRecPooled  0.27  0.12
## posRec1      -0.12 -0.17
## posRec2      -0.18  0.01
## altruism1    -0.01 -0.06
## altruism2     0.09 -0.01
## trust         0.17  0.13
## risk1         1.00  0.38
## risk2         0.38  1.00
##
## $pvalues
##              public negRecPooled posRec1 posRec2 altruism1 altruism2 trust
## public         0.00         0.17    0.21    0.15      0.05      0.01  0.02
## negRecPooled   0.17         0.00    0.00    0.01      0.00      0.31  0.15
## posRec1        0.21         0.00    0.00    0.01      0.01      0.31  0.26
## posRec2        0.15         0.01    0.01    0.00      0.05      0.38  0.64
## altruism1      0.05         0.00    0.01    0.05      0.00      0.06  0.02
## altruism2      0.01         0.31    0.31    0.38      0.06      0.00  0.01
## trust          0.02         0.15    0.26    0.64      0.02      0.01  0.00
## risk1          0.17         0.02    0.01    0.00      0.08      0.43  0.77
## risk2          0.22         0.04    0.02    0.01      0.01      0.33  0.49
##              risk1 risk2
## public        0.17  0.22
## negRecPooled  0.02  0.04
## posRec1       0.01  0.02
## posRec2       0.00  0.01
## altruism1     0.08  0.01
## altruism2     0.43  0.33
## trust         0.77  0.49
## risk1         0.00  0.01
## risk2         0.01  0.00
```

Now, the multivariate models are examined. The first model is the model found in best subset selection. The model has a R2 score 0.13, it is very significant effect for altruism1 (<0.001) and altruism2 (0.075).

```
bestSubFit <- lm(bestSubMdl, data = qualtricsPublic)
summary(bestSubFit)
```

```
##
## Call:
## lm(formula = bestSubMdl, data = qualtricsPublic)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1431 -2.4333 -0.6264  2.7360  6.7621
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.266396   0.703932   1.799   0.0735 .
```

```
## altruism1   0.398080   0.095796   4.155  4.8e-05 ***
## altruism2   0.003792   0.001404   2.701   0.0075 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.243 on 201 degrees of freedom
## Multiple R-squared:  0.132,  Adjusted R-squared:  0.1233
## F-statistic: 15.28 on 2 and 201 DF,  p-value: 6.645e-07
```

Adding the square root to altruism2 barely the reduces RMSE, and in this model the polynomial of the third degrees loses it's weak significance.

```
mdl2 <- public ~ poly(altruism1, 3) + sqrt(altruism2)
fit2 <- lm(mdl3, data = qualtricsPublic)
summary(fit2)
```

```
##
## Call:
## lm(formula = mdl3, data = qualtricsPublic)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4642 -2.4756 -0.4325  2.5953  7.2979
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.108281   0.275694  14.902  < 2e-16 ***
## poly(altruism1, 3)1 13.828944   3.274576   4.223 3.66e-05 ***
## poly(altruism1, 3)2 -6.553285   3.203889  -2.045  0.04213 *
## poly(altruism1, 3)3 -5.402534   3.203166  -1.687  0.09324 .
## altruism2             0.003685   0.001387   2.656  0.00854 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.203 on 199 degrees of freedom
## Multiple R-squared:  0.1616, Adjusted R-squared:  0.1447
## F-statistic: 9.588 on 4 and 199 DF,  p-value: 4.14e-07
```

Adding the polynomial to altruism1 is significant for first degree (<0.001), the second degree (0.04) and weakly for the third degree (0.09) the altruism2 is also very significant (0.009). This model explains 16.2% of all the variance in the model. The coefficients of the final model are 13.83, -6.55 and -5.44 for the altruism1 and polynomial terms. Altruism2 has a coefficient of 0.004.

```
finalFit <- public ~ poly(altruism1, 3) + altruism2
finalFit <- lm(finalFit, data = qualtricsPublic)
summary(finalFit)
```

```
##
## Call:
## lm(formula = finalFit, data = qualtricsPublic)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4642 -2.4756 -0.4325  2.5953  7.2979
##
## Coefficients:
```

```
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.108281   0.275694  14.902  < 2e-16 ***
## poly(altruism1, 3)1 13.828944   3.274576   4.223 3.66e-05 ***
## poly(altruism1, 3)2 -6.553285   3.203889  -2.045  0.04213 *
## poly(altruism1, 3)3 -5.402534   3.203166  -1.687  0.09324 .
## altruism2            0.003685   0.001387   2.656  0.00854 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.203 on 199 degrees of freedom
## Multiple R-squared:  0.1616, Adjusted R-squared:  0.1447
## F-statistic: 9.588 on 4 and 199 DF,  p-value: 4.14e-07
```

Finally a anova test is conducted, this test confirms these findings and show that adding polynomial terms is significant, but that adding square root term barely improves the model.

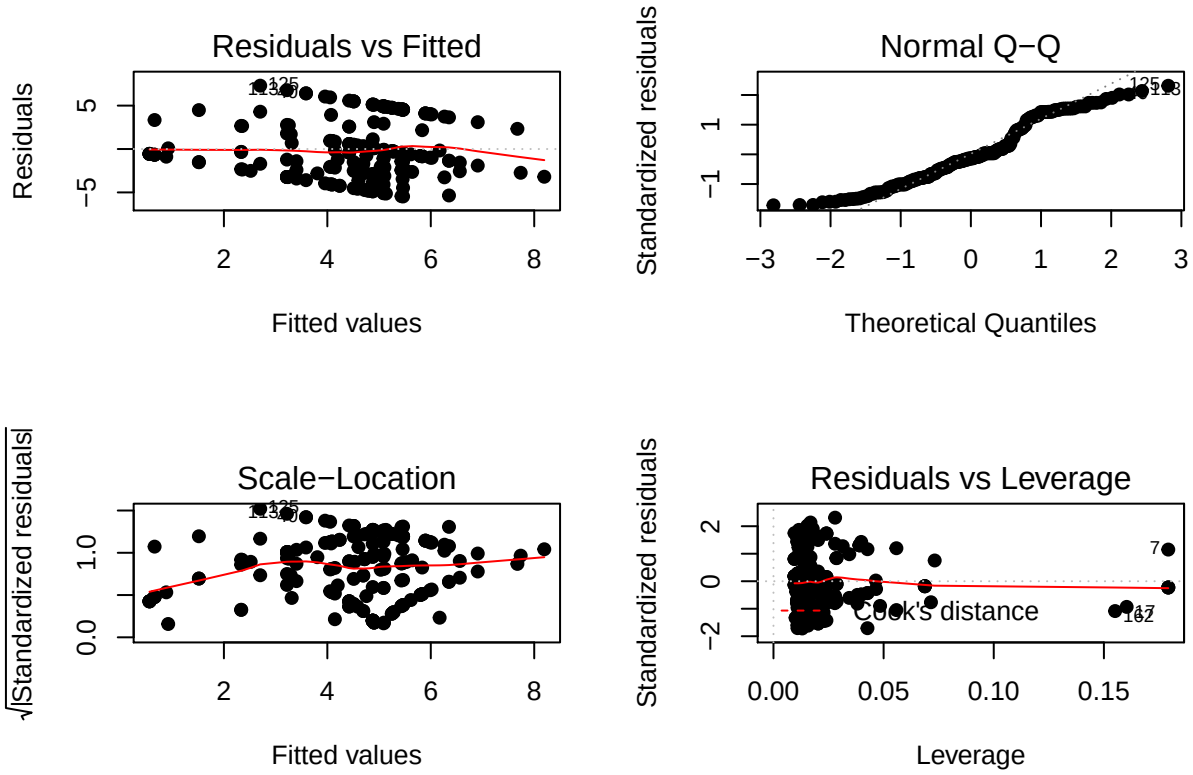
```
anova(bestSubFit, fit2, finalFit)
```

```
## Analysis of Variance Table
##
## Model 1: public ~ altruism1 + altruism2
## Model 2: public ~ poly(altruism1, 3) + altruism2
## Model 3: public ~ poly(altruism1, 3) + altruism2
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    201 2113.4
## 2    199 2041.4  2    72.072 3.5129 0.03167 *
## 3    199 2041.4  0     0.000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Diagnostics

In this section the diagnostics of the models are conducted. Beginning with a general overview plot of the model. It shows normality, no strong outliers or high influence points. Next the assumptions are investigated in more detail.

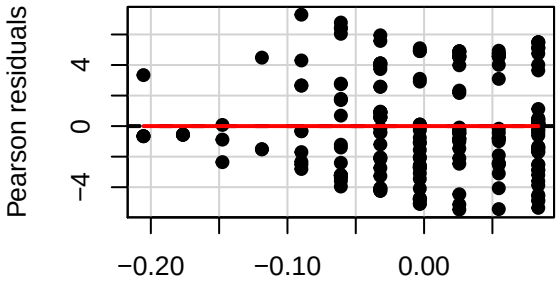
```
par(mfrow=c(2,2))
plot(finalFit, pch = 19)
```

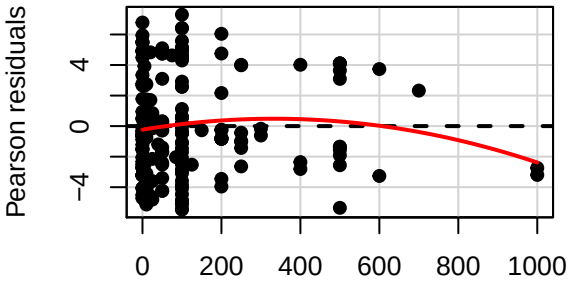


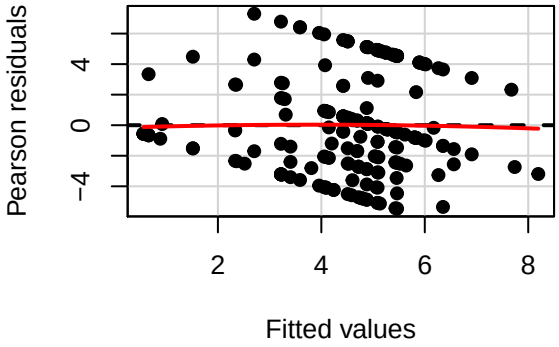
Non-linearity

As seen in the tested polynomial models above, there is no strong evidence that non-linearity would improve this model. This is also confirmed by the p-values underneath which show the effect of adding a second degree polynomial term of the variables. Most non-linearity is already accounted for due to the polynomial third degree term of altruism1.

```
par(mfrow=c(1,1))
residualPlots((lm(finalFit, data = qualtricsPublic)), pch=19)
```








```
##                     Test stat Pr(>|t|)
## poly(altruism1, 3)         NA       NA
## altruism2              -1.497    0.136
## Tukey test             -0.405    0.686
```

Correlation of error terms

The second test is an inspection of correlation of error terms, however there is no evidence for correlation of error-terms as found in the Durbin Watson test.

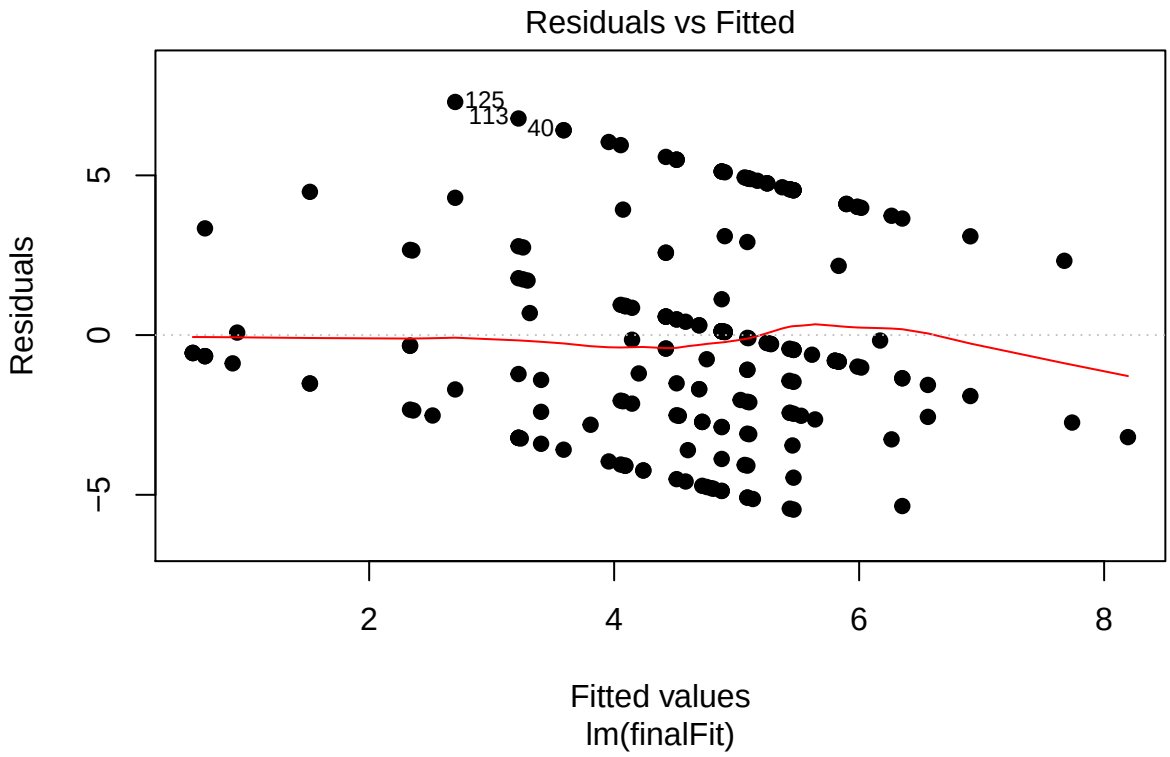
```
durbinWatsonTest(finalFit)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.02971765      2.049333   0.682
##  Alternative hypothesis: rho != 0
```

Heteroskedasticity

The residuals vs. predicted plot does show small evidence of heteroskedasticity, in the beginning the funnel is smaller and than it breaks and the width stays the same but in general the residuals form a attern to the lower numbers, it is important to note that the assumptions of confidence interval and hypothesis tests do rely on this assumption.

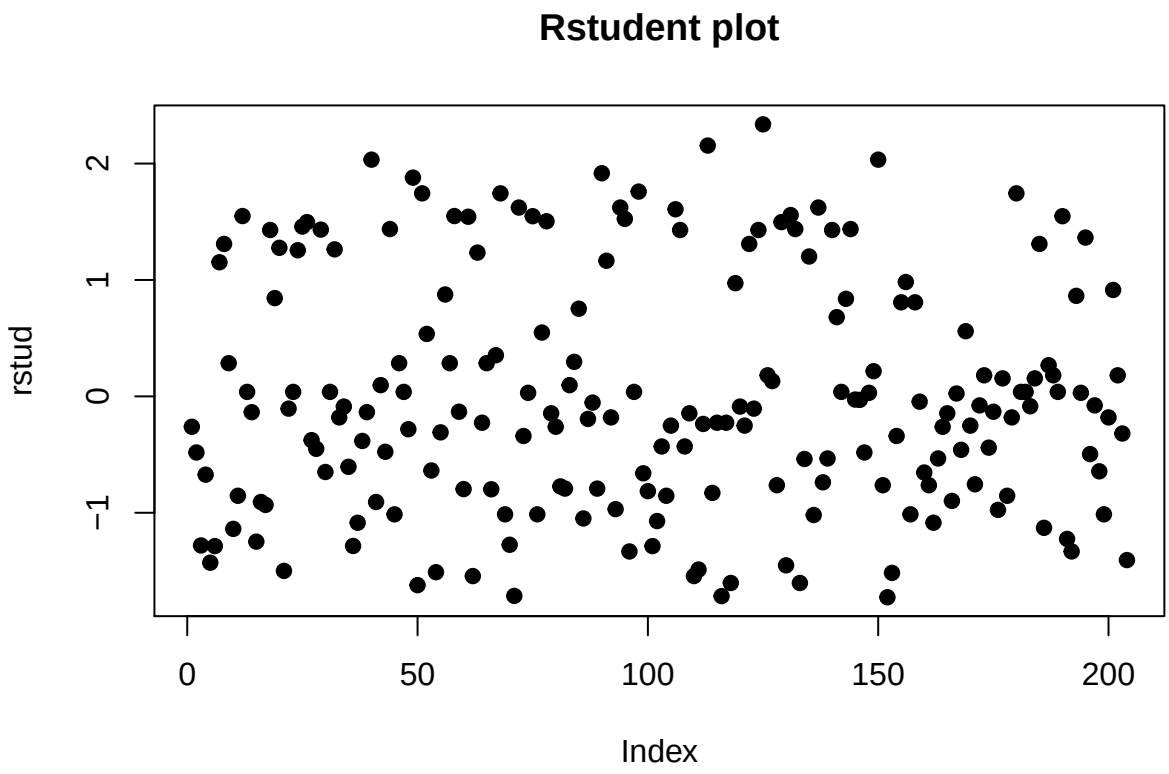
```
plot(finalFit, which = 1, pch = 19)
```



Outliers

There seems to be no evidence for outliers, this found in the studentized residuals plot. None of the residuals have a z-score > 2.

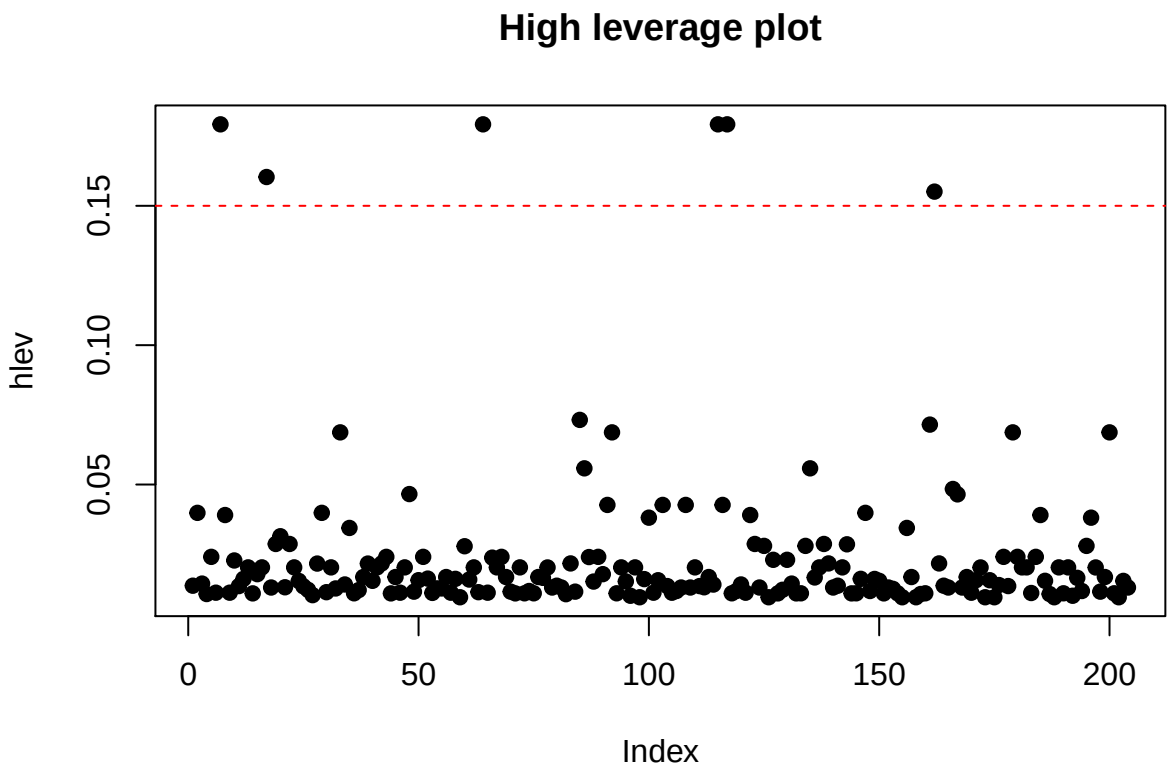
```
rstud <- rstudent(finalFit)
plot(rstud, pch = 19, main = "Rstudent plot")
```



High leverage points

There are four points that do have quite high leverage (>0.10). Removing these high leverage points and refitting the data, shows that the third degree polynomial of altruism1 disappears and the second degree term becomes less significant, furthermore the R2 and RMSE stays about the samen. Only the coefficient of the first poly seem to change a little, all the rest stay more or less the same.

```
hlev <- hatvalues(finalFit)
plot(hlev, pch = 19, main = "High leverage plot")
abline(h = 0.15, col = "red", lty = 2)
```




```
summary(lm(finalFit, data = qualtricsPublic))
```

```
##
## Call:
## lm(formula = finalFit, data = qualtricsPublic)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4642 -2.4756 -0.4325  2.5953  7.2979
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.108281   0.275694  14.902  < 2e-16 ***
## poly(altruism1, 3)1 13.828944   3.274576   4.223 3.66e-05 ***
## poly(altruism1, 3)2 -6.553285   3.203889  -2.045  0.04213 *
## poly(altruism1, 3)3 -5.402534   3.203166  -1.687  0.09324 .
## altruism2            0.003685   0.001387   2.656  0.00854 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.203 on 199 degrees of freedom
## Multiple R-squared:  0.1616, Adjusted R-squared:  0.1447
## F-statistic: 9.588 on 4 and 199 DF,  p-value: 4.14e-07
```

```
summary(lm(finalFit, data = qualtricsPublic[!(hlev > 0.15),]))
```

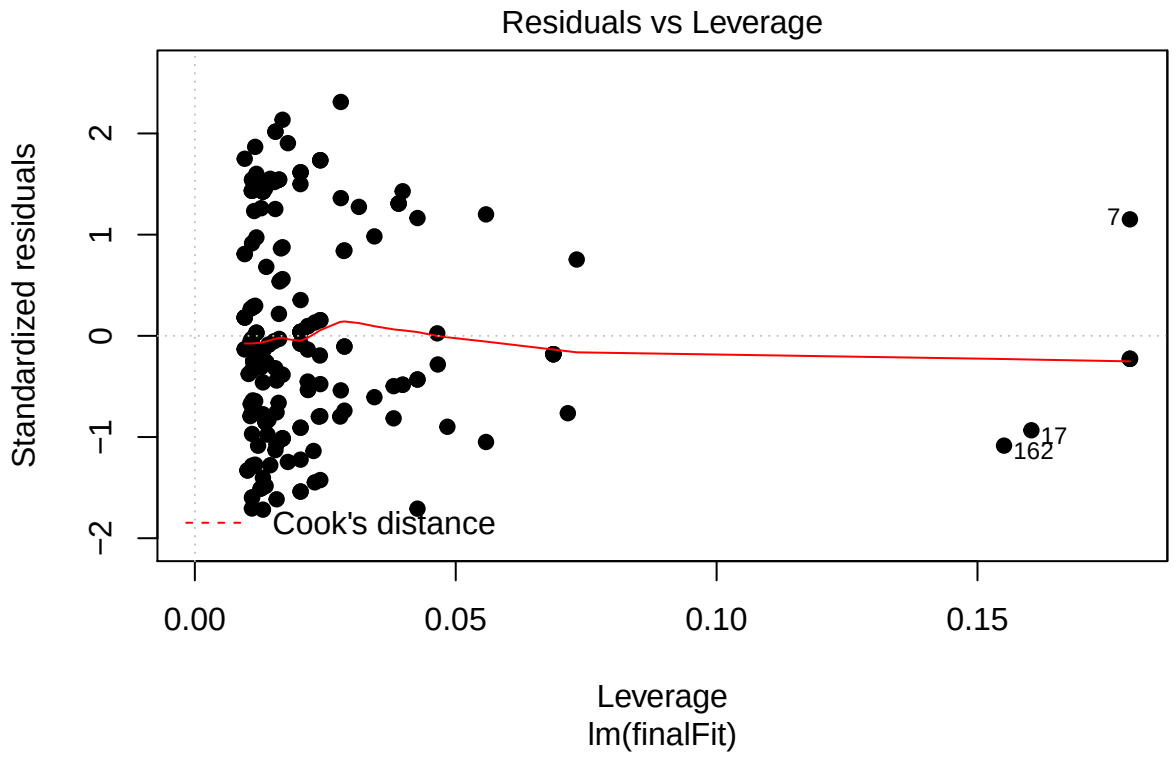
```
##
```

```
## Call:
## lm(formula = finalFit, data = qualtricsPublic[!(hlev > 0.15),
##     ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9493 -2.4240 -0.3537  2.5760  7.1535
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.960626   0.294557  13.446  < 2e-16 ***
## poly(altruism1, 3)1 14.523819   4.136042   3.512 0.000555 ***
## poly(altruism1, 3)2 -7.960512   4.703869  -1.692 0.092196 .
## poly(altruism1, 3)3 -3.836014   4.264419  -0.900 0.369486
## altruism2             0.005089   0.001656   3.074 0.002421 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.221 on 193 degrees of freedom
## Multiple R-squared:  0.1558, Adjusted R-squared:  0.1383
## F-statistic: 8.905 on 4 and 193 DF,  p-value: 1.279e-06
```

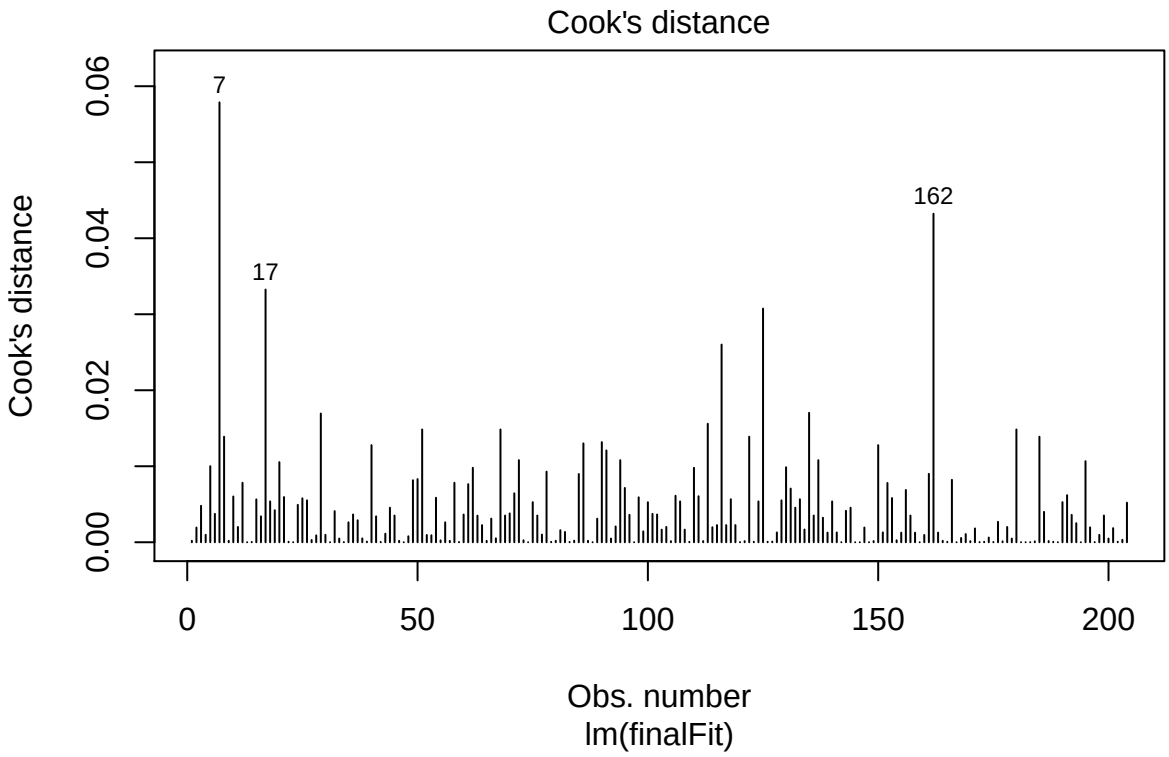
Influential points

All influential points are well under 1. The Residuals vs. Leverage Plot and the Cook's Distance plot show that there are high-leverage points, but there is no evidence for influental points due the lack of outliers.

```
plot(finalFit, which = 5, pch = 19)
```




```
plot(finalFit, which = 4, pch = 19)
```



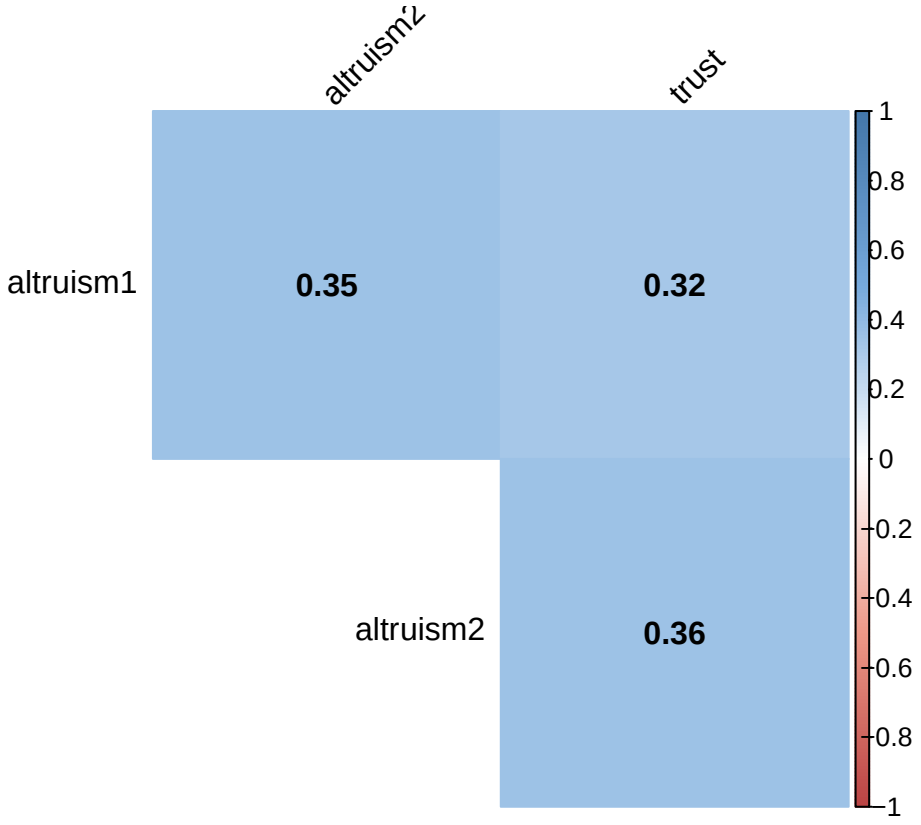
Collinearity

The last diagnostic tests collinearity, as observed earlier there is high collinearity between variables and this possibly influences the coefficients of the variables. However there is no evidence for multicollinearity as the VIF scores shows.

```
vif(finalFit)
```

```
##                         GVIF Df GVIF^(1/(2*Df))
## poly(altruism1, 3) 1.046184  3        1.007553
## altruism2          1.046184  1        1.022831
```

```
plotCor(data.frame(altruism1, altruism2, trust))
```




```
## $cor
##           altruism1 altruism2 trust
## altruism1      1.00      0.35  0.32
## altruism2      0.35      1.00  0.36
## trust          0.32      0.36  1.00
```

Offer

First a subset of trainQualtrics is taken with only the made offer in the ultimatum game as dependent variable and the preferences, all the other games are excluded.

```
trainOffer <- select(trainQualtrics, -c(public, respond, dilemma, chicken))
attach(trainOffer)
```

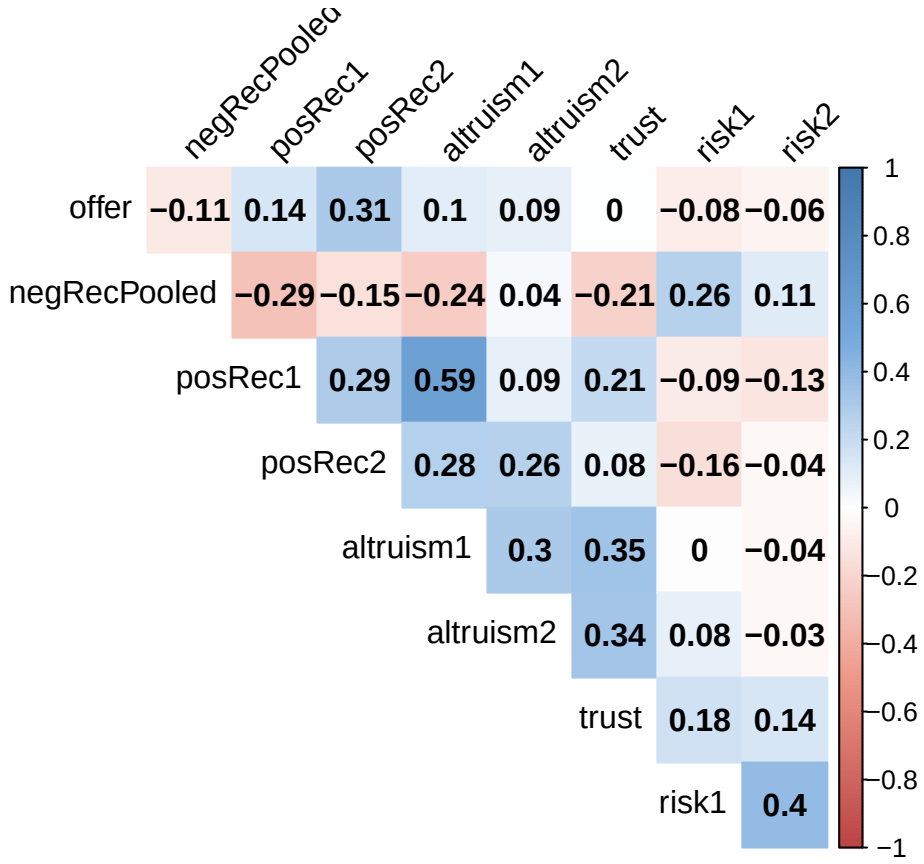
Bivariate analysis

For the same reason as in public goods, Spearman's rang correlation is chosen. Furthermore, it must be noted that these are just bivariate relationships and do not account for possible confounding relations.

The following (absolute) correlations of greater than 0.10 are found:

- Negative relationship with negRecPooled (-0.11)
- Positive relationship with posRec1 (0.13).
- Positive relationship with posRec2 (0.31).
- Positive relationship with altruism1 (0.1).

```
plotCor(trainOffer)
```




```
## $cor
##              offer negRecPooled posRec1 posRec2 altruism1 altruism2 trust
## offer         1.00        -0.11    0.14    0.31      0.10      0.09  0.00
## negRecPooled -0.11         1.00   -0.29   -0.15     -0.24      0.04 -0.21
## posRec1       0.14        -0.29    1.00    0.29      0.59      0.09  0.21
## posRec2       0.31        -0.15    0.29    1.00      0.28      0.26  0.08
## altruism1     0.10        -0.24    0.59    0.28      1.00      0.30  0.35
## altruism2     0.09         0.04    0.09    0.26      0.30      1.00  0.34
## trust         0.00        -0.21    0.21    0.08      0.35      0.34  1.00
## risk1        -0.08         0.26   -0.09   -0.16      0.00      0.08  0.18
## risk2        -0.06         0.11   -0.13   -0.04     -0.04     -0.03  0.14
##              risk1 risk2
## offer        -0.08 -0.06
## negRecPooled  0.26  0.11
## posRec1      -0.09 -0.13
## posRec2      -0.16 -0.04
## altruism1     0.00 -0.04
## altruism2     0.08 -0.03
## trust         0.18  0.14
## risk1         1.00  0.40
## risk2         0.40  1.00
```

Base model

A base model is conducted using the basemodelcv function. A cross-validated MSE score of 3.30 and a standard deviation of 1.70 and a RMSE of 1.82 are found.

```
baseline <- baseModelcv(trainOffer, foldid)
baseline
```

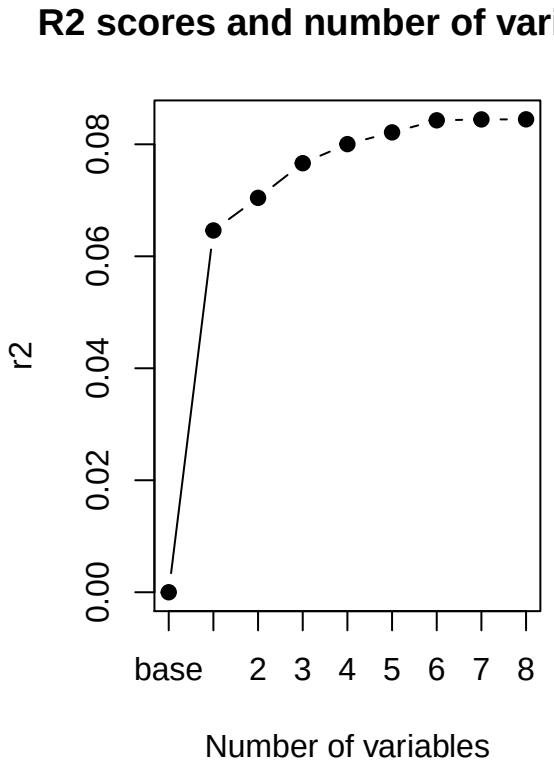
```
## $cv
## [1] 4.165441 1.918685 2.527682 5.878947 1.984483
##
## $meancv
## [1] 3.295048
##
## $sdcv
## [1] 1.7055
##
## $rmse
## [1] 1.815227
```

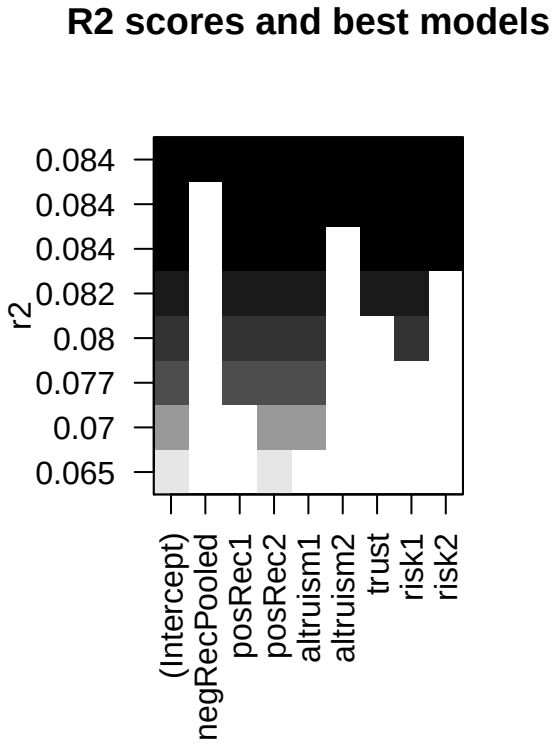
```
mseBase <- baseline$meancv
```

Best subset selection

A best subset selection is performed on all training data. As the plot shows posRec2 explains a little more than 6% of all the variance in the model. By adding any other variables, little increase is gained.

```
bestOffer <- regsubsets(offer ~ ., data = trainOffer)
plotR2Subs(bestOffer)
```






```
## Subset selection object
## Call: regsubsets.formula(offer ~ ., data = trainOffer)
## 8 Variables  (and intercept)
##              Forced in Forced out
## negRecPooled     FALSE      FALSE
## posRec1          FALSE      FALSE
## posRec2          FALSE      FALSE
## altruism1        FALSE      FALSE
## altruism2        FALSE      FALSE
## trust            FALSE      FALSE
## risk1            FALSE      FALSE
## risk2            FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          negRecPooled posRec1 posRec2 altruism1 altruism2 trust risk1
## 1  ( 1 ) " "          " "     "*"     " "       " "       " "   " "
## 2  ( 1 ) " "          " "     "*"     "*"       " "       " "   " "
## 3  ( 1 ) " "          "*"     "*"     "*"       " "       " "   " "
## 4  ( 1 ) " "          "*"     "*"     "*"       " "       " "   "*"
## 5  ( 1 ) " "          "*"     "*"     "*"       " "       "*"   "*"
## 6  ( 1 ) " "          "*"     "*"     "*"       " "       "*"   "*"
## 7  ( 1 ) " "          "*"     "*"     "*"       "*"       "*"   "*"
## 8  ( 1 ) "*"          "*"     "*"     "*"       "*"       "*"   "*"
##          risk2
## 1  ( 1 ) " "
## 2  ( 1 ) " "
```

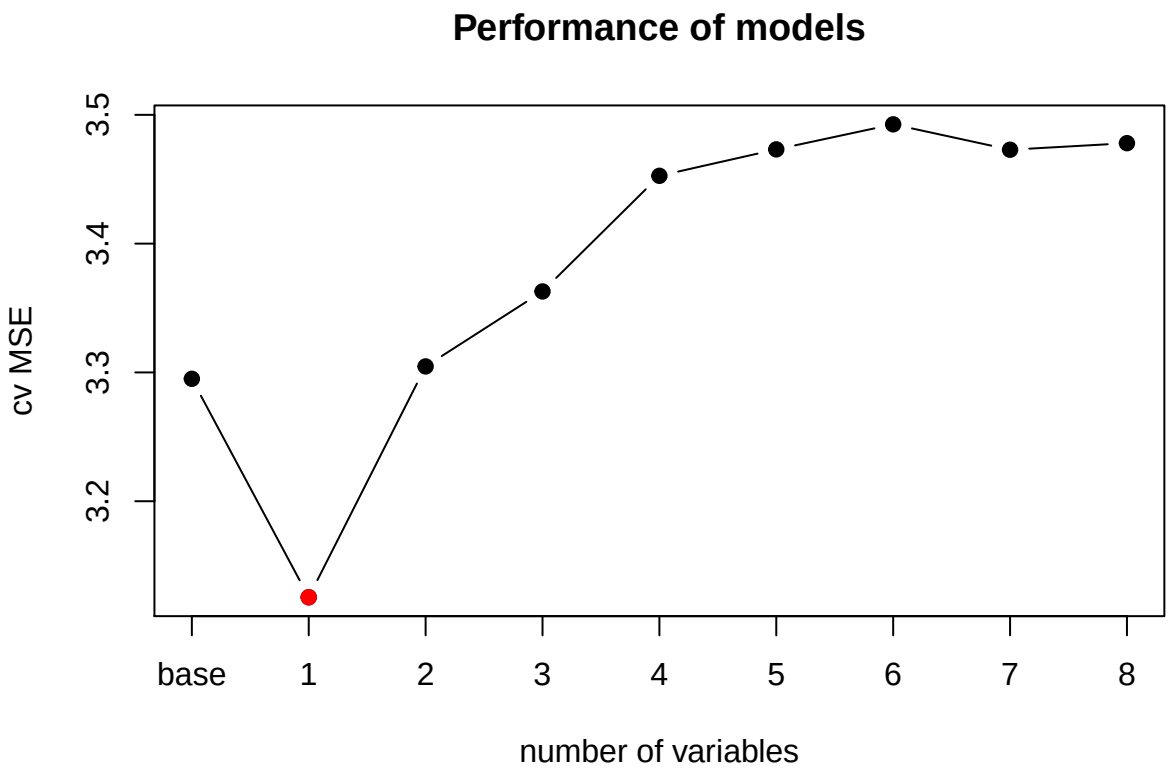
```
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) "*"
## 7  ( 1 ) "*"
## 8  ( 1 ) "*"
```

A best subset selection is conducted using the bestsubsetcv function. The best performing model has one variable, as expected reviewing the R2 score above. One variable model corresponds with a MSE of 3.13, a standard deviation of 1.8 and a RMSE of 1.77.

```
bestSubs <- bestSubsetcv(trainOffer, foldid)
bestSubs
```

```
## $cv
##             1        2        3        4        5        6        7
## [1,] 3.651744 3.692996 3.631239 3.680973 3.622586 3.680307 3.658505
## [2,] 1.900132 2.320495 2.386034 2.450904 2.480216 2.530563 2.519016
## [3,] 2.293332 2.399781 2.361474 2.321536 2.467750 2.571398 2.553420
## [4,] 6.050306 6.354057 6.692039 6.891730 6.922604 6.794507 6.767059
## [5,] 1.731740 1.756008 1.743606 1.918160 1.872696 1.886296 1.866530
##             8
## [1,] 3.659165
## [2,] 2.517341
## [3,] 2.541789
## [4,] 6.790006
## [5,] 1.881688
##
## $meancv
##        1        2        3        4        5        6        7        8
## 3.125451 3.304667 3.362878 3.452660 3.473170 3.492614 3.472906 3.477998
##
## $sdcv
##        1        2        3        4        5        6        7        8
## 1.800557 1.846303 1.983331 2.032054 2.029733 1.955260 1.950749 1.958939
##
## $bestnmdl
## [1] 1
##
## $bestnmean
## [1] 3.125451
##
## $bestnsd
## [1] 1.800557
##
## $bestnrmse
## [1] 1.767894
```

```
mseBestSub <- bestSubs$bestnmean
plotModels(bestSubs$meancv, baseline)
```




```
##     base        1        2        3        4        5        6        7
## 3.295048 3.125451 3.304667 3.362878 3.452660 3.473170 3.492614 3.472906
##        8
## 3.477998
```

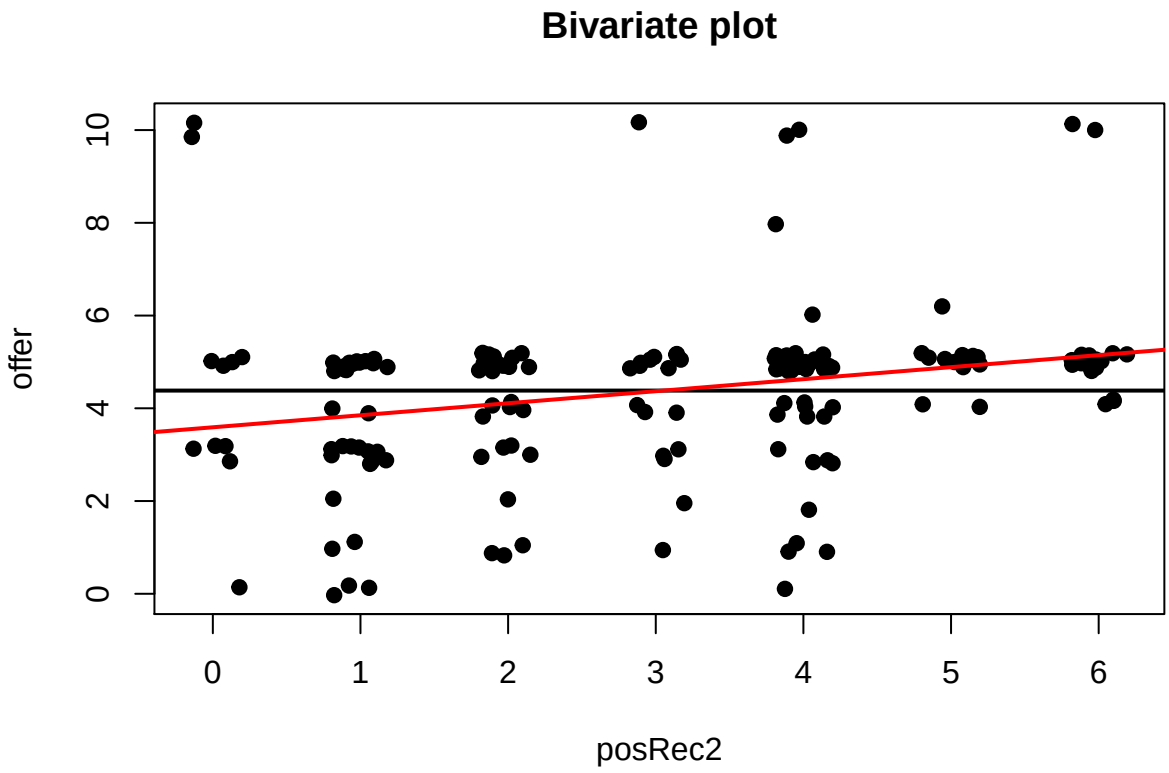
The variable for the best model is posRec2, this confirms with the findings of the R2 plot and the bivariate analysis. The corresponding coefficient is 0.26. This means that people that give 5 dollars extra to a helpful stranger, what corresponds with an increase of 1 in the non-recoded variable range (see posRec2), the amount offered increases on average with 0.259. Furthermore the model and the posRec2 variable is strongly significant (<0.001)

```
bestSubFit <- lm(offer ~ posRec2, data = trainOffer)
summary(bestSubFit)
```

```
##
## Call:
## lm(formula = offer ~ posRec2, data = trainOffer)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6280 -0.8499  0.1127  0.6314  6.4095
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.59054    0.26829  13.383  < 2e-16 ***
## posRec2      0.25936    0.07613   3.407 0.000823 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.747 on 168 degrees of freedom
## Multiple R-squared:  0.06461,    Adjusted R-squared:  0.05905
## F-statistic:  11.6 on 1 and 168 DF,  p-value: 0.0008231
```

```
plotBivariate(posRec2, offer)
```



Lasso

The best performing model has a lambda of 0.20 and a corresponding MSE of 3.17 and a RMSE of 1.78, which is slightly worse than the best subset selection method. However, the standard deviation of 0.79 is a lot lower and results are thus more stable. As the plot shows, the optimal lambda includes one variable, however the lambda plus 1 standard error corresponds with the base model, this shows that the difference isn't very substantial.

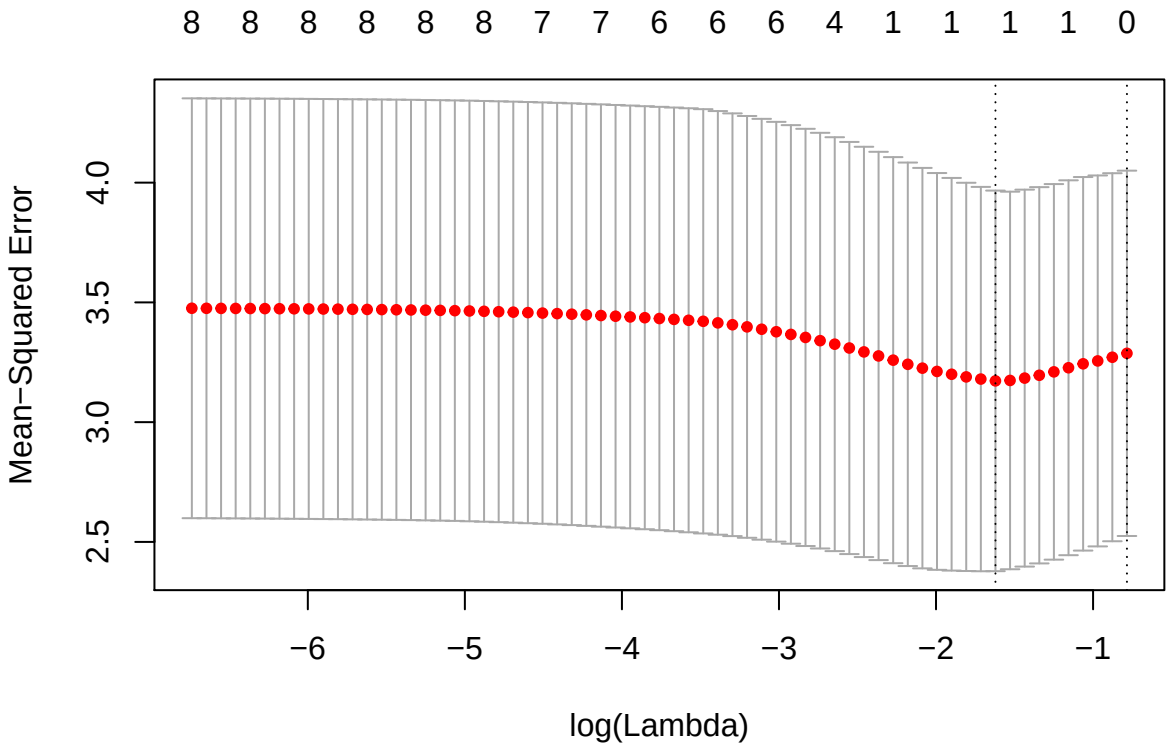
```
lassocv <- cv.glmnet(trainPreferences, offer, alpha = 1, foldid = foldid)
resultsLasso(lassocv)
```

```
## $glmnet.fit
##
## Call:  glmnet(x = trainPreferences, y = offer, alpha = 1)
##
##       Df    %Dev    Lambda
##  [1,]  0 0.00000 0.4564000
##  [2,]  1 0.01097 0.4159000
##  [3,]  1 0.02008 0.3789000
```

```
##  [4,]  1 0.02764 0.3453000
##  [5,]  1 0.03392 0.3146000
##  [6,]  1 0.03913 0.2867000
##  [7,]  1 0.04346 0.2612000
##  [8,]  1 0.04705 0.2380000
##  [9,]  1 0.05003 0.2168000
## [10,]  1 0.05251 0.1976000
## [11,]  1 0.05456 0.1800000
## [12,]  1 0.05627 0.1640000
## [13,]  1 0.05769 0.1495000
## [14,]  1 0.05886 0.1362000
## [15,]  1 0.05984 0.1241000
## [16,]  1 0.06065 0.1131000
## [17,]  1 0.06132 0.1030000
## [18,]  2 0.06257 0.0938700
## [19,]  2 0.06391 0.0855300
## [20,]  3 0.06565 0.0779300
## [21,]  4 0.06797 0.0710100
## [22,]  4 0.07002 0.0647000
## [23,]  4 0.07172 0.0589500
## [24,]  6 0.07354 0.0537100
## [25,]  6 0.07536 0.0489400
## [26,]  6 0.07688 0.0445900
## [27,]  6 0.07814 0.0406300
## [28,]  6 0.07918 0.0370200
## [29,]  6 0.08004 0.0337300
## [30,]  6 0.08076 0.0307400
## [31,]  6 0.08136 0.0280100
## [32,]  6 0.08185 0.0255200
## [33,]  6 0.08227 0.0232500
## [34,]  7 0.08264 0.0211900
## [35,]  7 0.08294 0.0193000
## [36,]  7 0.08319 0.0175900
## [37,]  7 0.08341 0.0160300
## [38,]  7 0.08358 0.0146000
## [39,]  7 0.08373 0.0133100
## [40,]  7 0.08385 0.0121200
## [41,]  7 0.08395 0.0110500
## [42,]  7 0.08403 0.0100700
## [43,]  7 0.08410 0.0091710
## [44,]  7 0.08416 0.0083560
## [45,]  8 0.08421 0.0076140
## [46,]  8 0.08425 0.0069380
## [47,]  8 0.08429 0.0063210
## [48,]  8 0.08432 0.0057600
## [49,]  8 0.08434 0.0052480
## [50,]  8 0.08436 0.0047820
## [51,]  8 0.08438 0.0043570
## [52,]  8 0.08439 0.0039700
## [53,]  8 0.08440 0.0036170
## [54,]  8 0.08441 0.0032960
## [55,]  8 0.08442 0.0030030
## [56,]  8 0.08442 0.0027360
## [57,]  8 0.08443 0.0024930
```

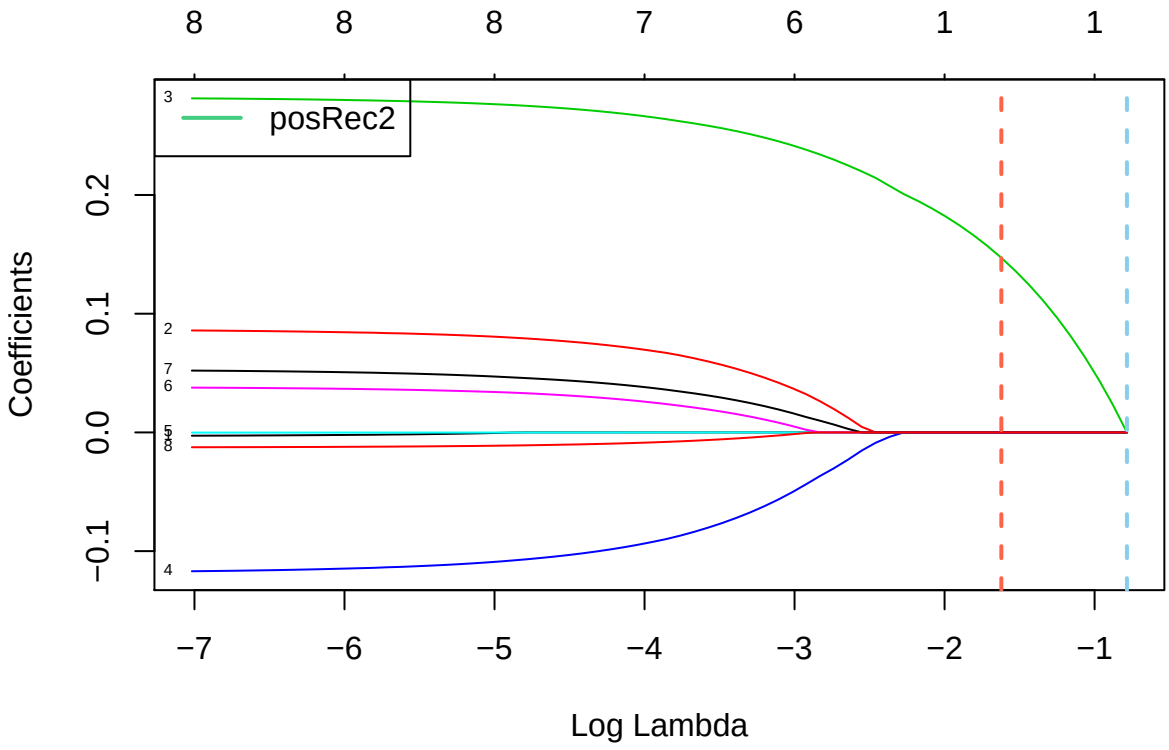
```
## [58,]  8 0.08443 0.0022720
## [59,]  8 0.08444 0.0020700
## [60,]  8 0.08444 0.0018860
## [61,]  8 0.08444 0.0017180
## [62,]  8 0.08445 0.0015660
## [63,]  8 0.08445 0.0014270
## [64,]  8 0.08445 0.0013000
## [65,]  8 0.08445 0.0011840
## [66,]  8 0.08445 0.0010790
## [67,]  8 0.08445 0.0009834
## [68,]  8 0.08445 0.0008960
##
## $bestlambda
## [1] 0.1975832
##
## $bestnmean
## [1] 3.172326
##
## $bestnsd
## [1] 0.7945144
##
## $bestnrmse
## [1] 1.781103
##
## $lambda1se
## [1] 0.4564429
```

```
mseLasso <- min(lassocv$cvm)
plot(lassocv)
```



The shrinkage plot shows the shrank coefficients. The plot shows the best lambda includes one variable, posRec2. This was also found in the best subset selection method. The coefficient corresponding with this penalty factor is 0.147.

```
plotShrinkage(trainPreferences, offer, lassocv)
legend('topleft', legend = c("posRec2"), lwd = 2, col= c("seagreen3"))
```



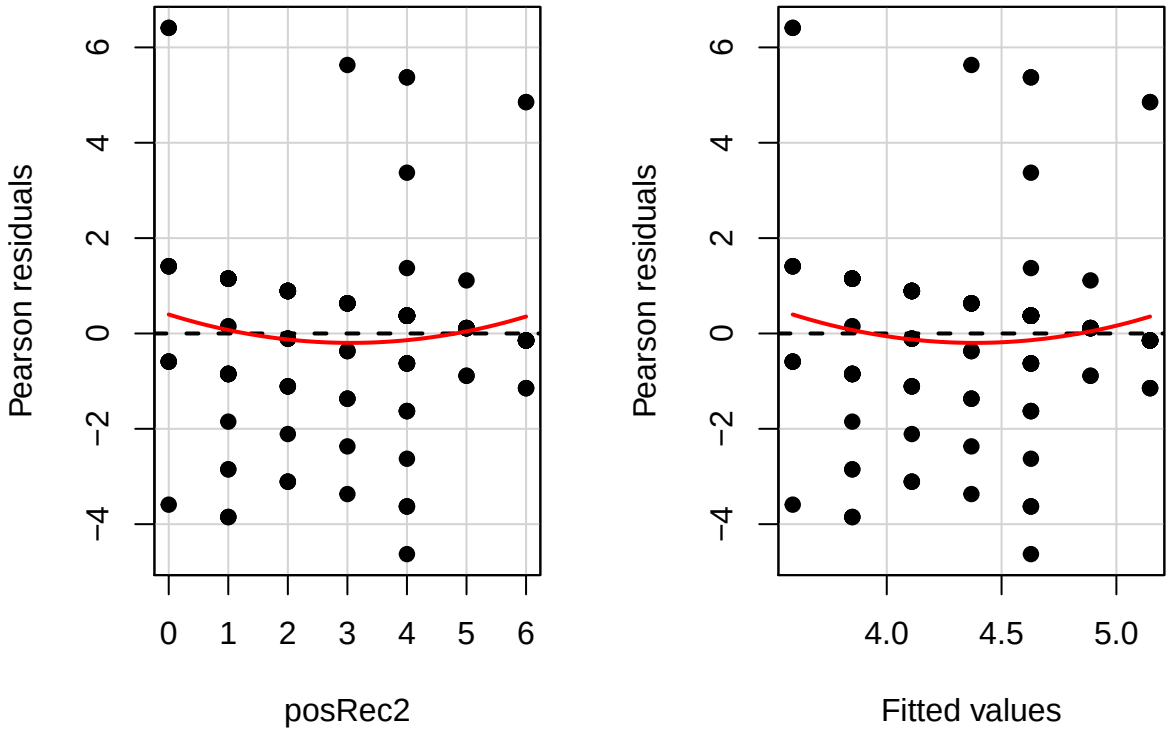

```
predict(lassocv, type = "coefficients", s = lassocv$lambda.min)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  3.9332992
## negRecPooled .
## posRec1      .
## posRec2      0.1470889
## altruism1    .
## altruism2    .
## trust        .
## risk1        .
## risk2        .
```

Non-linearity

Non-linear relations for the best subset model are examined. The residual plot shows linearity, adding a polynomial of posRec2 would not be significant, shown by a p-value of 0.147.

```
residualPlots((bestSubFit), pch=19)
```




```
##            Test stat Pr(>|t|)
## posRec2        1.459    0.147
## Tukey test     1.459    0.145
```

Boosting

The final model is a gradient boosting machine. First the parameters are set to train the model.

```
fitControl <- trainControl(method = "cv", number = 5)
gbmGrid <-  expand.grid(interaction.depth = c(1, 2, 4, 8), n.trees = (0.1:10)*100, n.minobsinnode = 20,
```

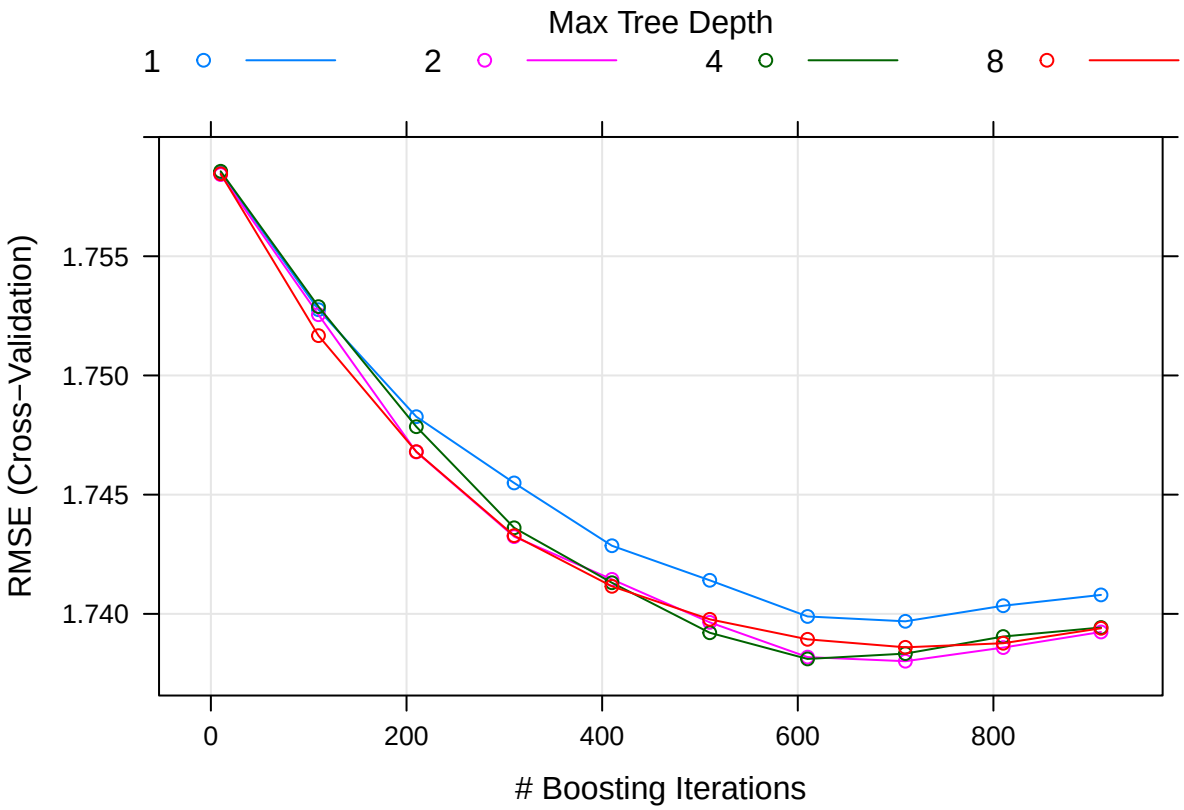
The model is trained. A MSE of 3.02 and a RMSE of 1.74 is found, which is lower than found in other simpler models. The optimal settings have 710 trees, an interaction.depth of 2, n.minosinnode of 20 and a shrinkage factor of 0.001.

```
set.seed(64)
gbmFit <- train(offer ~ ., data = trainOffer, method = "gbm", verbose = FALSE, trControl = fitControl,
gbmFit
```

```
## Stochastic Gradient Boosting
##
## 170 samples
##   8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 136, 135, 135, 137, 137
```

```
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared    MAE
##   1                   10      1.758504  0.05355262  1.232158
##   1                  110      1.752758  0.05424232  1.226737
##   1                  210      1.748270  0.05348925  1.221780
##   1                  310      1.745491  0.05169110  1.217838
##   1                  410      1.742857  0.05114180  1.213592
##   1                  510      1.741407  0.04999168  1.210854
##   1                  610      1.739891  0.04983830  1.208150
##   1                  710      1.739687  0.04787451  1.206170
##   1                  810      1.740341  0.04627499  1.205545
##   1                  910      1.740795  0.04577978  1.204197
##   2                   10      1.758426  0.05739167  1.232069
##   2                  110      1.752547  0.05238334  1.225812
##   2                  210      1.746792  0.05438553  1.219292
##   2                  310      1.743240  0.05119447  1.213935
##   2                  410      1.741447  0.04845976  1.210046
##   2                  510      1.739644  0.04923543  1.206130
##   2                  610      1.738187  0.04866560  1.202504
##   2                  710      1.738014  0.04781028  1.200530
##   2                  810      1.738586  0.04672419  1.198890
##   2                  910      1.739240  0.04572510  1.197157
##   4                   10      1.758565  0.04785077  1.232120
##   4                  110      1.752885  0.05067277  1.226116
##   4                  210      1.747850  0.04962255  1.219703
##   4                  310      1.743611  0.05021500  1.214053
##   4                  410      1.741305  0.04938456  1.210052
##   4                  510      1.739207  0.04906328  1.205647
##   4                  610      1.738110  0.04800577  1.202087
##   4                  710      1.738333  0.04626422  1.199943
##   4                  810      1.739049  0.04472462  1.198445
##   4                  910      1.739429  0.04402936  1.196236
##   8                   10      1.758461  0.04936088  1.232110
##   8                  110      1.751670  0.05638718  1.224856
##   8                  210      1.746803  0.05639116  1.218571
##   8                  310      1.743280  0.05408854  1.213720
##   8                  410      1.741156  0.05082555  1.209850
##   8                  510      1.739771  0.04892054  1.206788
##   8                  610      1.738933  0.04791811  1.203253
##   8                  710      1.738597  0.04717928  1.201492
##   8                  810      1.738768  0.04657342  1.199172
##   8                  910      1.739396  0.04648938  1.197771
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.001
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 20
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 710,
##  interaction.depth = 2, shrinkage = 0.001 and n.minobsinnode = 20.
```

```
plot(gbmFit)
```




```
gbmFit$results[which.min(gbmFit$results$RMSE),1:6]
```

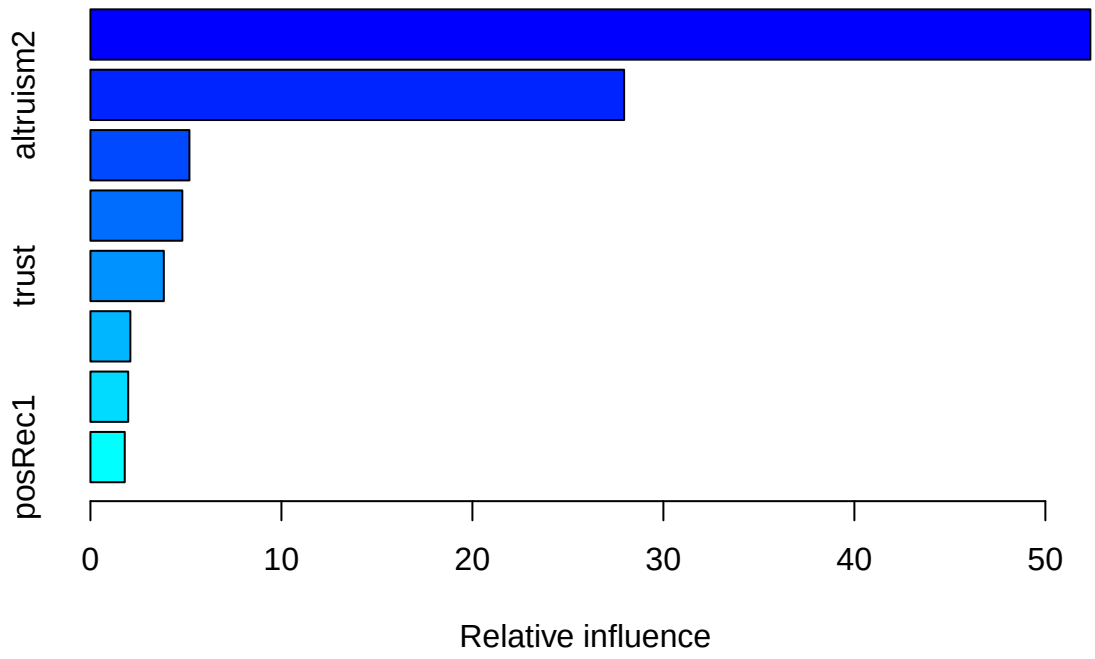
```
##    shrinkage interaction.depth n.minobsinnode n.trees     RMSE   Rsquared
## 18     0.001                 2             20     710 1.738014 0.04781028
```

```
mseBoost <- min(gbmFit$results$RMSE)^2
mseBoost
```

```
## [1] 3.020692
```

The most important variable is the same found in all other methods, posRec2. The second most important variable is altruism2. The relative influence of posRec2 is 52% and altruism2 is 28%, together having an importance of more than 80%.

```
set.seed(32)
gbmBestFit <- gbm(offer ~ ., data = trainOffer, distribution = "gaussian", n.trees= 710, n.minobsinnode
summary(gbmBestFit)
```




```
##                         var   rel.inf
## posRec2           posRec2 52.355861
## altruism2       altruism2 27.943124
## negRecPooled negRecPooled  5.182960
## risk2               risk2  4.811323
## trust               trust  3.841867
## risk1               risk1  2.090260
## altruism1       altruism1  1.977336
## posRec1           posRec1  1.797268
```

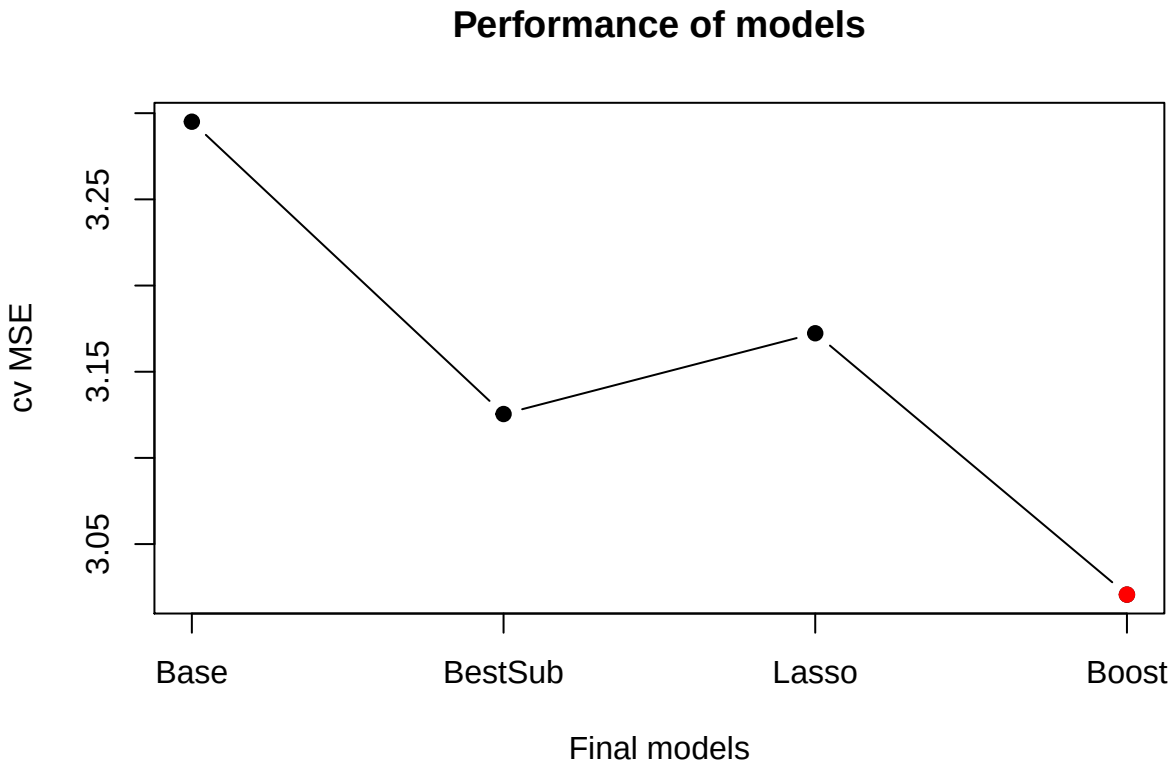
Final models

The final models are inspected. As can be seen, the best performing model is the boosting method. The best subset selection improves with more than 5% compared to the base model. The Lasso method performs slightly less, but has a lower standard deviation. A more complex method boosting decreases the RMSE to 3.02 and decreases compared to the base model with 8%. The boosting method is chosen.

```
finalModels <- data.frame(MSE = c(mseBase, mseBestSub, mseLasso, mseBoost), row.names = c("Base", "Best
finalModels
```

```
##              MSE
## Base    3.295048
## BestSub 3.125451
## Lasso   3.172326
## Boost   3.020692
```

```
plotModels(finalModels$MSE, xlab = "Final models", axislabels = rownames(finalModels))
```




```
## [1] 3.295048 3.125451 3.172326 3.020692
```

Final predictions

Now the final predictions on test set is performed. The model is already trained on all training data in the boosting section. The base model scores a MSE of 1.56 on the test set and a RMSE of 1.25. The best model scores slightly lower MSE with a 1.52 and a RMSE of 1.23. On the test set the final models performs barely 3% better than the base model, when measured in MSE.

```
ytrue <- qualtrics[test, "offer"]
baseMse <-  mean((ytrue - mean(trainOffer$offer))^2)
baseMse
```

```
## [1] 1.564879
```

```
sqrt(baseMse)
```

```
## [1] 1.250951
```

```
yhat <- predict(gbmBestFit, newdata = qualtrics[test,], n.trees = 710)
finalFit <- mean((ytrue - yhat)^2)
finalFit
```

```
## [1] 1.519434
```

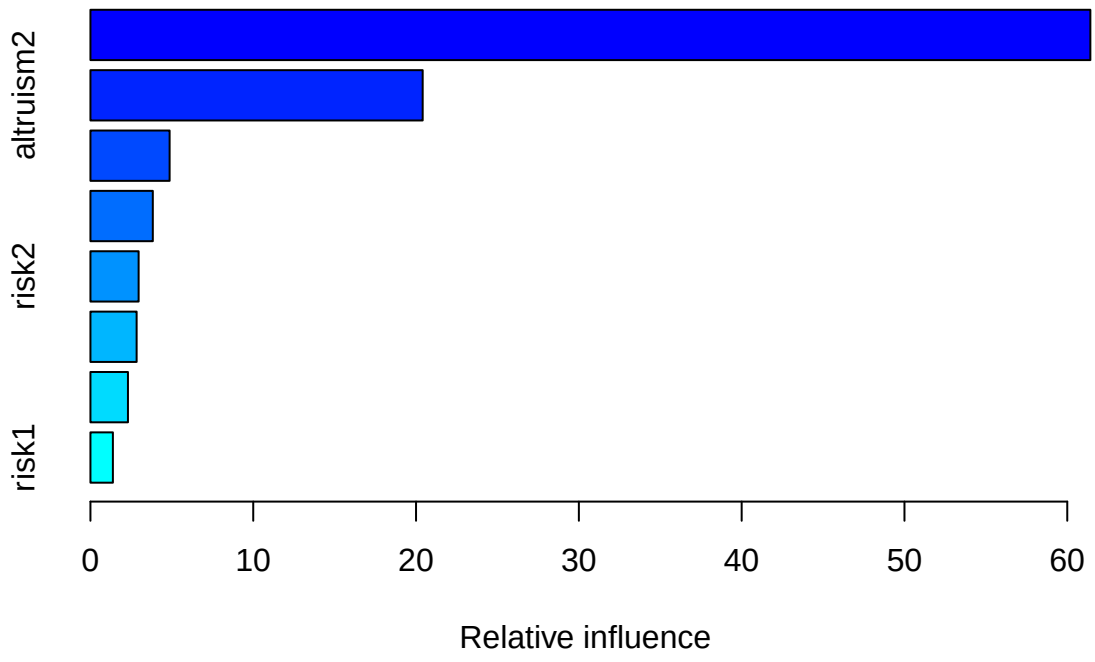
```
sqrt(finalFit)
```

```
## [1] 1.232653
```

Final model inspection

The best final model, boosting is trained on all available data and shows that the relative influence of posRec2 is 61% and of altruism2 is 20%. Together having an importance of greater than 80%.

```
qualtricsOffer <- select(qualtrics, -c(public, respond, dilemma, chicken))

set.seed(32)
gbmFinal <- gbm(offer ~ ., data = qualtricsOffer, distribution = "gaussian", n.trees= 710, n.minobsinnoc
summary(gbmFinal)
```

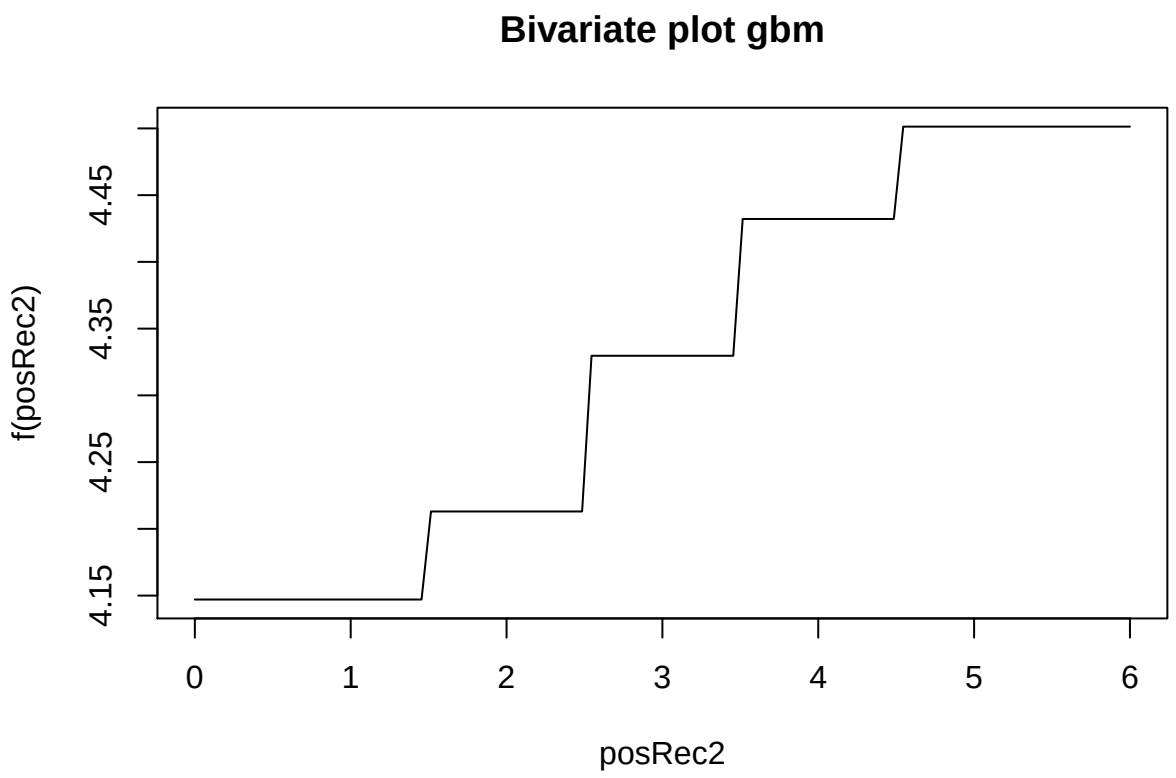



```
##                              var   rel.inf
## posRec2                  posRec2 61.418386
## altruism2              altruism2 20.406596
## altruism1              altruism1  4.863304
## trust                      trust  3.831322
## risk2                      risk2  2.960177
## negRecPooled        negRecPooled  2.836428
## posRec1                  posRec1  2.304127
## risk1                      risk1  1.379660
```

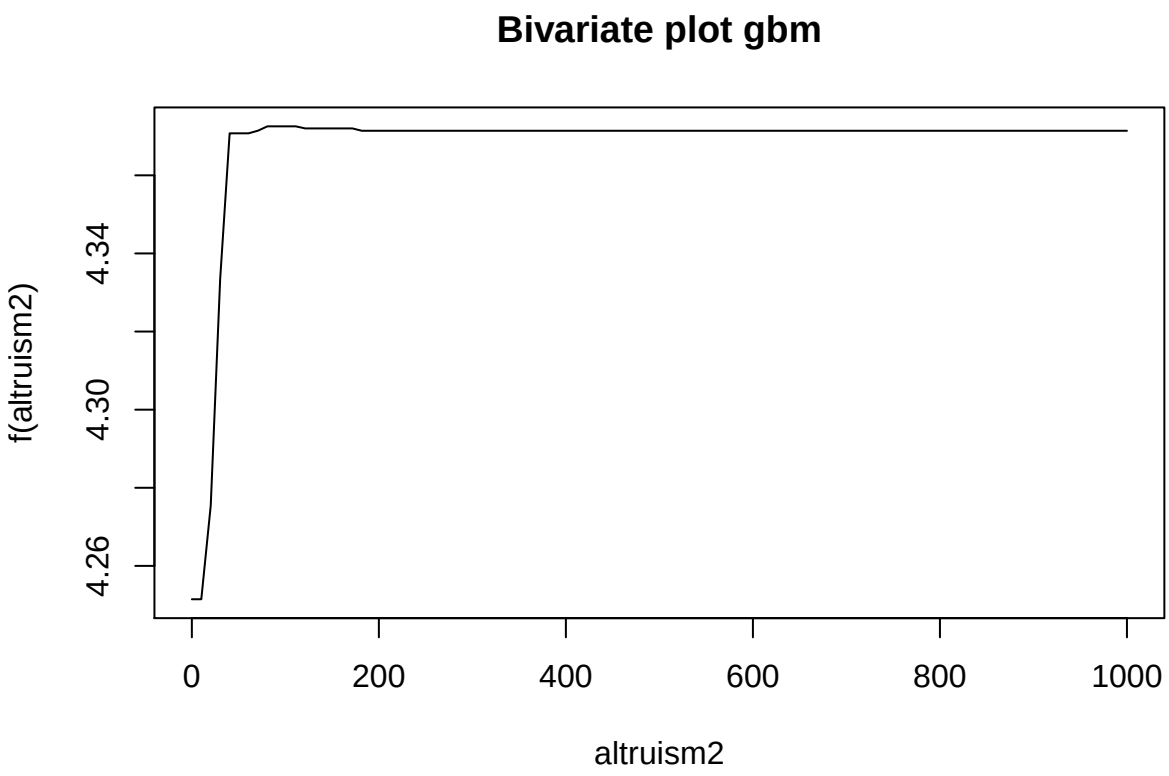
When this variable is plotted in more detail, it shows a step-wise positive relationship with posRec2 and the amount offered. Altruism2 very quickly rises and then after a certain amount, approximately 10-40 dollars it

stays fairly stable. So the higher amount of money donated to a good cause relates with higher offers up to a donation of 10-40 dollars, after this it has little explanatory value.

```
plot(gbmFinal, i = "posRec2", main = "Bivariate plot gbm")
```




```
plot(gbmFinal, i = "altruism2", main = "Bivariate plot gbm")
```



Respond

First a subset of trainQualtrics is taken with only the minimum acceptable amount in the ultimatum game as dependent variable and the preferences, all the other games are excluded.

```
trainRespond <- select(trainQualtrics, -c(public, offer, dilemma, chicken))
attach(trainRespond)
```

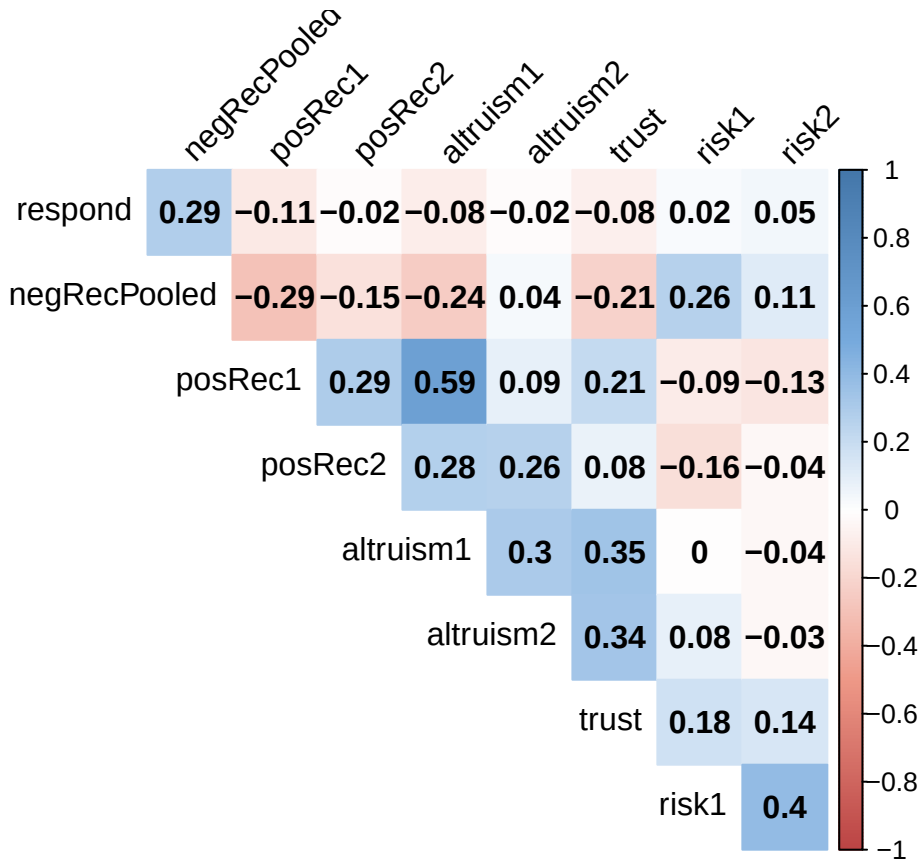
Bivariate analysis

For the same reason as above Spearman's rang correlation is chosen and bivariate relationships.

The following (absolute) correlations of greater than 0.10 are found:

- Positive relationship with negRecPooled (0.29)
- Negative relationship with posRec1 (-0.11).

```
plotCor(trainRespond)
```




```
## $cor
##              respond negRecPooled posRec1 posRec2 altruism1 altruism2
## respond         1.00         0.29   -0.11   -0.02     -0.08     -0.02
## negRecPooled    0.29         1.00   -0.29   -0.15     -0.24      0.04
## posRec1        -0.11        -0.29    1.00    0.29      0.59      0.09
## posRec2        -0.02        -0.15    0.29    1.00      0.28      0.26
## altruism1      -0.08        -0.24    0.59    0.28      1.00      0.30
## altruism2      -0.02         0.04    0.09    0.26      0.30      1.00
## trust          -0.08        -0.21    0.21    0.08      0.35      0.34
## risk1           0.02         0.26   -0.09   -0.16      0.00      0.08
## risk2           0.05         0.11   -0.13   -0.04     -0.04     -0.03
##              trust risk1 risk2
## respond      -0.08  0.02  0.05
## negRecPooled -0.21  0.26  0.11
## posRec1       0.21 -0.09 -0.13
## posRec2       0.08 -0.16 -0.04
## altruism1     0.35  0.00 -0.04
## altruism2     0.34  0.08 -0.03
## trust         1.00  0.18  0.14
## risk1         0.18  1.00  0.40
## risk2         0.14  0.40  1.00
```

Base model

A base model is conducted using the basemodelcv function. A cross-validated MSE score of 3.55, a standard deviation of 0.87 and a RMSE of 1.88 are found.

```
baseline <- baseModelcv(trainRespond, foldid)
baseline
```

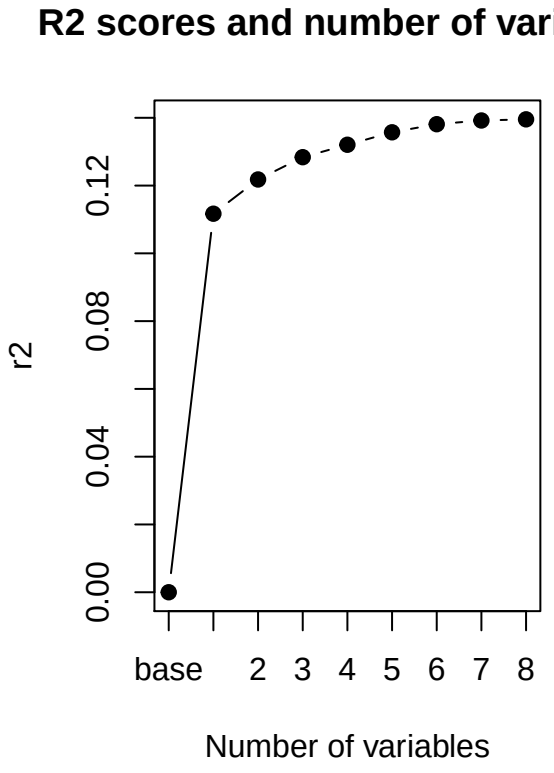
```
## $cv
## [1] 2.779033 4.147059 2.428417 4.326179 4.049524
##
## $meancv
## [1] 3.546042
##
## $sdcv
## [1] 0.8747452
##
## $rmse
## [1] 1.883094
```

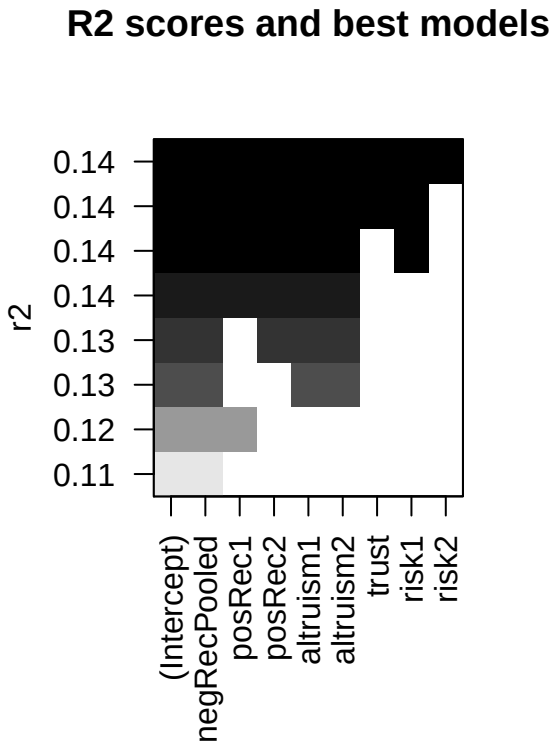
```
mseBase <- baseline$meancv
```

Best subset selection

A best subset selection is performed. As the plot shows negRecPooled explains a more than 11% of all the variance of respond. By adding any other variables, little increase is gained.

```
bestRespond <- regsubsets(respond ~ ., data = trainRespond)
plotR2Subs(bestRespond)
```






```
## Subset selection object
## Call: regsubsets.formula(respond ~ ., data = trainRespond)
## 8 Variables  (and intercept)
##              Forced in Forced out
## negRecPooled     FALSE      FALSE
## posRec1          FALSE      FALSE
## posRec2          FALSE      FALSE
## altruism1        FALSE      FALSE
## altruism2        FALSE      FALSE
## trust            FALSE      FALSE
## risk1            FALSE      FALSE
## risk2            FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          negRecPooled posRec1 posRec2 altruism1 altruism2 trust risk1
## 1  ( 1 ) "*"          " "     " "     " "       " "       " "   " "
## 2  ( 1 ) "*"          "*"     " "     " "       " "       " "   " "
## 3  ( 1 ) "*"          " "     " "     "*"       "*"       " "   " "
## 4  ( 1 ) "*"          " "     "*"     "*"       "*"       " "   " "
## 5  ( 1 ) "*"          "*"     "*"     "*"       "*"       " "   " "
## 6  ( 1 ) "*"          "*"     "*"     "*"       "*"       " "   "*"
## 7  ( 1 ) "*"          "*"     "*"     "*"       "*"       "*"   "*"
## 8  ( 1 ) "*"          "*"     "*"     "*"       "*"       "*"   "*"
##          risk2
## 1  ( 1 ) " "
## 2  ( 1 ) " "
```

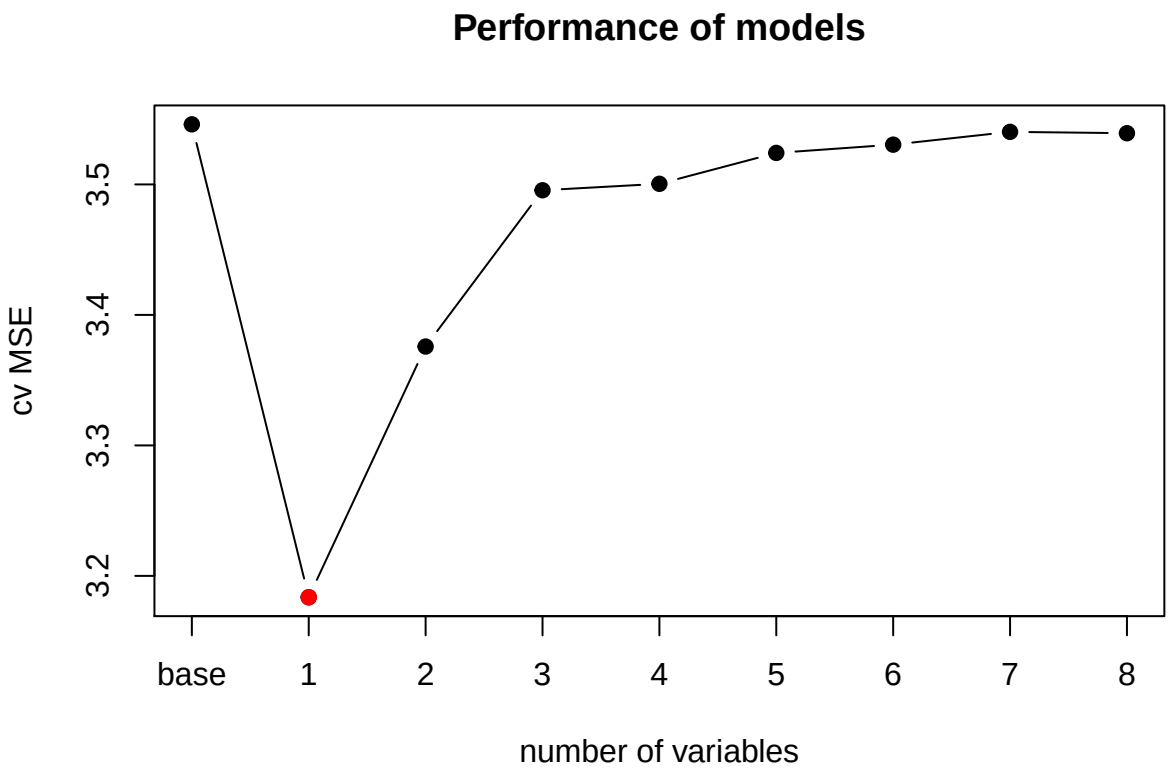
```
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) " "
## 7  ( 1 ) " "
## 8  ( 1 ) "*"
```

A best subset selection cross-validation is conducted using the bestsubsetcv function. The best performing model has one variable, as expected reviewing the R2 score above. The one variable model corresponds with a MSE of 3.18, a standard deviation of 0.45 and a RMSE of 1.78.

```
bestSubs <- bestSubsetcv(trainRespond, foldid)
bestSubs
```

```
## $cv
##             1        2        3        4        5        6        7
## [1,] 2.691973 3.259310 3.479347 3.558168 3.589364 3.566847 3.547417
## [2,] 3.556475 3.782918 3.951723 3.845288 3.828891 3.844892 3.946919
## [3,] 2.703440 2.877547 2.820629 2.820937 2.796779 2.817702 2.789355
## [4,] 3.420455 3.373848 3.384680 3.444494 3.525007 3.494453 3.484637
## [5,] 3.545820 3.585228 3.841313 3.833483 3.880693 3.928531 3.933130
##             8
## [1,] 3.548021
## [2,] 3.961416
## [3,] 2.776495
## [4,] 3.475343
## [5,] 3.934990
##
## $meancv
##        1        2        3        4        5        6        7        8
## 3.183633 3.375770 3.495538 3.500474 3.524147 3.530485 3.540292 3.539253
##
## $sdcv
##         1         2         3         4         5         6         7
## 0.4468191 0.3433055 0.4460153 0.4177395 0.4339676 0.4381702 0.4708293
##         8
## 0.4797893
##
## $bestnmdl
## [1] 1
##
## $bestnmean
## [1] 3.183633
##
## $bestnsd
## [1] 0.4468191
##
## $bestnrmse
## [1] 1.784274
```

```
mseBestSub <- bestSubs$bestnmean
plotModels(bestSubs$meancv, baseline)
```




```
##     base        1        2        3        4        5        6        7
## 3.546042 3.183633 3.375770 3.495538 3.500474 3.524147 3.530485 3.540292
##        8
## 3.539253
```

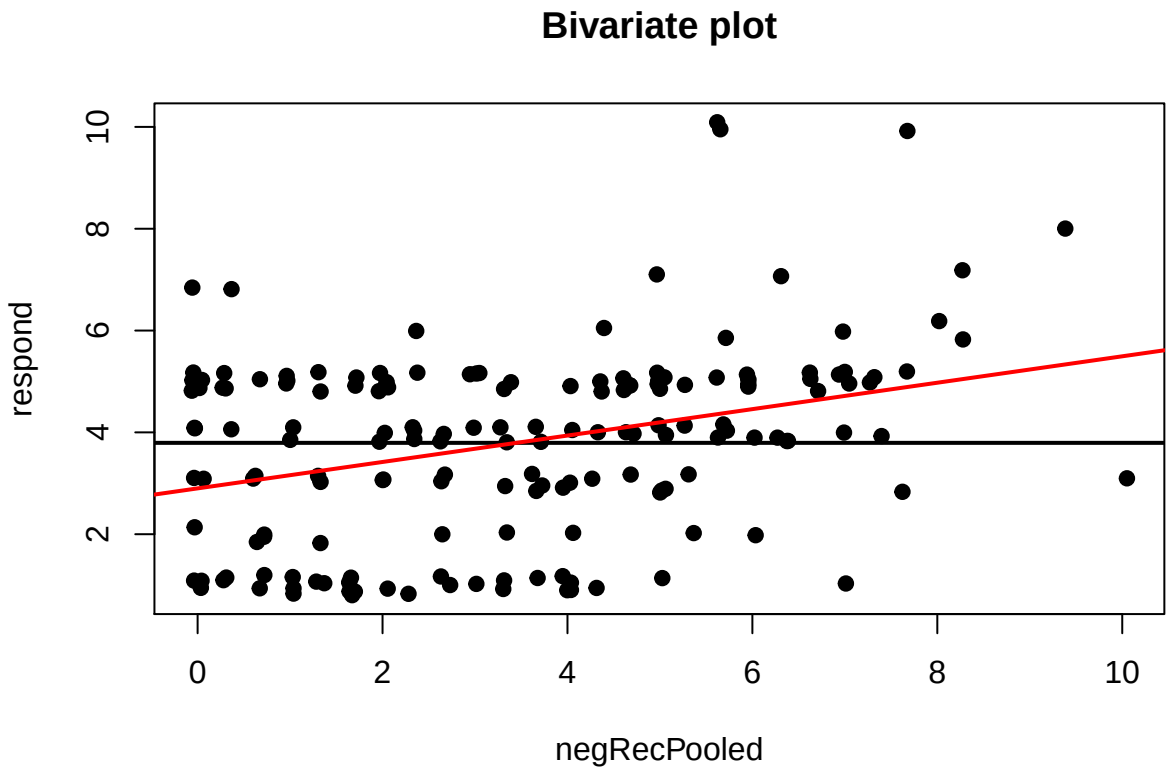
The variable for the best subset model is negRecPooled, this confirms with the findings of the R2 plot and the bivariate analysis. The corresponding coefficient is 0.26, which shows a positive relation. This means that when people score 1 point higher on the negRecPooled scale, the minimum acceptable amount increases on average with 0.26. Furthermore the model and the posRec2 variable is strongly significant (<0.001)

```
bestSubMdl <- respond ~ negRecPooled
bestSubFit <- lm(bestSubMdl, data = trainPublic)
summary(bestSubFit)
```

```
##
## Call:
## lm(formula = bestSubMdl, data = trainPublic)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7159 -1.1970  0.1733  1.0908  5.6300
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.89979    0.23734  12.218  < 2e-16 ***
## negRecPooled  0.25945    0.05645   4.596 8.42e-06 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.772 on 168 degrees of freedom
## Multiple R-squared:  0.1117, Adjusted R-squared:  0.1064
## F-statistic: 21.13 on 1 and 168 DF,  p-value: 8.419e-06
```

```
plotBivariate(negRecPooled, respond)
```



Lasso

The lasso model is conducted. The best performing model has a lambda of 0.25 and a corresponding MSE of 3.30 and a RMSE of 1.81, which is worse than the best subset selection method. However, the standard deviation of 0.27 is somewhat lower and results are somewhat more stable. As the plot shows, the optimal lambda includes one variable, however the lambda plus 1 standard error corresponds with the base model, this shows that the difference isn't very substantial.

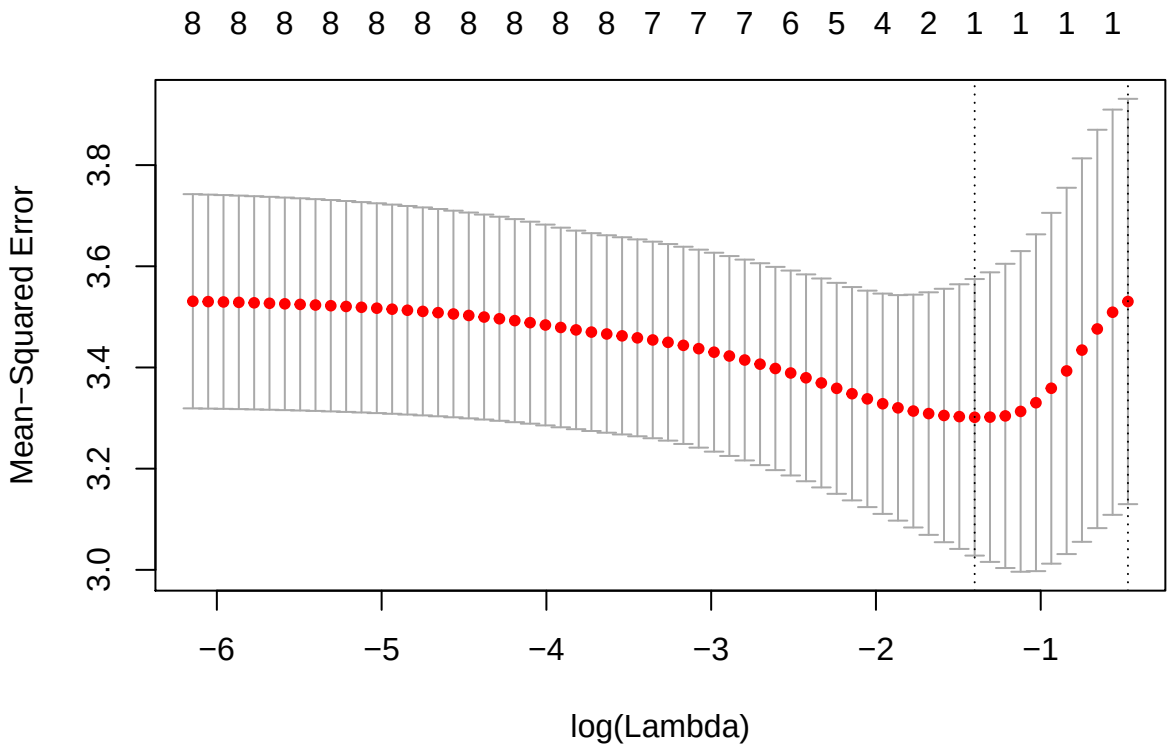
```
lassocv <- cv.glmnet(trainPreferences, respond, alpha = 1, foldid = foldid)
resultsLasso(lassocv)
```

```
## $glmnet.fit
##
## Call:  glmnet(x = trainPreferences, y = respond, alpha = 1)
##
##       Df    %Dev   Lambda
##  [1,]  0 0.00000 0.624600
##  [2,]  1 0.01896 0.569100
##  [3,]  1 0.03471 0.518600
```

```
##  [4,]  1 0.04778 0.472500
##  [5,]  1 0.05863 0.430500
##  [6,]  1 0.06764 0.392300
##  [7,]  1 0.07512 0.357400
##  [8,]  1 0.08133 0.325700
##  [9,]  1 0.08649 0.296700
## [10,]  1 0.09077 0.270400
## [11,]  1 0.09432 0.246400
## [12,]  2 0.09808 0.224500
## [13,]  2 0.10210 0.204500
## [14,]  2 0.10550 0.186400
## [15,]  2 0.10820 0.169800
## [16,]  3 0.11140 0.154700
## [17,]  4 0.11440 0.141000
## [18,]  4 0.11720 0.128500
## [19,]  4 0.11950 0.117000
## [20,]  5 0.12140 0.106600
## [21,]  5 0.12340 0.097170
## [22,]  6 0.12540 0.088540
## [23,]  6 0.12740 0.080670
## [24,]  6 0.12910 0.073510
## [25,]  7 0.13060 0.066980
## [26,]  7 0.13210 0.061030
## [27,]  7 0.13330 0.055610
## [28,]  7 0.13430 0.050670
## [29,]  7 0.13510 0.046160
## [30,]  7 0.13580 0.042060
## [31,]  7 0.13640 0.038330
## [32,]  7 0.13690 0.034920
## [33,]  7 0.13730 0.031820
## [34,]  7 0.13760 0.028990
## [35,]  8 0.13790 0.026420
## [36,]  8 0.13820 0.024070
## [37,]  8 0.13840 0.021930
## [38,]  8 0.13860 0.019980
## [39,]  8 0.13880 0.018210
## [40,]  8 0.13890 0.016590
## [41,]  8 0.13900 0.015120
## [42,]  8 0.13910 0.013770
## [43,]  8 0.13920 0.012550
## [44,]  8 0.13920 0.011440
## [45,]  8 0.13930 0.010420
## [46,]  8 0.13930 0.009494
## [47,]  8 0.13940 0.008650
## [48,]  8 0.13940 0.007882
## [49,]  8 0.13940 0.007182
## [50,]  8 0.13940 0.006544
## [51,]  8 0.13950 0.005962
## [52,]  8 0.13950 0.005433
## [53,]  8 0.13950 0.004950
## [54,]  8 0.13950 0.004510
## [55,]  8 0.13950 0.004110
## [56,]  8 0.13950 0.003745
## [57,]  8 0.13950 0.003412
```

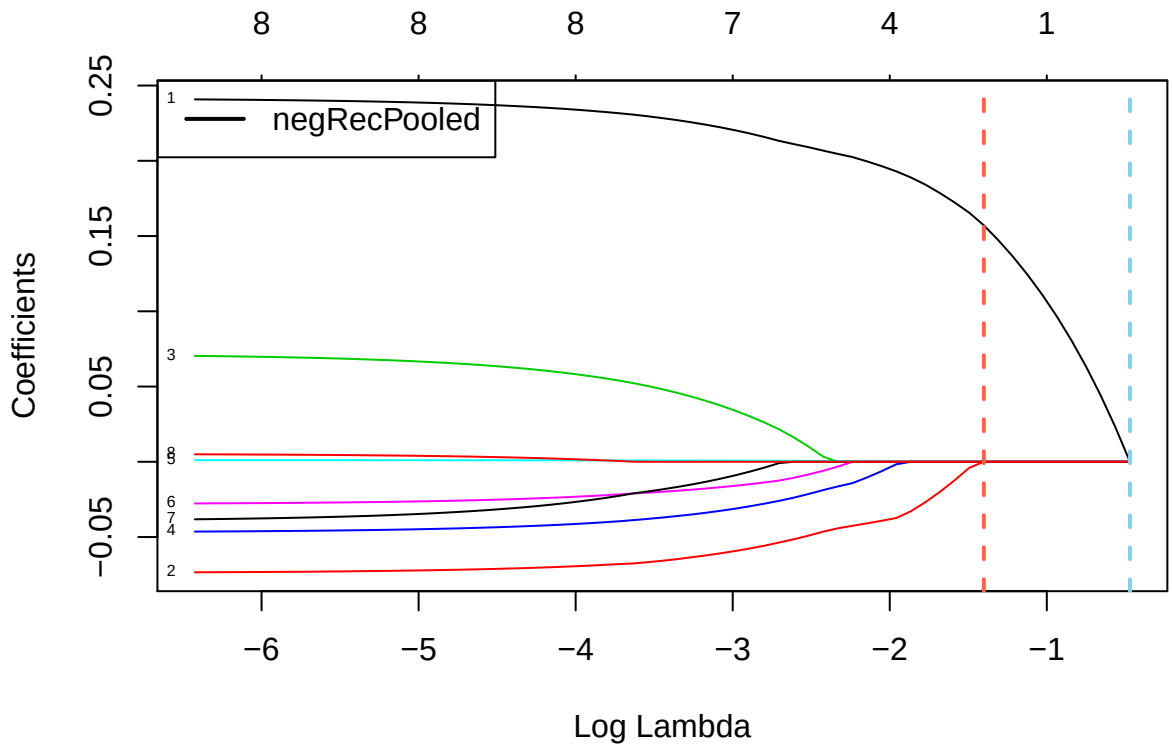
```
## [58,]  8 0.13950 0.003109
## [59,]  8 0.13950 0.002833
## [60,]  8 0.13950 0.002581
## [61,]  8 0.13950 0.002352
## [62,]  8 0.13950 0.002143
## [63,]  8 0.13950 0.001952
## [64,]  8 0.13950 0.001779
## [65,]  8 0.13950 0.001621
##
## $bestlambda
## [1] 0.2463651
##
## $bestnmean
## [1] 3.301508
##
## $bestnsd
## [1] 0.2732939
##
## $bestnrmse
## [1] 1.817005
##
## $lambda1se
## [1] 0.6246253
```

```
mseLasso <- min(lassocv$cvm)
plot(lassocv)
```



The shrinkage plot shows the shrank coefficients. The plot shows that the best lambda includes one variable, posRec2. This was also found via best subset selection method. The coefficient corresponding with this penalty factor is 0.157.

```
plotShrinkage(trainPreferences, respond, lassocv)
legend('topleft', legend = c("negRecPooled"), lwd = 2, col= c("black"))
```



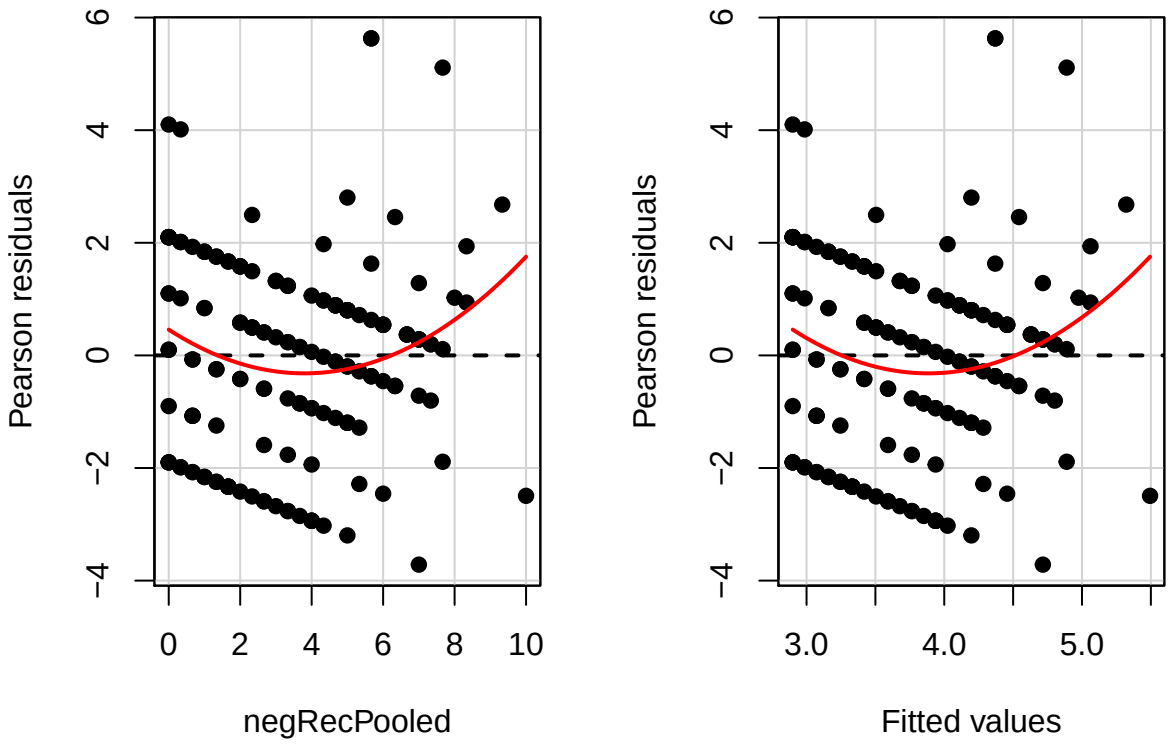

```
predict(lassocv, type = "coefficients", s = lassocv$lambda.min)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  3.2525321
## negRecPooled 0.1571153
## posRec1      .
## posRec2      .
## altruism1    .
## altruism2    .
## trust        .
## risk1        .
## risk2        .
```

Non-linearity

Non-linear relations of the best subset model are examined. The residual plot shows strong non-linearity mostly prevalent in negRecPooled, this is confirmed by the p-value of adding an negRecPooled second degree term of 0.016.

```
residualPlots((bestSubFit),pch=19)
```




```
##              Test stat Pr(>|t|)
## negRecPooled     2.444    0.016
## Tukey test       2.444    0.015
```

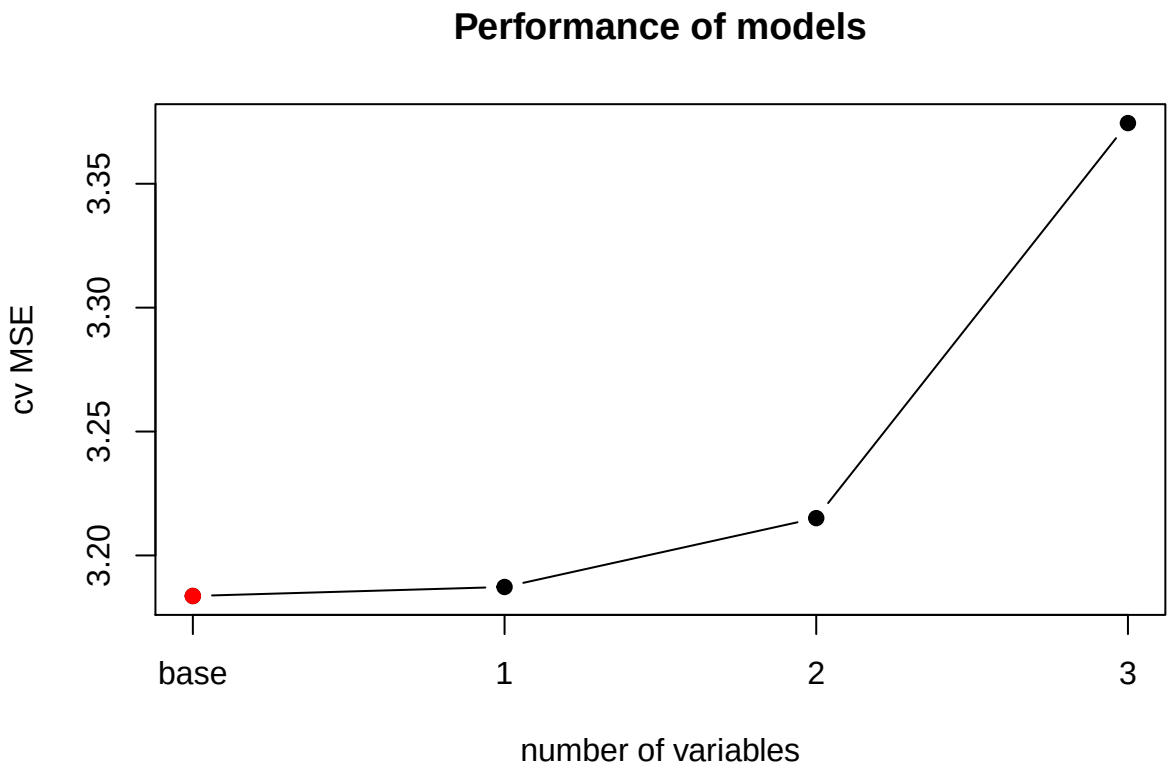
Therefore, models to the 4th degree are created. First a cross-validated nonlinear model is applied to test performance. However, the best performing model is the base model, unlike what the significance test shows, adding polynomials does not improve cross-validated MSE and increases the standard deviation strongly. Therefore, polynomials aren't added to the model.

```
mdl2 <- respond ~ poly(negRecPooled, 2)
mdl3 <- respond ~ poly(negRecPooled, 3)
mdl4 <- respond ~ poly(negRecPooled, 4)
models <- c(bestSubMdl, mdl2, mdl3, mdl4)

polynomial <- nonLinearcv(trainRespond, foldid, models)
polynomial
```

```
## $cv
##             1        2        3        4
## [1,] 2.691973 2.570541 2.635815 2.633299
## [2,] 3.556475 3.441973 3.215307 3.416455
## [3,] 2.703440 2.426184 2.477960 2.571071
## [4,] 3.420455 3.136464 3.622325 3.615034
## [5,] 3.545820 4.361170 4.123887 4.636564
##
```

```
## $meancv
##        1        2        3        4
## 3.183633 3.187267 3.215059 3.374484
##
## $sdcv
##         1         2         3         4
## 0.4468191 0.7754196 0.6838625 0.8436971
##
## $bestnmdl
## [1] 1
##
## $bestnmean
## [1] 3.183633
##
## $bestnsd
## [1] 0.4468191
##
## $bestnrmse
## [1] 1.784274
```

```
msePoly <- polynomial$bestnmean

plotModels(polynomial$meancv)
```




```
##        1        2        3        4
## 3.183633 3.187267 3.215059 3.374484
```

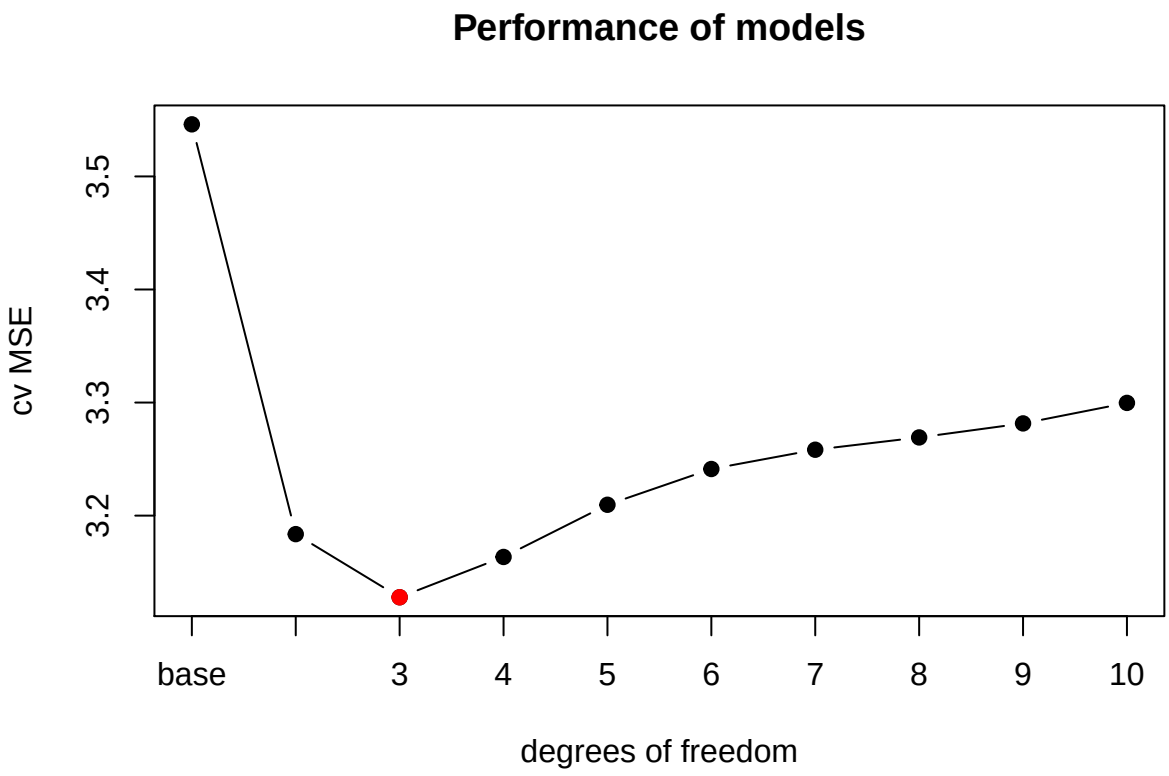
Smoothing spline

Next a smoothing spline is tried, to see whether this can capture the possible non-linear relationship. The best performing model is a smoothing spline with three degrees of freedom. The MSE is 3.13 which is lower than the best subset model which corresponds with 2 degrees of freedom. However, the standard deviation has increases as well.

```
smoothSpline <- smoothSplinecv(trainRespond, foldid, "negRecPooled", maxdegree = 10)
smoothSpline
```

```
## $cv
##             2        3        4        5        6        7        8
## [1,] 2.691968 2.587098 2.586529 2.601996 2.603408 2.598714 2.598892
## [2,] 3.556470 3.433549 3.375777 3.355598 3.346432 3.334805 3.325358
## [3,] 2.703433 2.499804 2.461204 2.494560 2.545428 2.604890 2.677603
## [4,] 3.420448 3.277705 3.378067 3.493505 3.538125 3.535854 3.516379
## [5,] 3.545825 3.840955 4.015663 4.102354 4.172800 4.217173 4.227548
##             9       10
## [1,] 2.607487 2.622174
## [2,] 3.325084 3.336518
## [3,] 2.767444 2.874481
## [4,] 3.496351 3.482285
## [5,] 4.211398 4.182971
##
## $meancv
##        2        3        4        5        6        7        8        9
## 3.183629 3.127822 3.163448 3.209603 3.241239 3.258287 3.269156 3.281553
##       10
## 3.299686
##
## $sdcv
##         2         3         4         5         6         7         8
## 0.4468214 0.5725560 0.6409752 0.6669693 0.6815412 0.6827013 0.6674543
##         9        10
## 0.6386152 0.6030575
##
## $bestnmdl
## [1] 2
##
## $bestdf
## [1] 3
##
## $bestnmean
## [1] 3.127822
##
## $bestnsd
## [1] 0.572556
##
## $bestnrmse
## [1] 1.768565
```

```
mseSmooth <- smoothSpline$bestnmean
plotModels(smoothSpline$meancv, baseline, xlab = "degrees of freedom", axislabels = c("base", "bestsub"
```




```
##     base        2        3        4        5        6        7        8
## 3.546042 3.183629 3.127822 3.163448 3.209603 3.241239 3.258287 3.269156
##        9       10
## 3.281553 3.299686
```

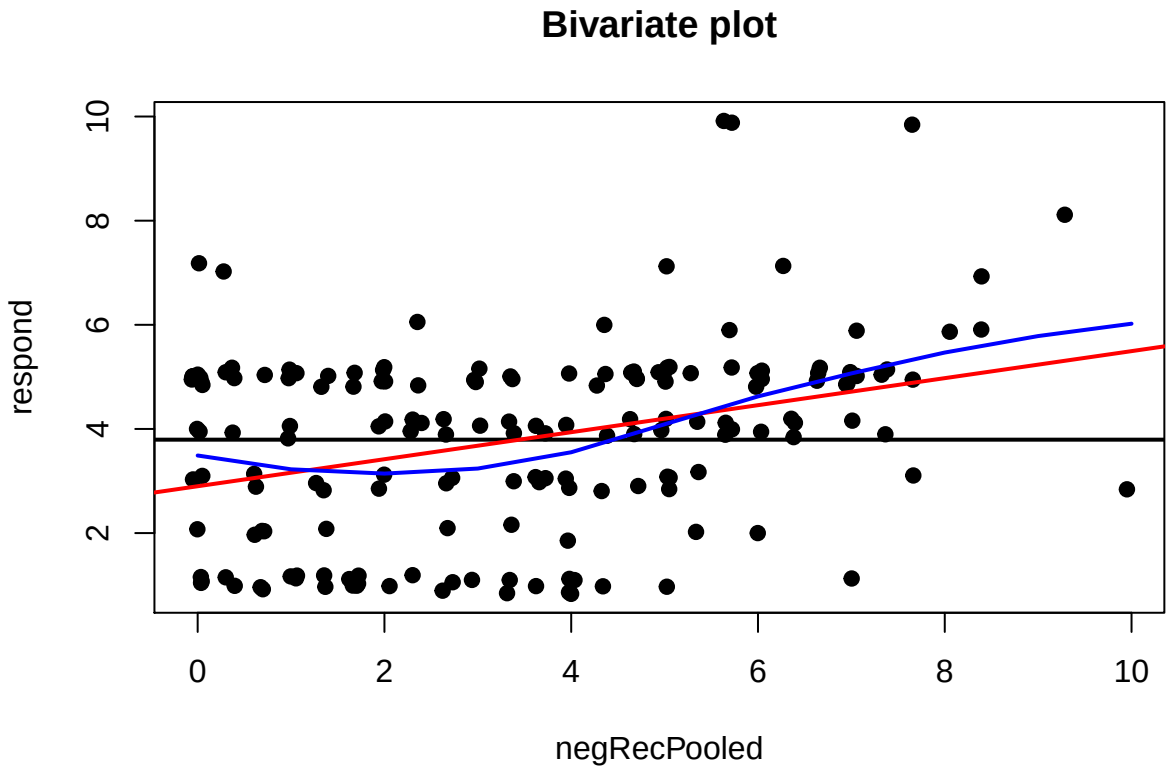
The best performing spline is constructed via a general additive model with a smoothing spline to third degree. It shows that negRecPooled with the smoothing spline is still significant. The bivariate plot includes the smoothing line in blue and it can be seen that it better fits respondents with higher negRecPooled scores, but might be slightly to wiggly before that.

```
gam3 <- gam(respond ~ s(negRecPooled, 3))
summary(gam3)
```

```
##
## Call: gam(formula = respond ~ s(negRecPooled, 3))
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -4.066691 -1.259976  0.003974  1.237179  5.541213
##
## (Dispersion Parameter for gaussian family taken to be 2.9853)
##
##     Null Deviance: 593.7941 on 169 degrees of freedom
## Residual Deviance: 495.5521 on 165.9998 degrees of freedom
## AIC: 674.318
##
## Number of Local Scoring Iterations: 2
##
```

```
## Anova for Parametric Effects
##                       Df Sum Sq Mean Sq F value    Pr(>F)
## s(negRecPooled, 3)     1  66.33  66.327  22.218 5.126e-06 ***
## Residuals            166 495.55   2.985
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                      Npar Df Npar F    Pr(F)
## (Intercept)
## s(negRecPooled, 3)         2 5.3451 0.005627 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
preds <- predict(gam3, newdata = list(negRecPooled = 0:max(negRecPooled)))

plotBivariate(negRecPooled, respond)
lines(0:max(negRecPooled), preds, col = "blue", lwd = 2)
```



Boosting

The final model is boosting. First the parameters are set to train the model.

```
fitControl <- trainControl(method = "cv", number = 5)
gbmGrid <-  expand.grid(interaction.depth = c(1, 2, 4, 8), n.trees = (1:10)*200, n.minobsinnode = 10, sh
```

A MSE of 3.06, and a RMSE of 1.75 is found, which is lower than found in other simpler models. The optimal

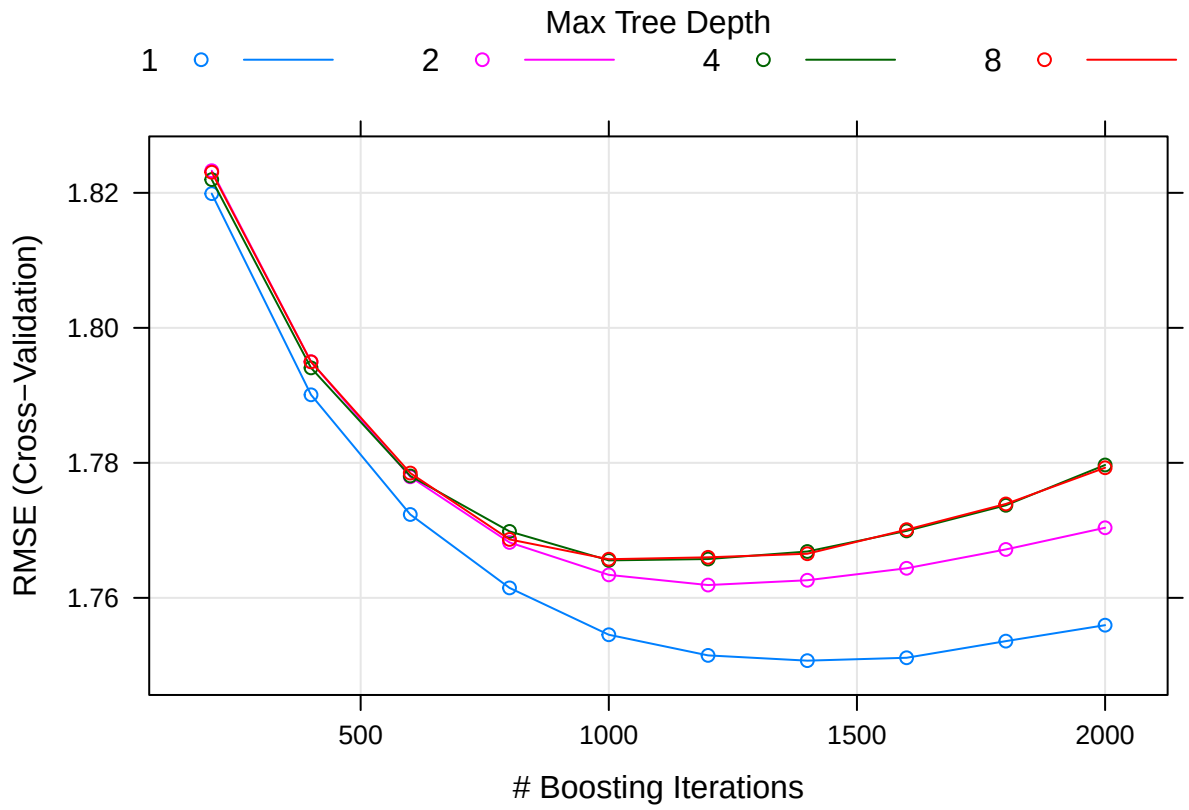
settings are 1400 trees, an interaction.depth of 1, n.minosinnode of 10 and a shrinkage factor of 0.001.

```
set.seed(64)
gbmFit <- train(respond ~ ., data = trainRespond, method = "gbm", verbose = FALSE, trControl = fitContr
gbmFit
```

```
## Stochastic Gradient Boosting
##
## 170 samples
##   8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 137, 136, 136, 136, 135
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared   MAE
##   1                   200     1.819882  0.1749525  1.448896
##   1                   400     1.790079  0.1714465  1.436644
##   1                   600     1.772352  0.1650928  1.430150
##   1                   800     1.761473  0.1608637  1.426877
##   1                  1000     1.754536  0.1575578  1.424066
##   1                  1200     1.751476  0.1533685  1.423879
##   1                  1400     1.750696  0.1491977  1.424533
##   1                  1600     1.751124  0.1456003  1.426729
##   1                  1800     1.753592  0.1408586  1.430611
##   1                  2000     1.755951  0.1370818  1.433682
##   2                   200     1.823279  0.1445017  1.453149
##   2                   400     1.794979  0.1425779  1.443904
##   2                   600     1.777933  0.1396577  1.439136
##   2                   800     1.768211  0.1372899  1.436443
##   2                  1000     1.763407  0.1339832  1.435514
##   2                  1200     1.761897  0.1303462  1.436661
##   2                  1400     1.762610  0.1263715  1.439326
##   2                  1600     1.764380  0.1232530  1.442558
##   2                  1800     1.767171  0.1199859  1.447169
##   2                  2000     1.770384  0.1169927  1.451117
##   4                   200     1.821963  0.1412922  1.453059
##   4                   400     1.794070  0.1340700  1.444014
##   4                   600     1.778054  0.1302454  1.439610
##   4                   800     1.769840  0.1254248  1.438095
##   4                  1000     1.765552  0.1222641  1.436848
##   4                  1200     1.765749  0.1178524  1.439924
##   4                  1400     1.766868  0.1146180  1.442803
##   4                  1600     1.769921  0.1117680  1.448267
##   4                  1800     1.773711  0.1089611  1.453034
##   4                  2000     1.779680  0.1049104  1.459230
##   8                   200     1.823051  0.1296732  1.453512
##   8                   400     1.794961  0.1294060  1.445280
##   8                   600     1.778489  0.1276879  1.440873
##   8                   800     1.768655  0.1266635  1.438338
##   8                  1000     1.765725  0.1221262  1.438070
##   8                  1200     1.766006  0.1178451  1.440776
##   8                  1400     1.766535  0.1161849  1.443220
##   8                  1600     1.770103  0.1128629  1.447665
```

```
##   8                  1800     1.773911  0.1105109  1.452494
##   8                  2000     1.779276  0.1066506  1.458017
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.001
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 1400,
##  interaction.depth = 1, shrinkage = 0.001 and n.minobsinnode = 10.
```

```
plot(gbmFit)
```




```
gbmFit$results[which.min(gbmFit$results$RMSE),1:6]
```

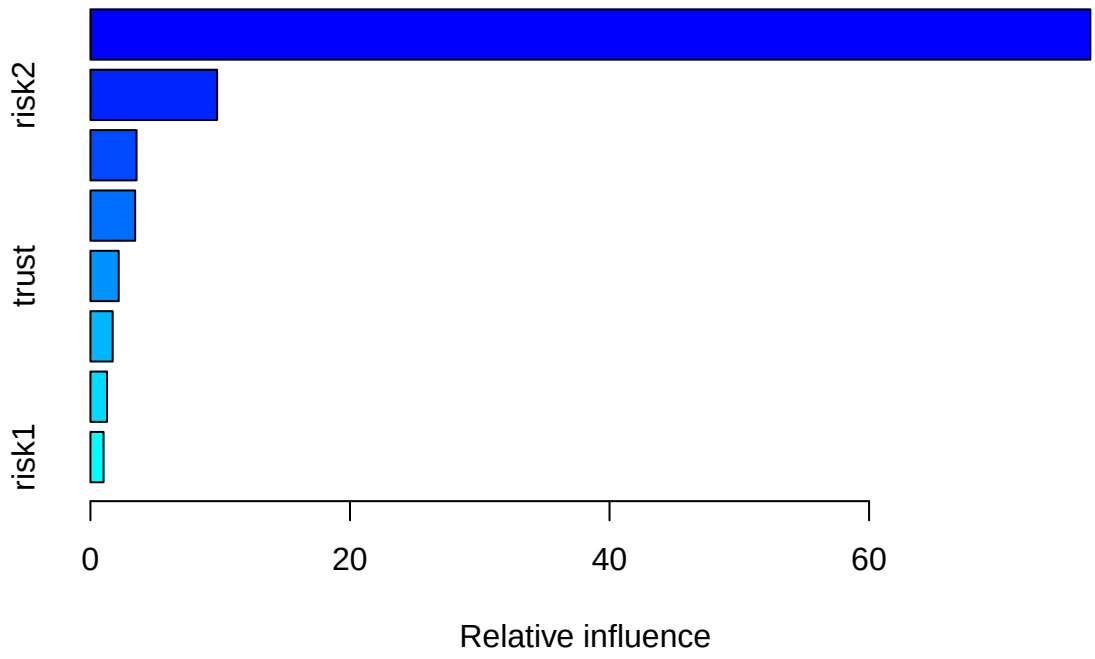
```
##   shrinkage interaction.depth n.minobsinnode n.trees     RMSE  Rsquared
## 7     0.001                 1             10    1400 1.750696 0.1491977
```

```
mseBoost <- min(gbmFit$results$RMSE)^2
mseBoost
```

```
## [1] 3.064937
```

The most important variable is the same found in all other methods negRecpooled, the second most important variable is risk2. The relative influence of negRecPooled is 77%, followed by risk2 with 10%.

```
set.seed(32)
gbmBestFit <- gbm(respond ~ ., data = trainRespond, distribution = "gaussian", n.trees = 1400)
summary(gbmBestFit)
```




```
##                         var   rel.inf
## negRecPooled negRecPooled 77.050141
## risk2               risk2  9.759750
## posRec1           posRec1  3.555875
## altruism2       altruism2  3.445885
## trust               trust  2.180779
## posRec2           posRec2  1.716807
## altruism1       altruism1  1.279875
## risk1               risk1  1.010890
```

Final models

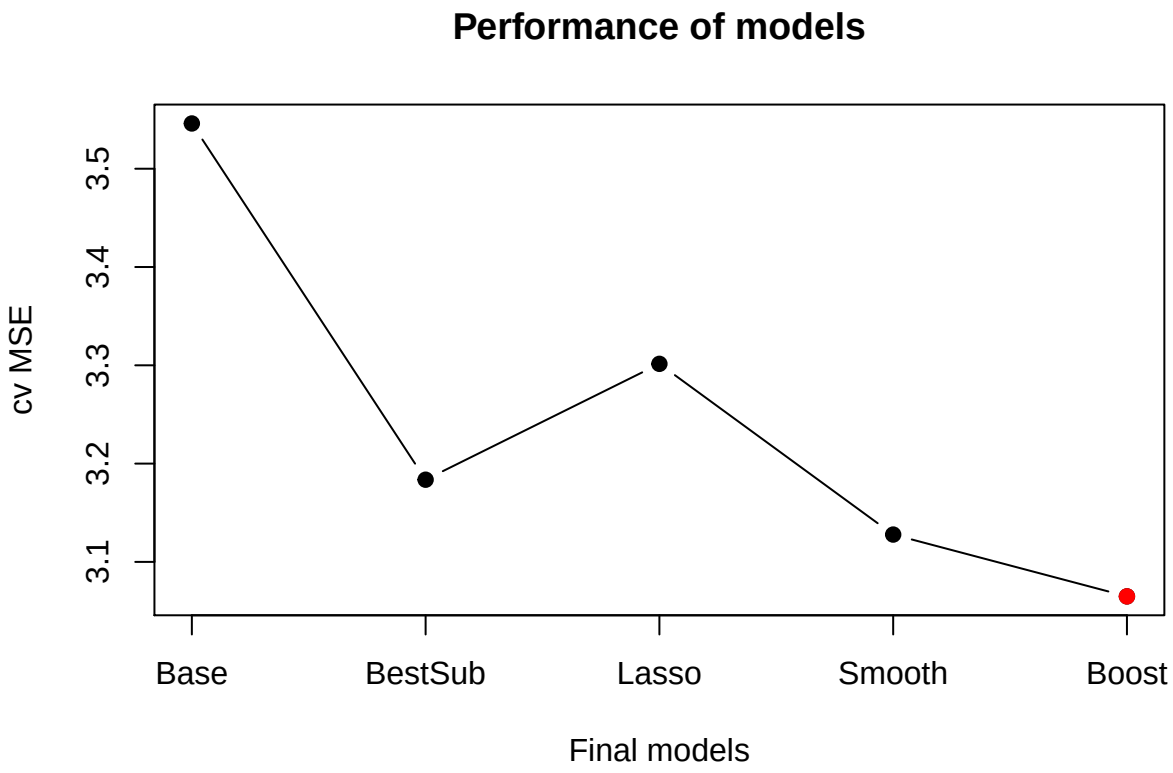
The final models are inspected. As can be seen, the best performing model is the boosting method. The best subset selection improves with more than 10% compared to the base model. The Lasso method performs worse. Adding a polynomial to the best subset selection model was not helpful. However a smoothing spline to the third degree improved compared to the best subset selection with a small 2%, however the standard deviation increases too. The more complex boosting method decreased the MSE to 3.06 and decreased compared to the base model with more than 14%. The boosting method is chosen.

```
finalModels <- data.frame(MSE = c(mseBase, mseBestSub, mseLasso, mseSmooth, mseBoost), row.names = c("Ba
finalModels
```

```
##              MSE
## Base    3.546042
## BestSub 3.183633
## Lasso   3.301508
```

```
## Smooth  3.127822
## Boost   3.064937
```

```
plotModels(finalModels$MSE, xlab = "Final models", axislabels = rownames(finalModels))
```




```
## [1] 3.546042 3.183633 3.301508 3.127822 3.064937
```

Final predictions

Now the final predictions on test set is performed. The model is already trained on all training data in the boosting section. The base model scores a MSE of 2.15 on the test set and a RMSE of 1.47. The best model performs better and has a MSE with 2.00 and a RMSE of 1.42. On the test set the final models performs 7% better than the base model, when measured in MSE.

```
ytrue <- qualtrics[test, "respond"]
baseMse <- mean((ytrue - mean(respond))^2)
baseMse
```

```
## [1] 2.149654
```

```
sqrt(baseMse)
```

```
## [1] 1.46617
```

```
yhat <- predict(gbmBestFit, newdata = qualtrics[test,], n.trees = 1400)
finalFit <- mean((ytrue - yhat)^2)
finalFit
```

```
## [1] 1.984596
```

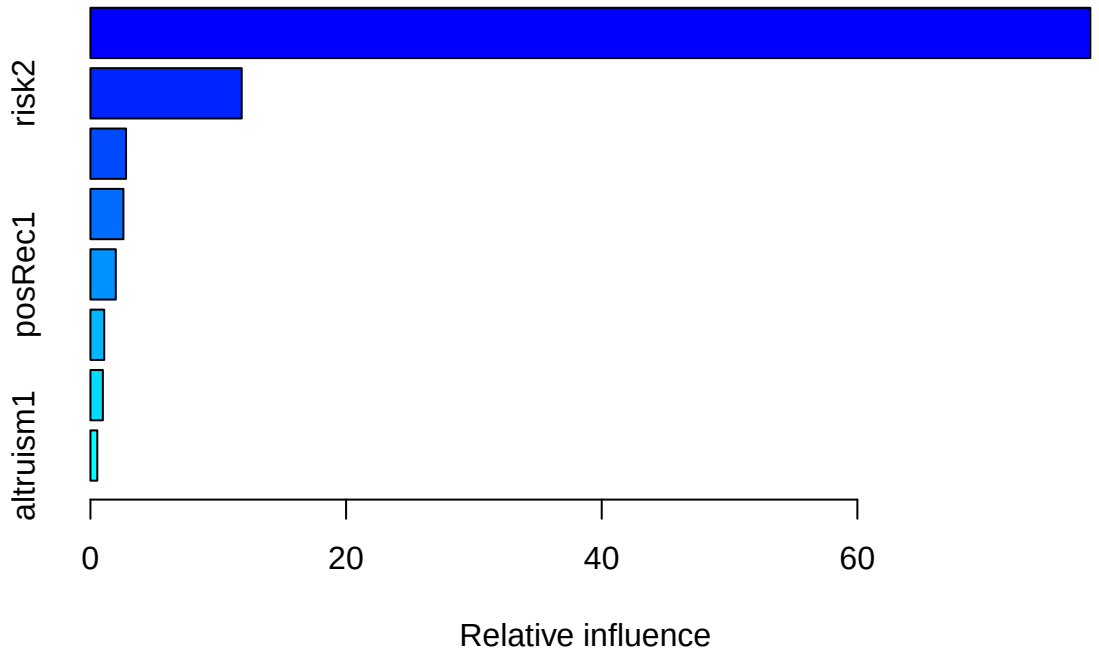
```
sqrt(finalFit)
```

```
## [1] 1.408757
```

Final model inspection

The best final model, boosting is trained on all available data and shows that the relative influence of negRecPooled is 78% and of risk2 is 12%. Together having an importance of around 90%.

```
qualtricsRespond <- select(qualtrics, -c(public, offer, dilemma, chicken))

set.seed(32)
gbmBest <- gbm(respond ~ ., data = qualtricsRespond, distribution = "gaussian", n.trees = 1400, interac
summary(gbmBest)
```

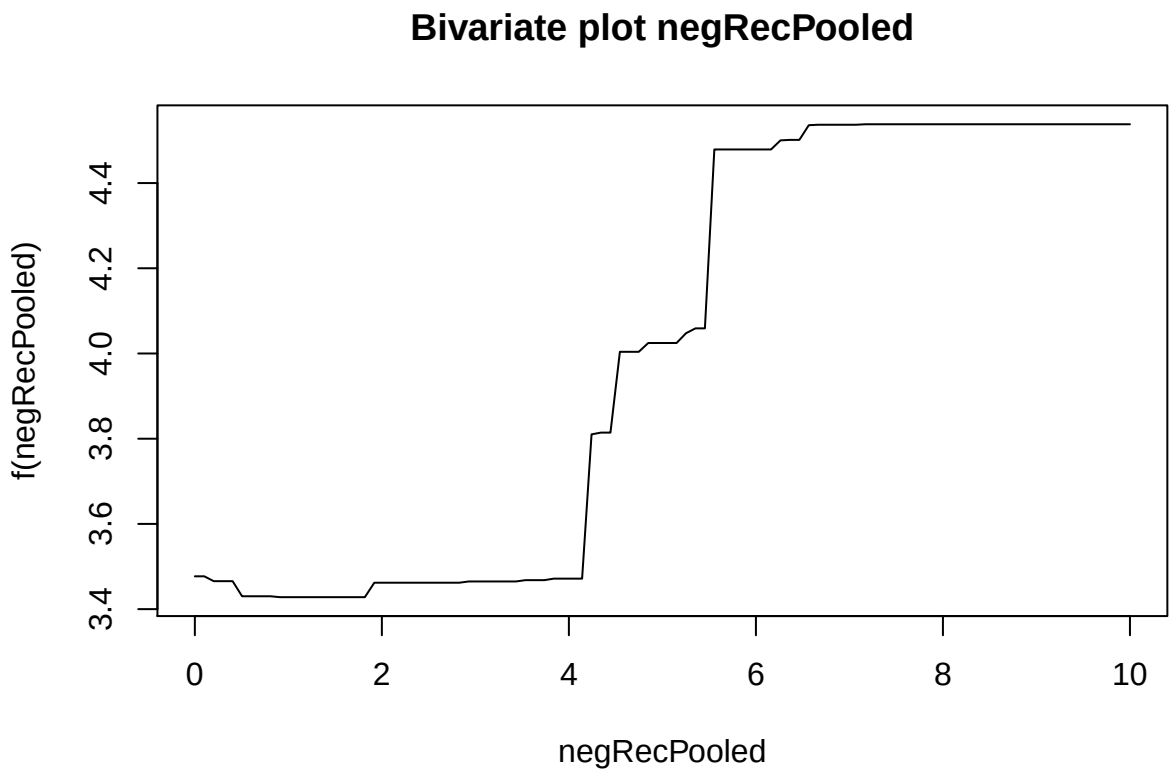



```
##                       var    rel.inf
## negRecPooled negRecPooled 78.2282738
## risk2               risk2 11.8355545
## trust               trust  2.7851533
## altruism2       altruism2  2.5802977
## posRec1           posRec1  1.9850001
## risk1               risk1  1.0775695
## posRec2           posRec2  0.9711512
## altruism1       altruism1  0.5369998
```

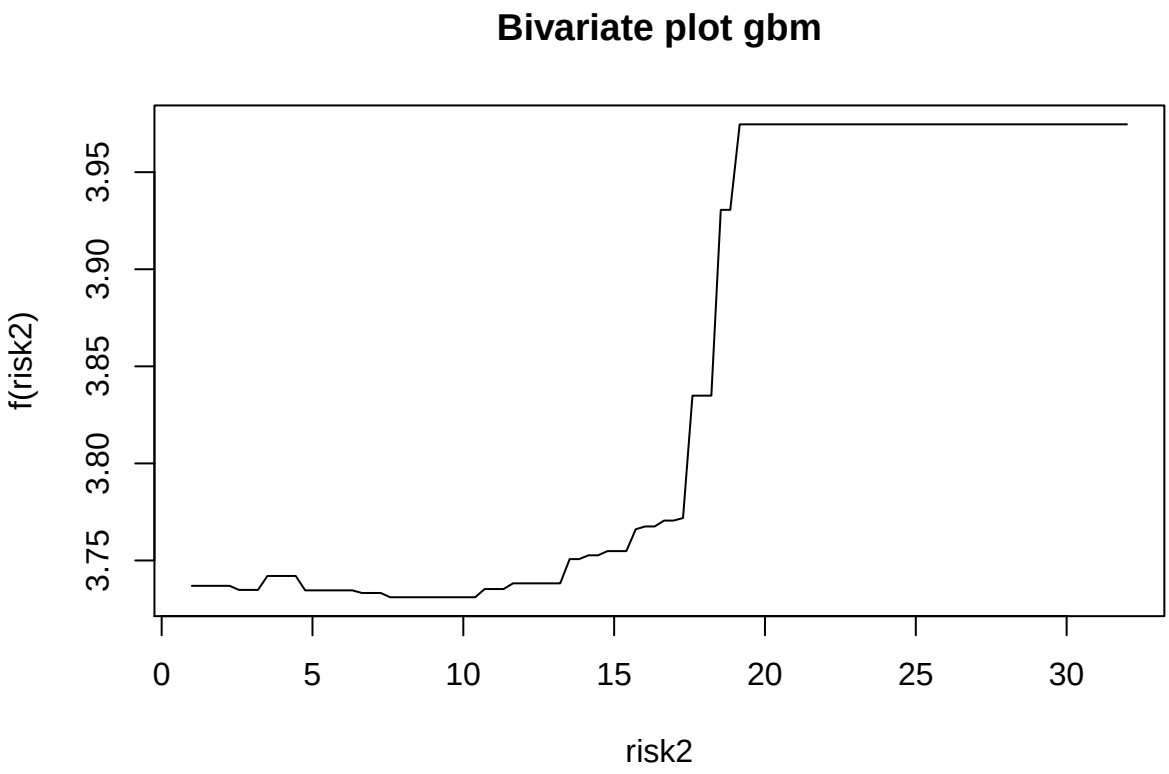
When these variables are plotted in more detail, it shows a positive relationship with negRecPooled and the minimum acceptable offer, but this only seems to increase very strongly after a score of 4 until 6. A score

smaller than 4 scores stable, just as a score greater than 6. For risk2 almost the same pattern is found but here the increase starts slowly at 13, but very strongly at a score of 17 to 19, which approximately corresponds with accepting a sure payment of 165 to 185 instead of 50% chance of receiving 300. The minimum acceptable amount is thus higher for person that are risk-seeking, where risk-neutral corresponds 150 dollars. After and before this the minimum acceptable offer stays fairly stable.

```
plot(gbmBest, i = "negRecPooled", main = "Bivariate plot negRecPooled")
```




```
plot(gbmBest, i = "risk2", main = "Bivariate plot gbm")
```



Prisoner's dilemma

First a subset of trainQualtrics is taken with only prisoner's dilemma as dependent variable and the preferences, all the other games are excluded. Furthermore a binary variable is made, where 0 corresponds with cooperate and 1 with defect. All models are, unlike the previous games, classification models.

```
trainDilemma <- select(trainQualtrics, -c(chicken, offer, respond, public))
attach(trainDilemma)
dilemmaBinary <- ifelse(dilemma == "cooperate", 0, 1)
```

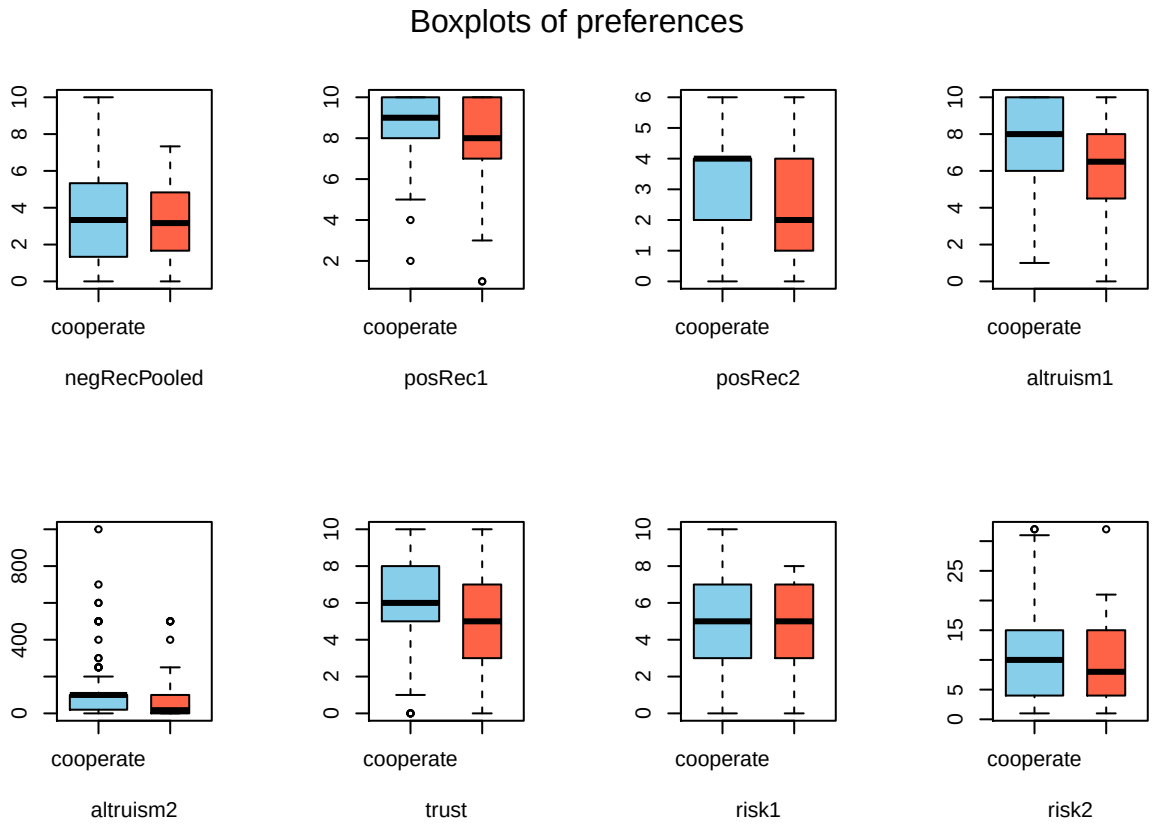
Bivariate analysis

The distributions are inspected via boxplots. The following main differences in distribution are found.

- A negative relationship with posRec1
- A negative relationship with posRec2
- A negative relationship with altruism1
- A negative relationship with altruism2
- A negative relationship with trust

Where a negative relationship corresponds with a distribution more tended to the lower spectrum for that preference for the defecting group, or the binary 1 compared to the cooperate group.

```
plotBoxes(trainQualtrics, dilemma)
```



Base model

A base model is conducted using the basemodelcv function, instead of using the mean as baseline it uses the majority class to predict, which in all cases is cooperate. The following accuracies are found, a mean cross-validation accuracy of 0.69 (69%) and a standard deviation of 0.10 (10%).

```
baseline <- baseModelcv(trainDilemma, foldid)
baseline
```

```
## $cv
## [1] 0.6470588 0.8235294 0.5588235 0.7058824 0.7352941
##
## $meancv
## [1] 0.6941176
##
## $sdcv
## [1] 0.09886904
```

```
accBase <- baseline$meancv
```

Best subset selection

A best subset selection method is performed on the all trainings data and then cross-validation is used to chose the best model for every number of variables, this method uses a logistic regression. The best model contains 1 variable, altruism1. This variable correspond with the differences found in the bivariate analysis.

```
set.seed(32)
bestDilemma <- bestglm(data.frame(trainPreferences, dilemmaBinary), IC = "CV", CVArgs = list(Method="HT
```

```
## Morgan-Tatar search since family is non-gaussian.
```

```
bestDilemma$Subsets
```

```
##    Intercept negRecPooled posRec1 posRec2 altruism1 altruism2 trust risk1
## 0       TRUE        FALSE   FALSE   FALSE     FALSE     FALSE FALSE FALSE
## 1*      TRUE        FALSE   FALSE   FALSE      TRUE     FALSE FALSE FALSE
## 2       TRUE        FALSE   FALSE   FALSE      TRUE     FALSE  TRUE FALSE
## 3       TRUE         TRUE   FALSE   FALSE      TRUE     FALSE  TRUE FALSE
## 4       TRUE         TRUE   FALSE    TRUE      TRUE     FALSE  TRUE FALSE
## 5       TRUE         TRUE   FALSE    TRUE      TRUE      TRUE  TRUE FALSE
## 6       TRUE         TRUE   FALSE    TRUE      TRUE      TRUE  TRUE FALSE
## 7       TRUE         TRUE   FALSE    TRUE      TRUE      TRUE  TRUE  TRUE
## 8       TRUE         TRUE    TRUE    TRUE      TRUE      TRUE  TRUE  TRUE
##    risk2 logLikelihood        CV        sdCV
## 0  FALSE    -104.68028 0.2139922 0.009796364
## 1* FALSE     -95.96515 0.1931352 0.010057494
## 2  FALSE     -94.09167 0.2016100 0.023580208
## 3  FALSE     -92.27832 0.1925178 0.016908494
## 4  FALSE     -91.38390 0.1922879 0.022411046
## 5  FALSE     -91.19767 0.1970894 0.023064290
## 6   TRUE     -91.11187 0.1915853 0.012744675
## 7   TRUE     -91.07136 0.1982825 0.019858549
## 8   TRUE     -91.03406 0.2039413 0.015646667
```

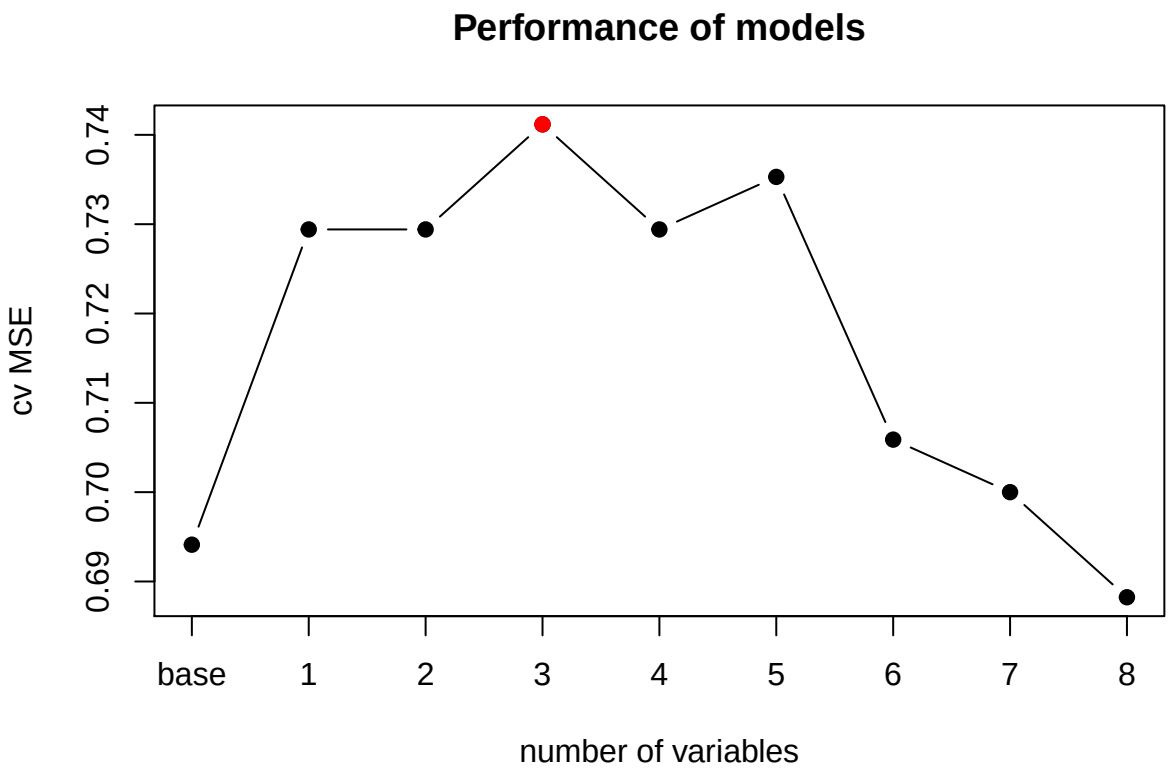
Next, since these cross-validation scores use different folds and are not comparable with the other models, the found best models are used as input for bestsubsetcv function and then a logistic regression is applied. The models are manually constructed. Using these foldid's, a different optimal model is found, the best performing model has three variables, altruism1, trust and negRecPooled. The first two corresponds with the findings of the bivariate analysis, the latter not. These variables correspond with a accuracy of 0.741 (74.1%) and a standard deviation of 0.12 (12%).

```
mdl1 <- dilemma ~ altruism1
mdl2 <- dilemma ~ altruism1 + trust
mdl3 <- dilemma ~ altruism1 + trust + negRecPooled
mdl4 <- dilemma ~ altruism1 + trust + negRecPooled + posRec2
mdl5 <- dilemma ~ altruism1 + trust + negRecPooled + posRec2 + altruism2
mdl6 <- dilemma ~ altruism1 + trust + negRecPooled + posRec2 + altruism2 + risk2
mdl7 <- dilemma ~ altruism1 + trust + negRecPooled + posRec2 + altruism2 + risk2 + risk1
mdl8 <- dilemma ~ .
models <- c(mdl1, mdl2, mdl3, mdl4, mdl5, mdl6, mdl7, mdl8)

bestSubs <- bestSubsetcv(trainDilemma, foldid, models)
bestSubs
```

```
## $cv
##              1         2         3         4         5         6         7
## [1,] 0.6764706 0.7058824 0.6764706 0.7058824 0.7058824 0.6764706 0.6764706
## [2,] 0.8823529 0.9117647 0.8823529 0.8529412 0.8529412 0.7941176 0.7941176
## [3,] 0.5882353 0.5588235 0.5882353 0.5882353 0.5882353 0.5882353 0.5294118
## [4,] 0.6764706 0.6470588 0.7352941 0.7058824 0.7058824 0.7058824 0.7058824
## [5,] 0.8235294 0.8235294 0.8235294 0.7941176 0.8235294 0.7647059 0.7941176
##              8
```

```
## [1,] 0.6470588
## [2,] 0.7941176
## [3,] 0.5294118
## [4,] 0.7058824
## [5,] 0.7647059
##
## $meancv
##         1         2         3         4         5         6         7
## 0.7294118 0.7294118 0.7411765 0.7294118 0.7352941 0.7058824 0.7000000
##         8
## 0.6882353
##
## $sdcv
##          1          2          3          4          5          6
## 0.12019304 0.14013093 0.11653890 0.10060371 0.10604563 0.08054743
##          7          8
## 0.10886327 0.10522673
##
## $bestnmdl
## [1] 3
##
## $bestnmean
## [1] 0.7411765
##
## $bestnsd
## [1] 0.1165389
```

```
accBestSub <- bestSubs$bestnmean

plotModels(bestSubs$meancv, baseline, classification = TRUE)
```




```
##      base         1         2         3         4         5         6
## 0.6941176 0.7294118 0.7294118 0.7411765 0.7294118 0.7352941 0.7058824
##         7         8
## 0.7000000 0.6882353
```

As mentioned before, the variables for the best model are altruism1, trust and negRecPooled. The AIC of the final model is 192.56. The corresponding Log coefficients are -0.278 for altruism1, -0.16 for trust and -0.152 for negRecPooled, these are all negative relationships. The first two were also found in the bivariate analysis. Altruism1 is very significant (0.001) and trust is moderately significant (0.03), negRecPooled is weakly significant with a p-value of (0.063). Since all variables are on the same scale the coefficients can be compared and show that altruism1 is the most important variable followed by trust and negRecPooled, which have about the same coefficients.

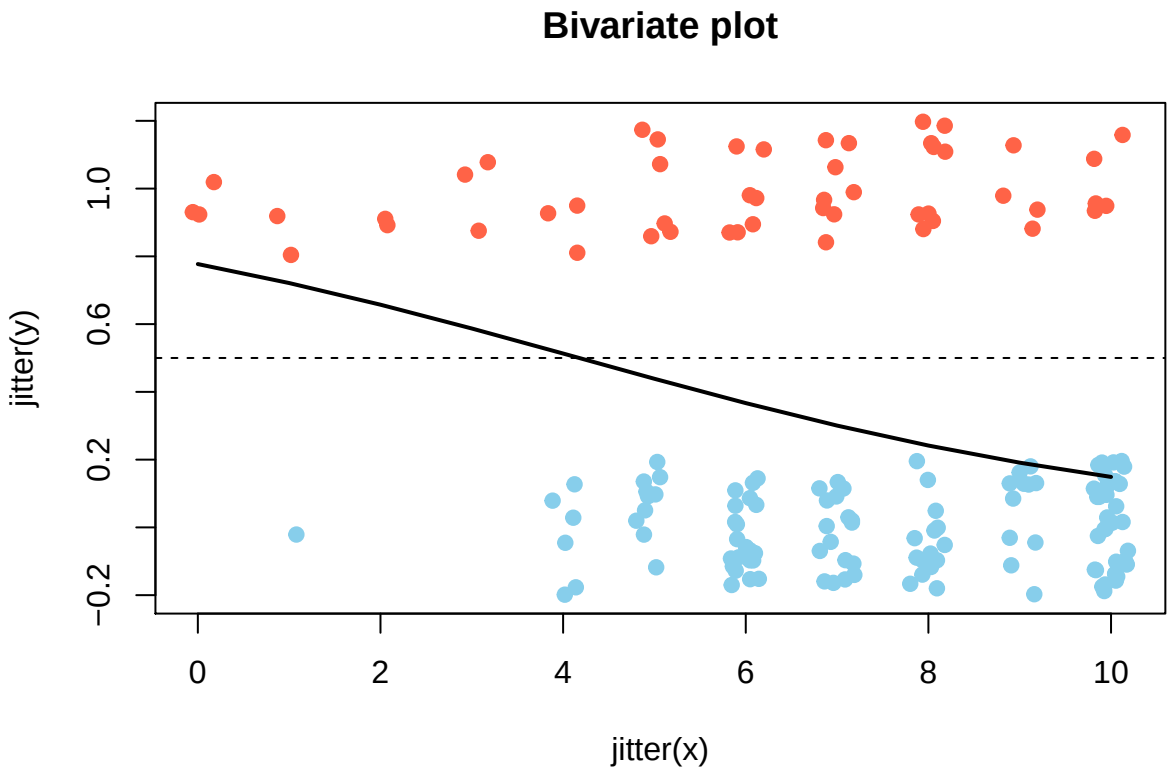
```
fit3 <- glm(dilemma ~ altruism1 + trust + negRecPooled, family = binomial)
summary(fit3)
```

```
##
## Call:
## glm(formula = dilemma ~ altruism1 + trust + negRecPooled, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5685  -0.8132  -0.5797   1.0164   2.1091
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.46652    0.77005   3.203  0.00136 **
```

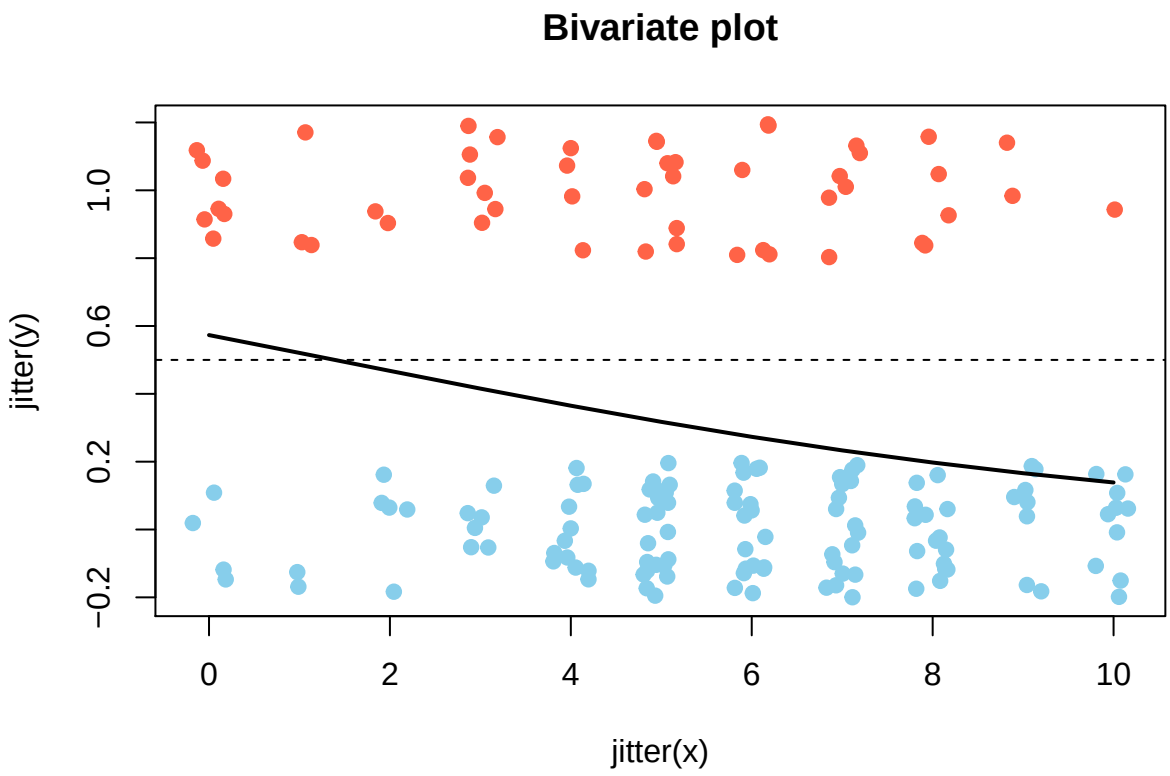
```
## altruism1    -0.27766    0.08449  -3.286  0.00102 **
## trust        -0.15968    0.07465  -2.139  0.03242 *
## negRecPooled -0.15207    0.08181  -1.859  0.06307 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 209.36  on 169  degrees of freedom
## Residual deviance: 184.56  on 166  degrees of freedom
## AIC: 192.56
##
## Number of Fisher Scoring iterations: 4
```

Next, the bivariate plots are inspected all show a negative relationship with dilemma. The lower the scores the more chance on defecting and vice versa. The strongest relation seem to be with altruism1, followed by trust and negRecPooled. These plots are Bivariate plots, so the true relationships in the model are different and a more complex combination of these.

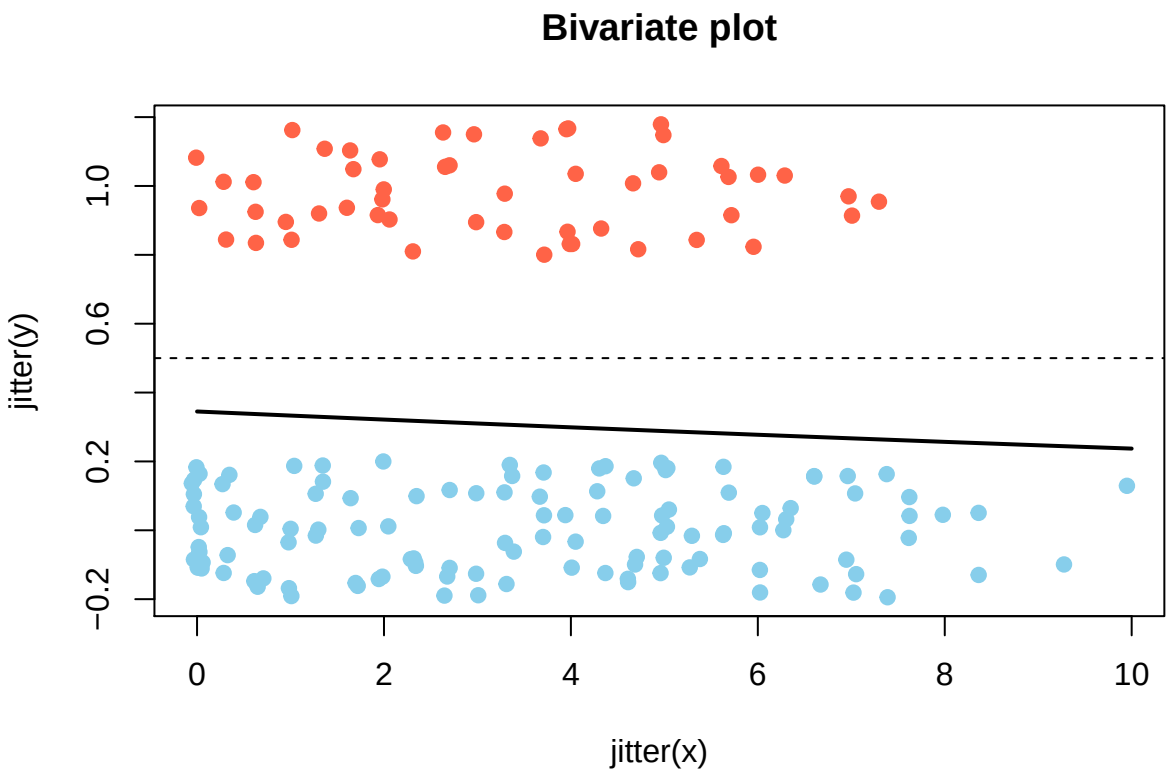
```
plotBivariate(altruism1, dilemmaBinary)
```




```
plotBivariate(trust, dilemmaBinary)
```

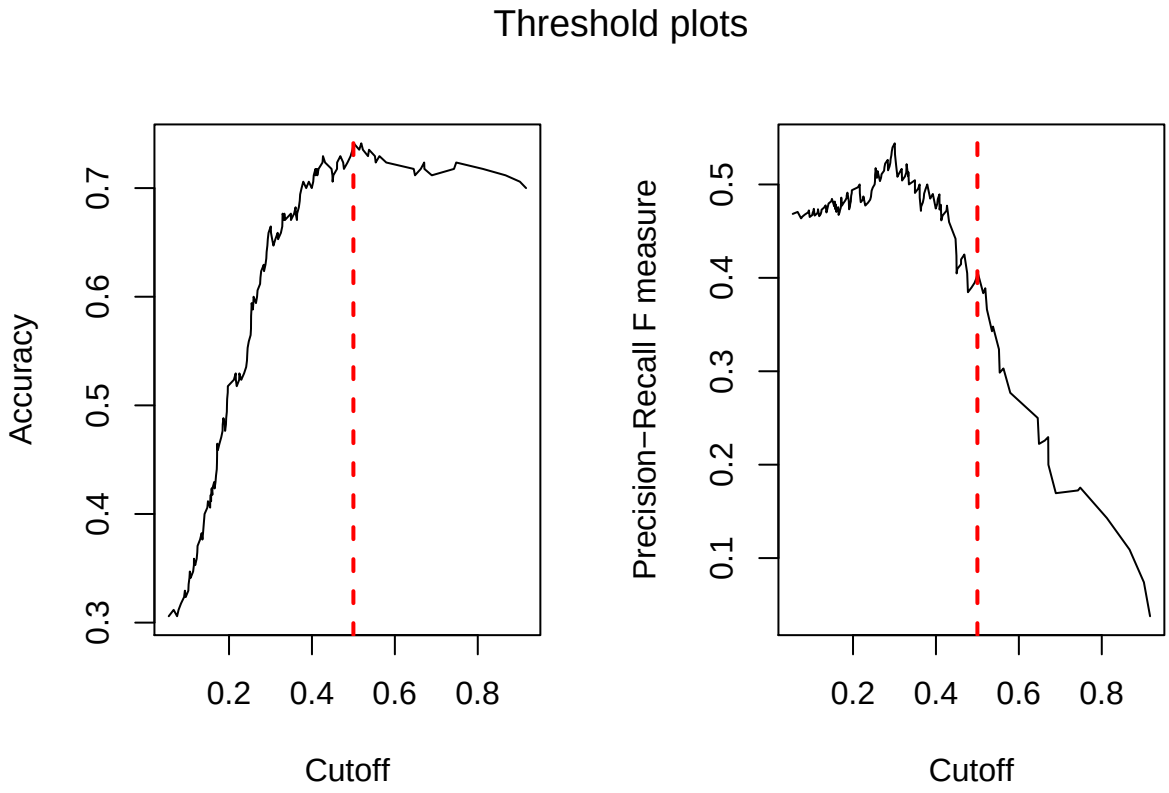



```
plotBivariate(negRecPooled, dilemmaBinary)
```

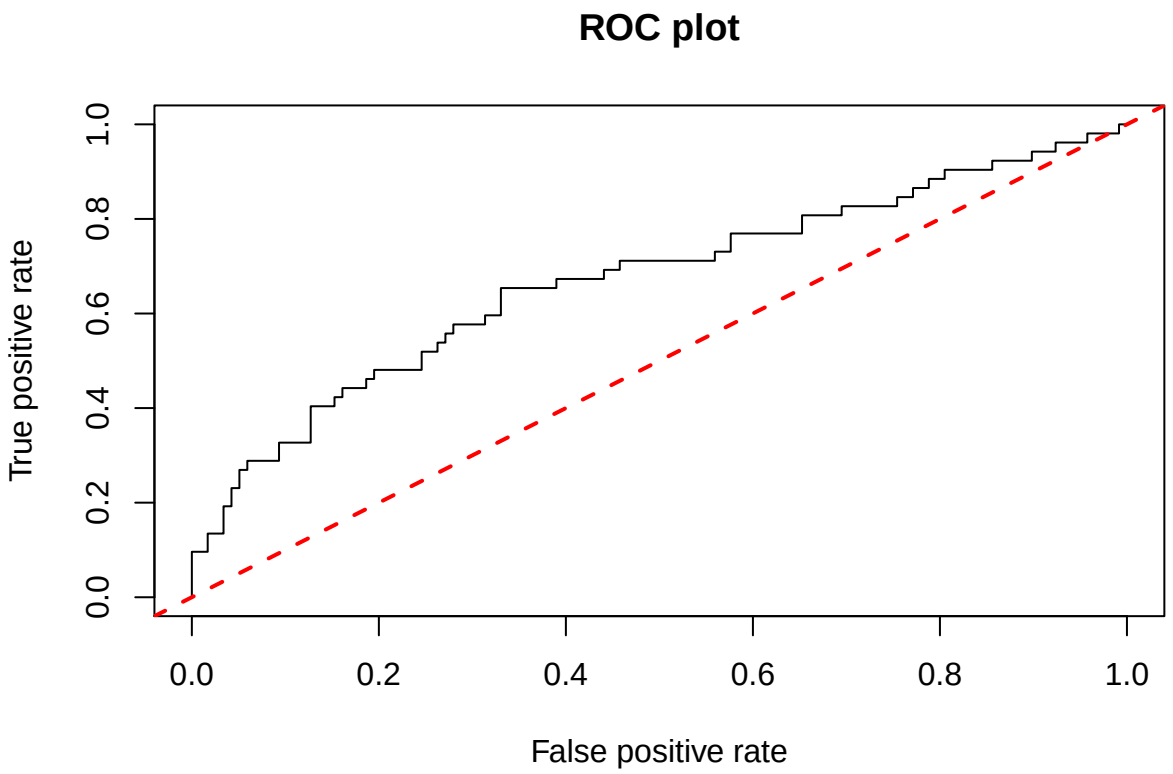


plotThreshold shows that the threshold of 0.50 (50 %) corresponds with the maximum accuracy score, on average for all folds. However the F-score is suboptimal, so if a precision, recall or combination might be preferred, a lower threshold would improve performance. Furthermore, the ROC plot shows that the AUC is 0.67 poor to fair, but scores better than the random line of TP-rate and FP-rate of 0.5.

```
plotThreshold(trainDilemma, foldid, mdl3)
```




```
plotThreshold(trainDilemma, foldid, md13, "ROC")
```




```
## [1] 0.6706323
```

Lasso

The lasso model is performed, using a logistic regression. The best performing model has a lambda of 0.21 and a corresponding accuracy of 0.73 (73%) and a standard deviation of 0.04 (4%). The accuracy is just slightly worse than best subset selection method, but the standard deviation of 0.04 is quite lower and has thus a lot more stable results. As the plot shows, the optimal lambda includes four variables, however the lambda plus 1 standard error corresponds with the base model, this shows that the difference isn't very substantial.

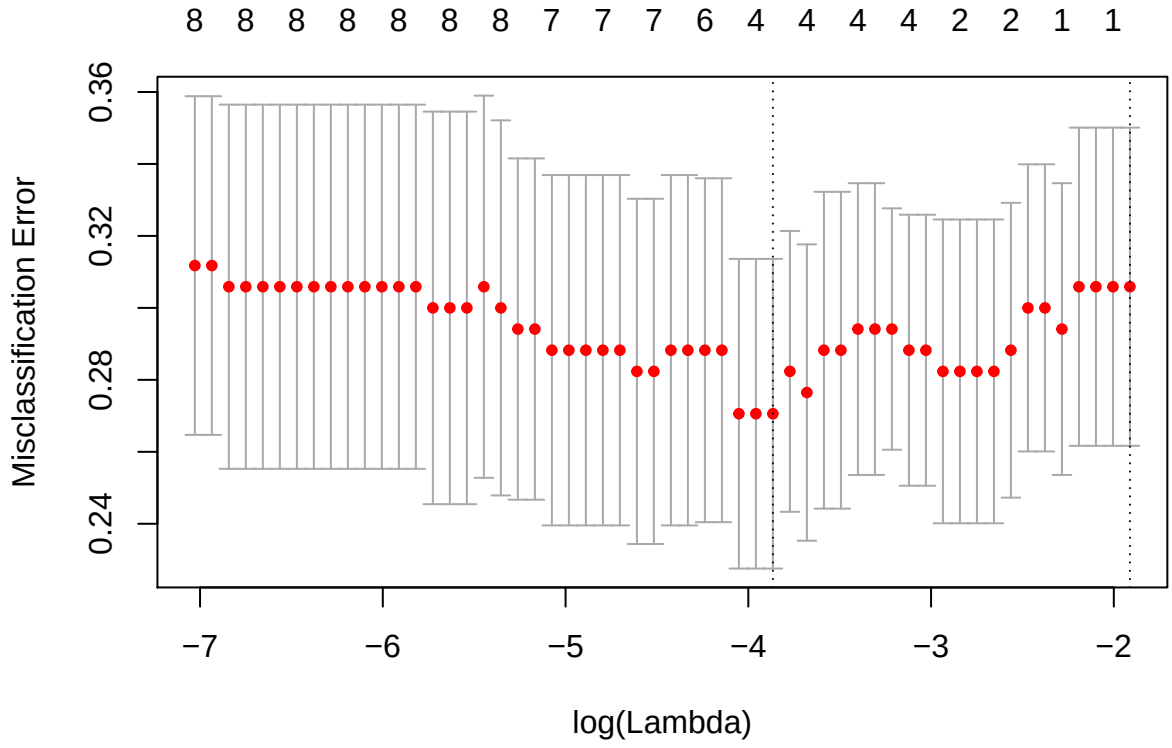
```
lassocv <- cv.glmnet(trainPreferences, dilemma, alpha = 1, foldid = foldid,
  family = "binomial", type.measure = "class")
resultsLasso(lassocv, measure = "acc")
```

```
## $glmnet.fit
##
## Call:  glmnet(x = trainPreferences, y = dilemma, alpha = 1, family = "binomial")
##
##       Df       %Dev    Lambda
##  [1,]  0 -9.015e-16 0.1478000
##  [2,]  1  1.408e-02 0.1347000
##  [3,]  1  2.562e-02 0.1227000
##  [4,]  1  3.514e-02 0.1118000
##  [5,]  1  4.304e-02 0.1019000
##  [6,]  2  5.057e-02 0.0928400
```

```
##  [7,]  2  5.863e-02 0.0845900
##  [8,]  2  6.539e-02 0.0770800
##  [9,]  2  7.107e-02 0.0702300
## [10,]  2  7.585e-02 0.0639900
## [11,]  2  7.988e-02 0.0583100
## [12,]  2  8.327e-02 0.0531300
## [13,]  3  8.633e-02 0.0484100
## [14,]  4  9.245e-02 0.0441100
## [15,]  4  9.783e-02 0.0401900
## [16,]  4  1.024e-01 0.0366200
## [17,]  4  1.062e-01 0.0333700
## [18,]  4  1.095e-01 0.0304000
## [19,]  4  1.123e-01 0.0277000
## [20,]  4  1.146e-01 0.0252400
## [21,]  4  1.166e-01 0.0230000
## [22,]  4  1.182e-01 0.0209500
## [23,]  4  1.196e-01 0.0190900
## [24,]  4  1.208e-01 0.0174000
## [25,]  5  1.219e-01 0.0158500
## [26,]  6  1.228e-01 0.0144400
## [27,]  6  1.237e-01 0.0131600
## [28,]  6  1.245e-01 0.0119900
## [29,]  7  1.253e-01 0.0109300
## [30,]  7  1.261e-01 0.0099550
## [31,]  7  1.267e-01 0.0090710
## [32,]  7  1.272e-01 0.0082650
## [33,]  7  1.277e-01 0.0075310
## [34,]  7  1.281e-01 0.0068620
## [35,]  7  1.284e-01 0.0062520
## [36,]  7  1.286e-01 0.0056970
## [37,]  7  1.288e-01 0.0051910
## [38,]  8  1.291e-01 0.0047300
## [39,]  8  1.293e-01 0.0043090
## [40,]  8  1.295e-01 0.0039270
## [41,]  8  1.296e-01 0.0035780
## [42,]  8  1.297e-01 0.0032600
## [43,]  8  1.298e-01 0.0029700
## [44,]  8  1.299e-01 0.0027060
## [45,]  8  1.300e-01 0.0024660
## [46,]  8  1.301e-01 0.0022470
## [47,]  8  1.301e-01 0.0020470
## [48,]  8  1.302e-01 0.0018650
## [49,]  8  1.302e-01 0.0017000
## [50,]  8  1.302e-01 0.0015490
## [51,]  8  1.302e-01 0.0014110
## [52,]  8  1.303e-01 0.0012860
## [53,]  8  1.303e-01 0.0011720
## [54,]  8  1.303e-01 0.0010670
## [55,]  8  1.303e-01 0.0009726
## [56,]  8  1.303e-01 0.0008862
##
## $bestlambda
## [1] 0.02095466
##
```

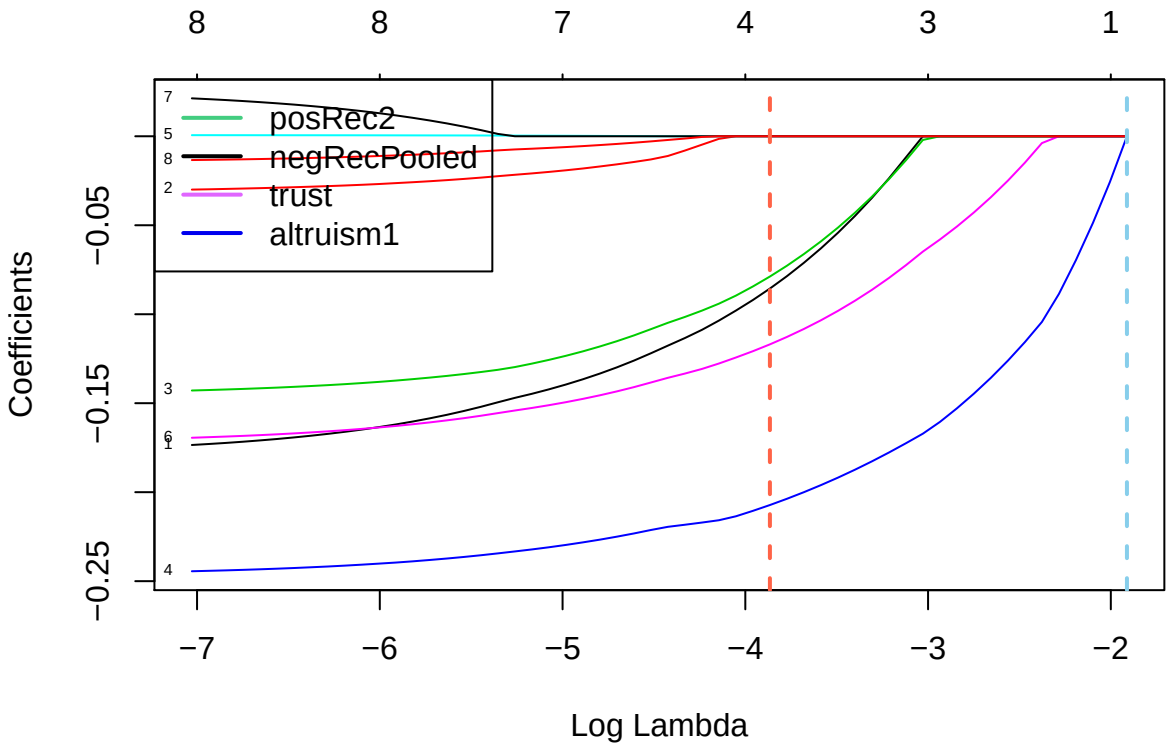
```
## $bestnmean
## [1] 0.7294118
##
## $bestnsd
## [1] 0.0430257
##
## $lambda1se
## [1] 0.147831
```

```
accLasso <- 1 - min(lassocv$cvm)
plot(lassocv)
```



The shrinkage plot shows the shrank coefficients for all variables. As the plot shows the best lambda includes altruism1, trust, negRecPooled and posRec2. All variables have negative coefficients. The first three variables were also found using the best subset selection method, but now posRec2 is included, for this variable a different distribution was found in the bivariate analysis. The Log coefficients that correspond with the optimal penalty factor can be found underneath.

```
plotShrinkage(trainPreferences, dilemma, lassocv)
legend('topleft', legend = c("posRec2", "negRecPooled", "trust", "altruism1"),
  lwd = 2, col= c("seagreen3", "black", "mediumorchid1", "blue2"))
```

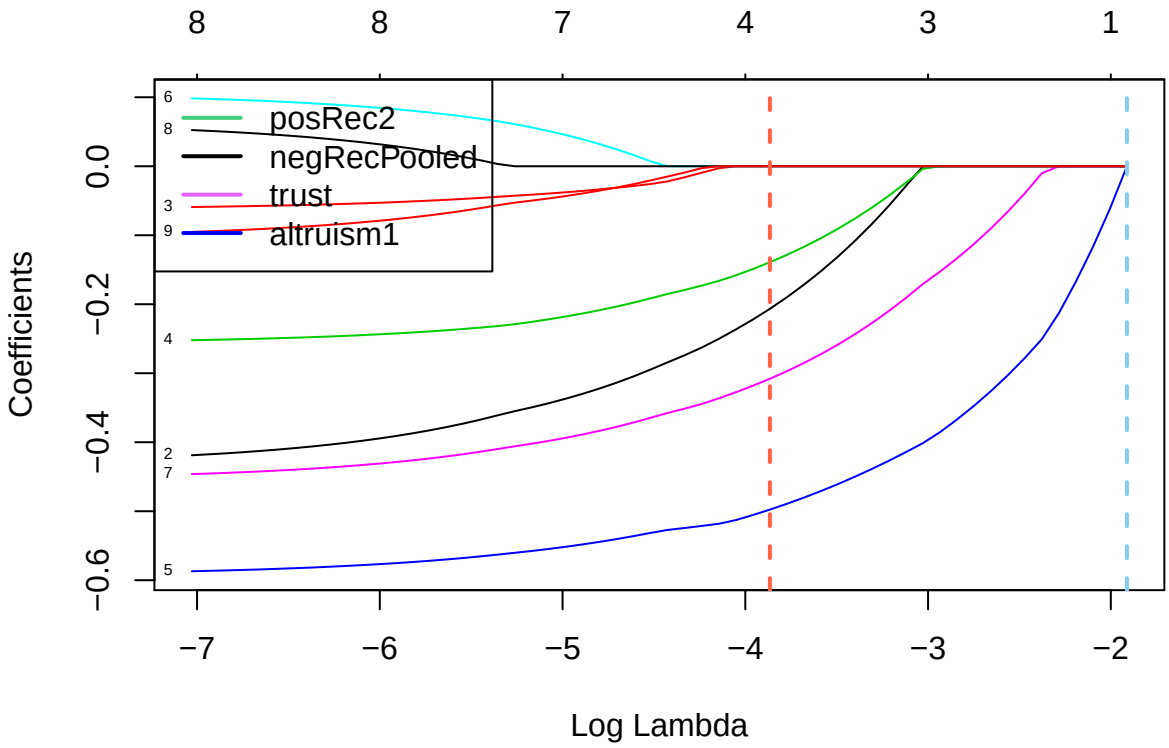



```
predict(lassocv, type = "coefficients", s = lassocv$lambda.min)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)   1.77392685
## negRecPooled -0.08556851
## posRec1       .
## posRec2      -0.07881334
## altruism1    -0.20721202
## altruism2     .
## trust        -0.11693435
## risk1         .
## risk2         .
```

This next shrinkage plot shows the same coefficients but then on a scaled dataset. So the coefficients can be compared. As can be seen, altruism1 and trust have the highest coefficients, followed by negRecPooled and posRec2.

```
plotShrinkage(trainPreferencesScaled, dilemma, lassocv)
legend('topleft', legend = c("posRec2", "negRecPooled", "trust", "altruism1"),
  lwd = 2, col= c("seagreen3", "black", "mediumorchid1", "blue2"))
```



Boosting

The final model is boosting. First the parameters are set to train the model.

```
fitControl <- trainControl(method = "cv", number = 5)
gbmGrid <-  expand.grid(interaction.depth = c(1, 2, 4, 8), n.trees = (1:10)*600, n.minobsinnode = 20, sh
```

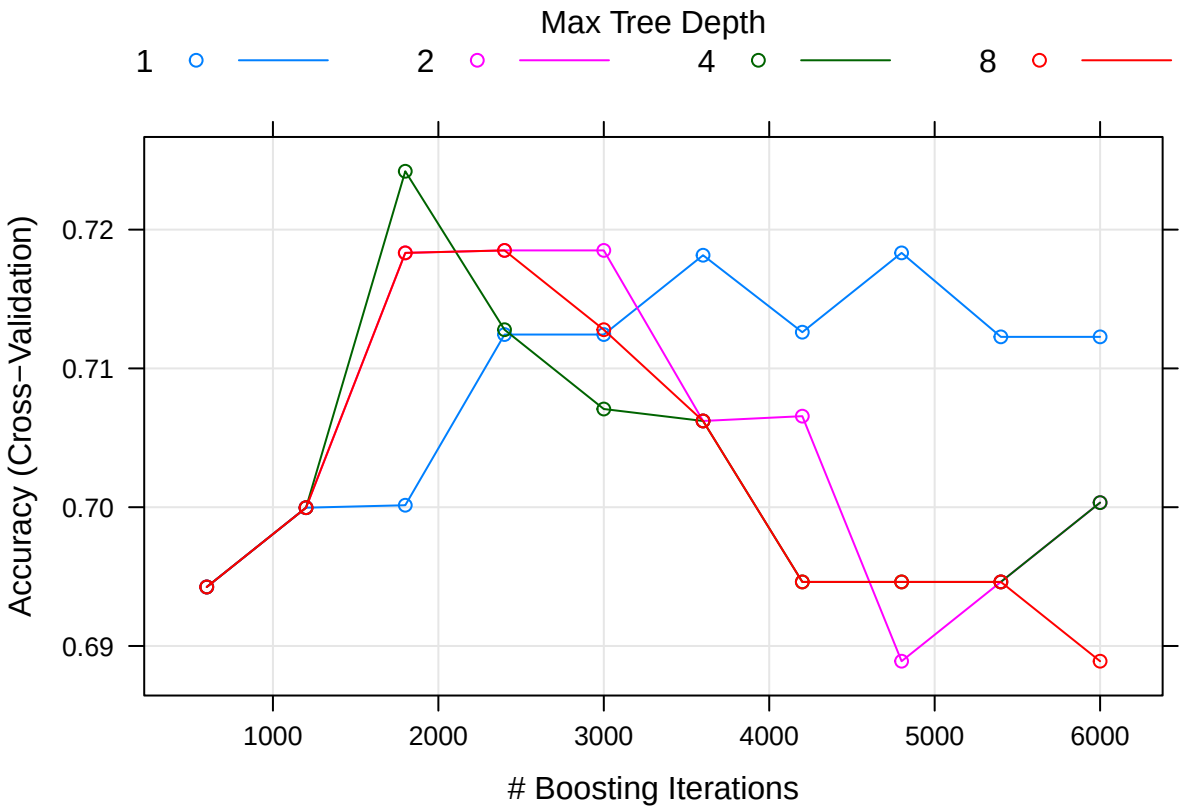
After training an accuracy of 0.724 is found, which is worse than the previous models. The optimal settings are ntrees of 1800, interaction.depth of 4, n.minobsinnode of 20 and shrinkage factor of 0.001.

```
set.seed(64)
gbmFit <- train(dilemma ~ ., data = trainDilemma, method = "gbm", verbose = FALSE, trControl = fitContro
gbmFit
```

```
## Stochastic Gradient Boosting
##
## 170 samples
##   8 predictor
##   2 classes: 'cooperate', 'defect'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 137, 136, 135, 135, 137
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
```

```
##   1                   600     0.6942501  0.00000000
##   1                  1200     0.6999643  0.02412060
##   1                  1800     0.7001426  0.03964333
##   1                  2400     0.7124421  0.11620261
##   1                  3000     0.7124421  0.12698767
##   1                  3600     0.7181564  0.17038613
##   1                  4200     0.7126101  0.17407993
##   1                  4800     0.7183244  0.19599176
##   1                  5400     0.7122740  0.19683934
##   1                  6000     0.7122740  0.19683934
##   2                   600     0.6942501  0.00000000
##   2                  1200     0.6999643  0.02412060
##   2                  1800     0.7183244  0.11619195
##   2                  2400     0.7185027  0.15161039
##   2                  3000     0.7185027  0.17193423
##   2                  3600     0.7062134  0.16887563
##   2                  4200     0.7065597  0.19489633
##   2                  4800     0.6889025  0.16365404
##   2                  5400     0.6946168  0.18314048
##   2                  6000     0.7003310  0.20214686
##   4                   600     0.6942501  0.00000000
##   4                  1200     0.6999643  0.02412060
##   4                  1800     0.7242068  0.12749139
##   4                  2400     0.7127884  0.12748979
##   4                  3000     0.7070741  0.14052314
##   4                  3600     0.7062134  0.18467949
##   4                  4200     0.6946168  0.16500980
##   4                  4800     0.6946168  0.17433235
##   4                  5400     0.6946168  0.18314048
##   4                  6000     0.7003310  0.20214686
##   8                   600     0.6942501  0.00000000
##   8                  1200     0.6999643  0.02412060
##   8                  1800     0.7183244  0.11619195
##   8                  2400     0.7185027  0.15161039
##   8                  3000     0.7127884  0.16116896
##   8                  3600     0.7062134  0.18467949
##   8                  4200     0.6946168  0.17381794
##   8                  4800     0.6946168  0.17381794
##   8                  5400     0.6946168  0.18314048
##   8                  6000     0.6889025  0.17215077
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.001
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 20
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 1800,
##  interaction.depth = 4, shrinkage = 0.001 and n.minobsinnode = 20.
```

```
plot(gbmFit)
```



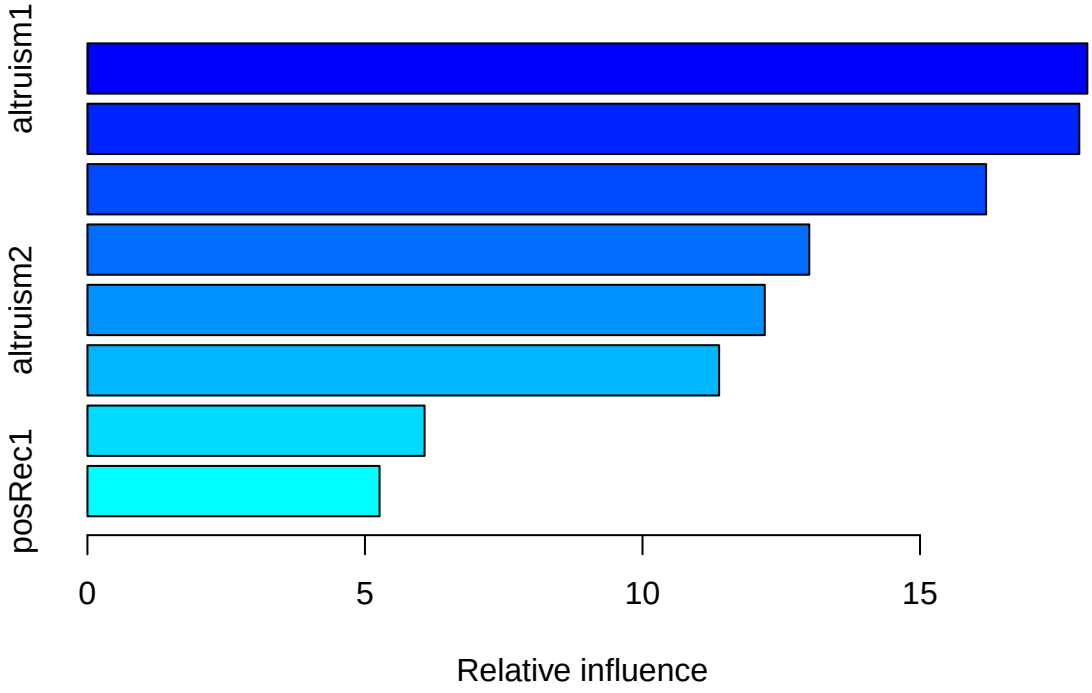

```
gbmFit$results[which.max(gbmFit$results$Accuracy),1:6]
```

```
##    shrinkage interaction.depth n.minobsinnode n.trees  Accuracy     Kappa
## 23     0.001                 4             20    1800 0.7242068 0.1274914
```

```
accBoost <- max(gbmFit$results$Accuracy)
```

The most important variable is altruism1 with 18%, followed by negRecPooled with 18% and risk2 with 16%. Followed by trust, altruism2 and posRec2 with respectively 13, 12 and 11%. Some of these effects might be caused due to interaction effects or confounding variables. The first two correspond with the best subset selection method, and trust with the fourth. There is no one strong influencer.

```
set.seed(32)
gbmBestFit <- gbm(dilemmaBinary ~ . -dilemma, data = trainDilemma, n.trees = 1800, interaction.depth = 4
summary(gbmBestFit)
```




```
##                       var   rel.inf
## altruism1       altruism1 18.014569
## negRecPooled negRecPooled 17.869321
## risk2               risk2 16.190349
## trust               trust 13.005004
## altruism2       altruism2 12.202622
## posRec2           posRec2 11.380609
## risk1               risk1  6.074279
## posRec1           posRec1  5.263247
```

Final models

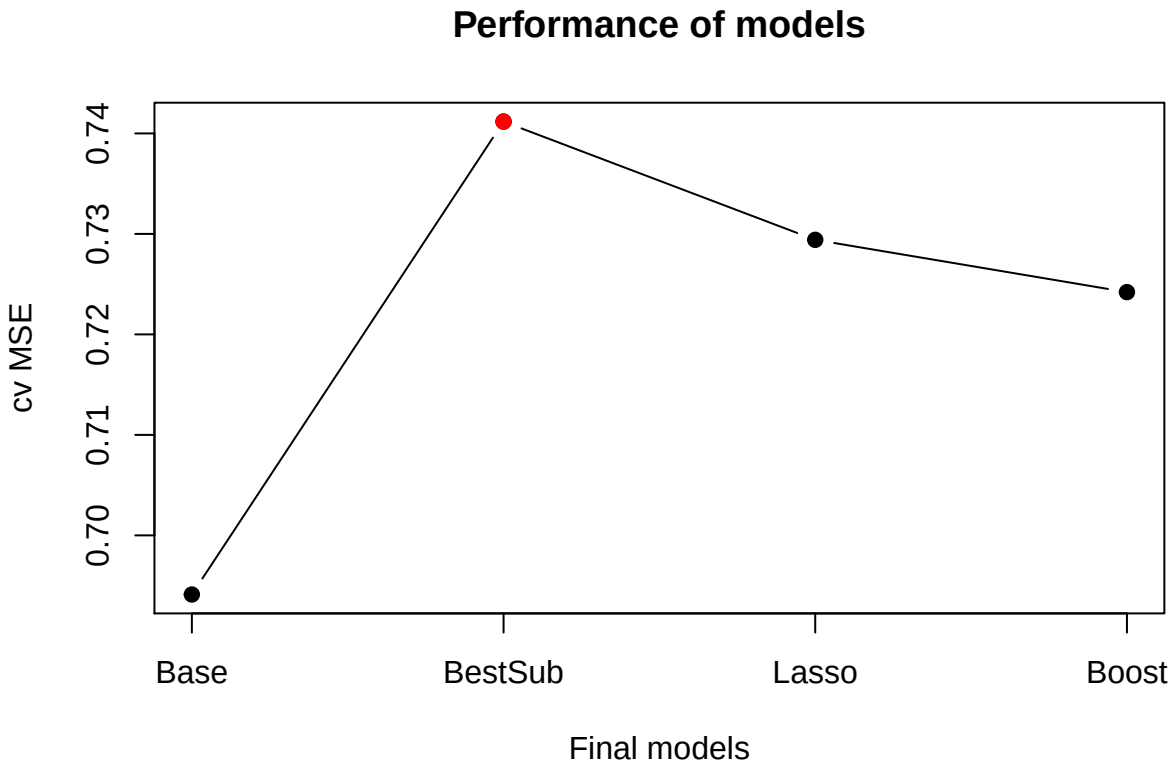
The final models are inspected. As can be seen the best performing model is the best subset selection method. The best subset selection improves with almost 7% compared to the base model. The Lasso method performs just slightly less, and improves with 5% compared with the base model, but the standard deviation decreases quite strongly from 0.12 (12%) in the best subset selection method to 0.04 (4%). Therefore, a lasso model is a lot more stable. The more complex boosting model does not perform better than lasso. Because of the strong decrease in standard deviation and more stable results, the lasso model is preferred.

```
finalModels <- data.frame(acc = c(accBase, accBestSub, accLasso, accBoost), row.names = c(c("Base", "Bes
finalModels
```

```
##               acc
## Base    0.6941176
## BestSub 0.7411765
## Lasso   0.7294118
```

```
## Boost   0.7242068
```

```
plotModels(finalModels$acc, xlab = "Final models", axislabels = rownames(finalModels), classification =
```




```
## [1] 0.6941176 0.7411765 0.7294118 0.7242068
```

Final predictions

Now the final predictions on test set is performed. The model is trained on all training data. The base model has an accuracy of 0.58 (58%) on the test set. The lasso model performs better and has an accuracy of 0.64 (64%). On the test set the final models performs 10% better compared to the base model and 6% in absolute percentage points.

```
ytrue <- qualtrics[test, "dilemma"]

majorityIndex <- which.max(table(trainDilemma$dilemma))
majorityClass <- levels(trainDilemma$dilemma)[majorityIndex]
sum(ytrue == majorityClass)/length(ytrue)
```

```
## [1] 0.5882353
```

```
finalFit <- glmnet(trainPreferences, dilemma, alpha = 1, family = "binomial", lambda =  lassocv$lambda.
yhatProb <- predict(finalFit, s = lassocv$lambda.min,  newx = preferences[test,], type = "response")
yhatClass <- ifelse(yhatProb < 0.5, "cooperate", "defect")
sum(ytrue == yhatClass)/length(ytrue)
```

```
## [1] 0.6470588
```

Final model inspection

As mentioned before, the variables for the best model are altruism1, trust and negRecPooled. The Lasso model explains 10% of the deviance and corresponds with 3 degrees of freedom, unlike as in the trainings model it does not contain posRec2, and only the three variables altruism1, trust and negRecPooled. Wherein, altruism1 and trust have almost equal coefficients with a Log coefficients of 0.164 and 0.163 respectively. negRecPooled has a Log coefficients of 0.038 and is almost 4 times smaller than the other variables.

```
qualtricsDilemma <- select(qualtrics, -c(public, offer, respond, chicken))
attach(qualtricsDilemma)

finalFit <- glmnet(preferences, dilemma, alpha = 1, family = "binomial", lambda = lassocv$lambda.min)
finalFit
```

```
##
## Call:  glmnet(x = preferences, y = dilemma, family = "binomial", alpha = 1,      lambda = lassocv$lam
##
##      Df   %Dev  Lambda
## [1,]  3 0.1001 0.02095
```

```
coef(finalFit)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)   1.37614225
## negRecPooled -0.03828614
## posRec1       .
## posRec2       .
## altruism1    -0.16408354
## altruism2     .
## trust        -0.16289182
## risk1         .
## risk2         .
```

Chicken game

First a subset of trainQualtrics is taken with only the chicken game as dependent variable and the preferences, all the other games are excluded. Furthermore a binary variable is made, where 0 corresponds with chicken and 1 with dare. The models are again, classifcation models.

```
trainChicken <- select(trainQualtrics, -c(dilemma, offer, respond, public))
attach(trainChicken)
chickenBinary <- ifelse(chicken == "chicken", 0, 1)
```

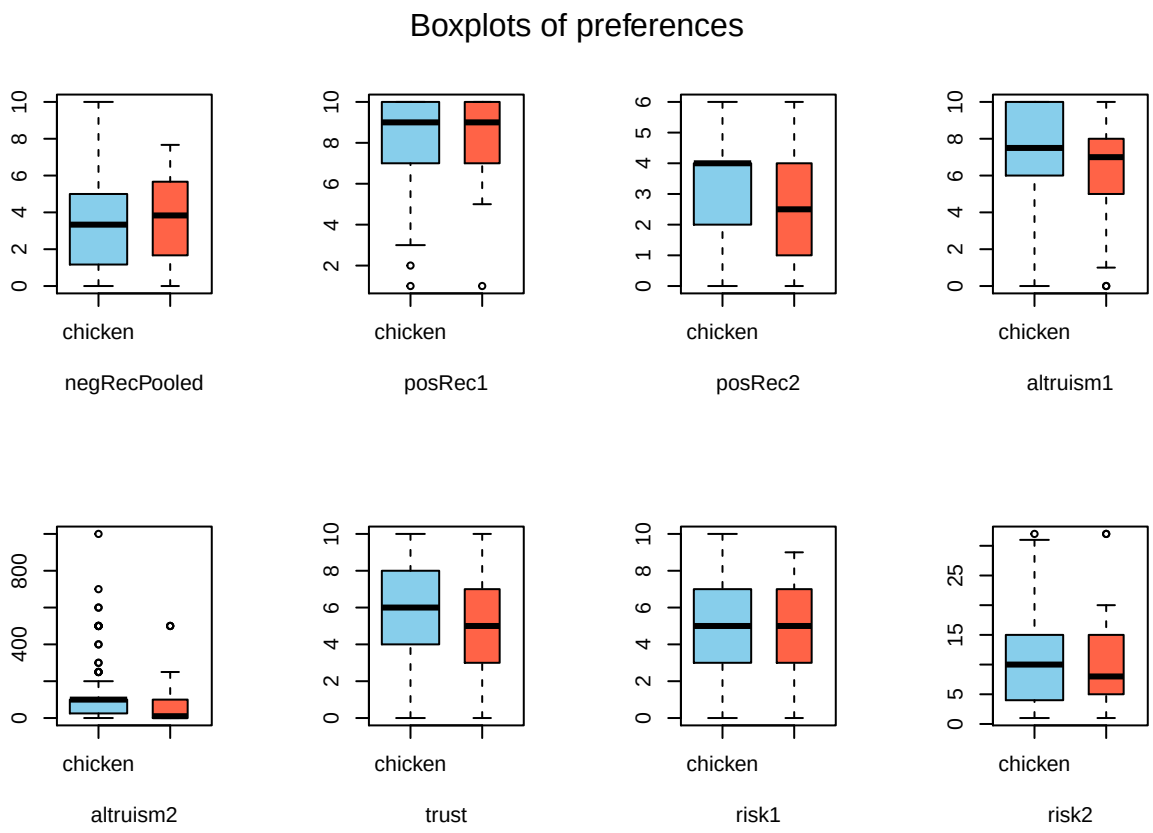
Bivariate analysis

The distributions are inspected via boxplots. The following main differences in distribution are found.

- A positive relationship with negRecPooled
- A negative relationship with posRec2
- A negative relationship with altruism1
- A negative relationship with altruism2
- A negative relationship with trust

Where a negative relationship corresponds with a distribution more tended to the lower spectrum for that preference for the defecting group, or the binary 1 compared to the cooperate group. Most of these differences correspond with prisoner's dilemma except for the somewhat small difference found in negRecPooled, and the small difference of distribution found posRec1 found in prisoner's dilemma, but not in the chicken game. The games do have a similar character, except of the more risky nature of the chicken game, however there was no difference found between the chicken and dare groups for risk.

```
plotBoxes(trainQualtrics, chicken)
```



Base model

A base model is conducted using the basemodelcv function. The following accuracies are found, a mean cross-validation accuracy of 0.73 (73%) and a standard deviation of 0.025 (2.5%).

```
baseline <- baseModelcv(trainChicken, foldid)
baseline
```

```
## $cv
## [1] 0.7058824 0.7352941 0.7058824 0.7352941 0.7647059
##
## $meancv
## [1] 0.7294118
##
## $sdcv
## [1] 0.02460765
```

```
accBase <- baseline$meancv
```

Best subset selection

A best subset selection method is performed on the all trainings data and then cross-validation is used to chose the best model for every number of variables, this method uses a logistic regression. The best model contains no variables and is thus the same as the base model.

```
set.seed(32)
bestChicken <- bestglm(data.frame(trainPreferences, chickenBinary), IC = "CV", CVArgs = list(Method="HT
```

```
## Morgan-Tatar search since family is non-gaussian.
```

```
bestChicken$Subsets
```

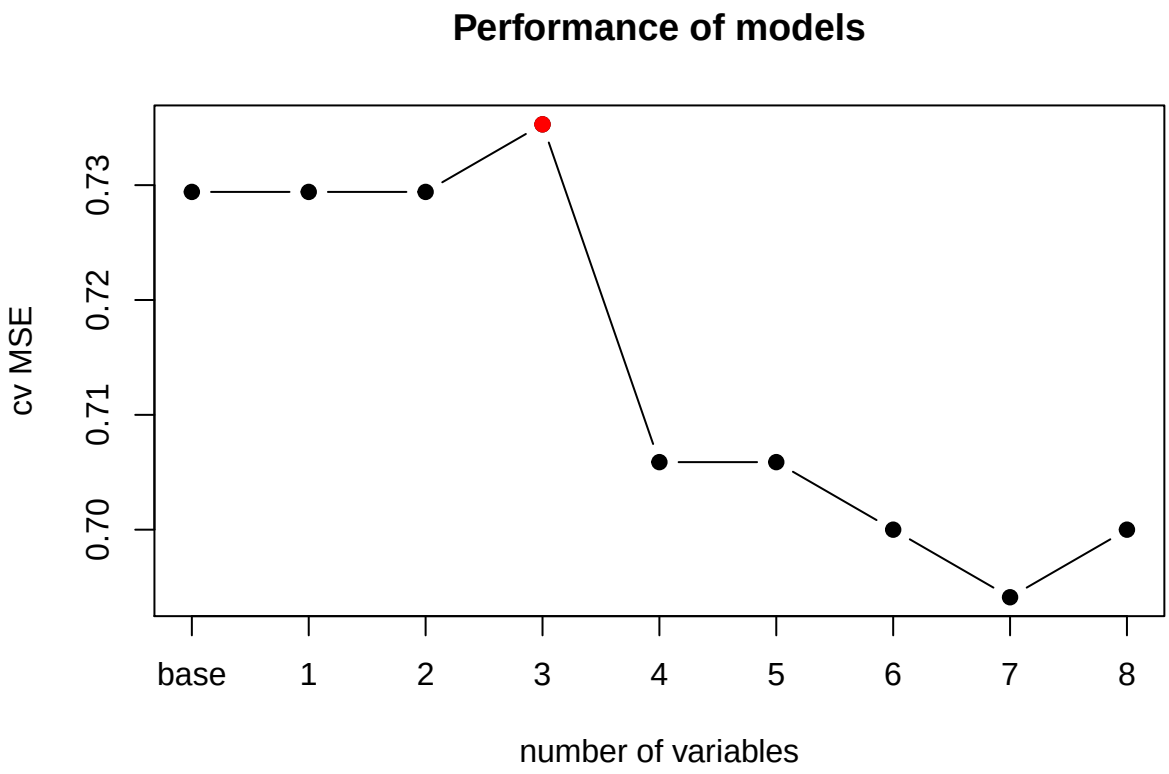
```
##    Intercept negRecPooled posRec1 posRec2 altruism1 altruism2 trust risk1
## 0*      TRUE        FALSE   FALSE   FALSE     FALSE     FALSE FALSE FALSE
## 1       TRUE        FALSE   FALSE   FALSE     FALSE      TRUE FALSE FALSE
## 2       TRUE        FALSE   FALSE   FALSE     FALSE      TRUE  TRUE FALSE
## 3       TRUE        FALSE   FALSE   FALSE     FALSE      TRUE  TRUE  TRUE
## 4       TRUE        FALSE    TRUE   FALSE      TRUE      TRUE FALSE  TRUE
## 5       TRUE        FALSE    TRUE   FALSE      TRUE      TRUE  TRUE  TRUE
## 6       TRUE        FALSE    TRUE    TRUE      TRUE      TRUE  TRUE  TRUE
## 7       TRUE        FALSE    TRUE    TRUE      TRUE      TRUE  TRUE  TRUE
## 8       TRUE         TRUE    TRUE    TRUE      TRUE      TRUE  TRUE  TRUE
##    risk2 logLikelihood        CV       sdCV
## 0* FALSE     -99.25332 0.2028979 0.02448929
## 1  FALSE     -96.92668 0.1962126 0.01682185
## 2  FALSE     -96.01381 0.2002795 0.02011046
## 3  FALSE     -94.81301 0.2019947 0.01580632
## 4  FALSE     -94.45253 0.2032336 0.01504482
## 5  FALSE     -93.79328 0.1942290 0.01439031
## 6  FALSE     -93.47261 0.1976622 0.01589942
## 7   TRUE     -93.34551 0.2118603 0.02708876
## 8   TRUE     -93.28897 0.2034294 0.01772673
```

Next, since these cross-validation scores use different folds and are not comparable with the other models, the found best models are used as input for bestsubsetcv function. The models are manually constructed. Using these foldid's, a different optimal model is found, the best performing model has three variables, altruism2, trust and risk1 The first two corresponds with the findings of the bivariate analysis, the latter not. These variables correspond with a accuracy of 0.735 (73.5%) and a standard deviation of 0.02 (2%).This is just a very tiny improvement compared to the base model.

```
mdl1 <- chicken ~ altruism2
mdl2 <- chicken ~ altruism2 + trust
mdl3 <- chicken ~ altruism2 + trust + risk1
mdl4 <- chicken ~ altruism2 + risk1 + altruism1 + posRec1
mdl5 <- chicken ~ altruism2 + risk1 + altruism1 + posRec1 + trust
mdl6 <- chicken ~ altruism2 + risk1 + altruism1 + posRec1 + trust + posRec2
mdl7 <- chicken ~ altruism2 + risk1 + altruism1 + posRec1 + trust + posRec2 + risk2
mdl8 <- chicken ~ .
models <- c(mdl1, mdl2, mdl3, mdl4, mdl5, mdl6, mdl7, mdl8)

bestSubs <- bestSubsetcv(trainChicken, foldid, models)
bestSubs
```

```
## $cv
##              1         2         3         4         5         6         7
## [1,] 0.7058824 0.7058824 0.7352941 0.7058824 0.7352941 0.7352941 0.7058824
## [2,] 0.7352941 0.7352941 0.7352941 0.7058824 0.7058824 0.6764706 0.6764706
## [3,] 0.7058824 0.7058824 0.7058824 0.6764706 0.6764706 0.6764706 0.6470588
## [4,] 0.7352941 0.7352941 0.7352941 0.6764706 0.6764706 0.6764706 0.7058824
## [5,] 0.7647059 0.7647059 0.7647059 0.7647059 0.7352941 0.7352941 0.7352941
##              8
## [1,] 0.7058824
## [2,] 0.6764706
## [3,] 0.6764706
## [4,] 0.7058824
## [5,] 0.7352941
##
## $meancv
##         1         2         3         4         5         6         7
## 0.7294118 0.7294118 0.7352941 0.7058824 0.7058824 0.7000000 0.6941176
##         8
## 0.7000000
##
## $sdcv
##          1          2          3          4          5          6
## 0.02460765 0.02460765 0.02079726 0.03602191 0.02941176 0.03221897
##          7          8
## 0.03353457 0.02460765
##
## $bestnmdl
## [1] 3
##
## $bestnmean
## [1] 0.7352941
##
## $bestnsd
## [1] 0.02079726
```

```
accBestSub <- bestSubs$bestnmean

plotModels(bestSubs$meancv, baseline, classification = TRUE)
```




```
##      base         1         2         3         4         5         6
## 0.7294118 0.7294118 0.7294118 0.7352941 0.7058824 0.7058824 0.7000000
##         7         8
## 0.6941176 0.7000000
```

As mentioned before, the variables for the best model are altruism2, trust and risk2. The AIC of the final model is 197.63 The corresponding Log coefficients are -0.0026 for altruism2, -0.11 for trust and 0.12 for risk2, the first two have a negative relationships with chicken game, the latter a positive. Where the first two were also found in the bivariate analysis. Altruism2 is weakly significant (0.083), where trust (0.11) and negRecPooled (0.13) are not significant. Since the variable altruism2 is on a different scale, the standardized preferences are used to compare the coefficients, in this case altruism2 has Log coefficients of -0.41, trust of -0.29 and risk1 of 0.28.

```
fit3 <- glm(mdl3, trainDilemma, family = "binomial")
summary(fit3)
```

```
##
## Call:
## glm(formula = mdl3, family = "binomial", data = trainDilemma)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1679  -0.8172  -0.6703   1.2418   2.1920
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.709011   0.487862  -1.453   0.1461
```

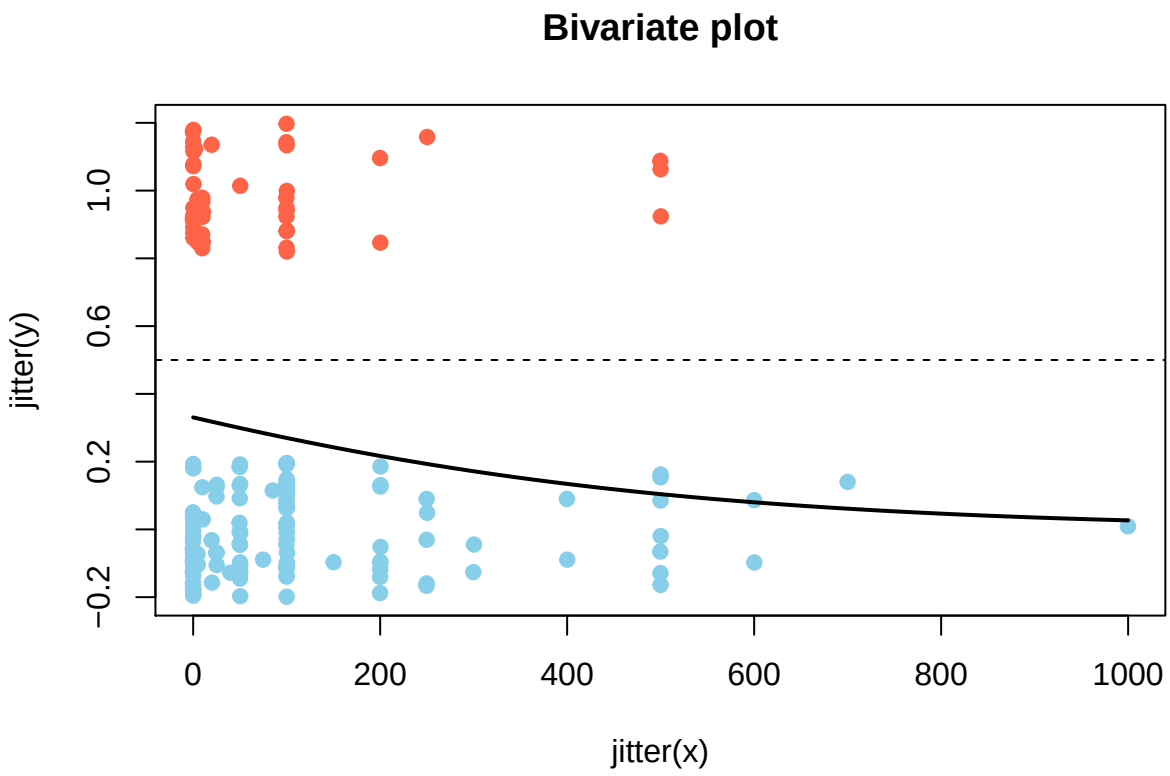
```
## altruism2   -0.002576   0.001484  -1.736   0.0826 .
## trust       -0.108949   0.069521  -1.567   0.1171
## risk1        0.114428   0.074838   1.529   0.1263
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 198.51  on 169  degrees of freedom
## Residual deviance: 189.63  on 166  degrees of freedom
## AIC: 197.63
##
## Number of Fisher Scoring iterations: 4
```

```
trainChickenScaled <- data.frame(chicken, trainPreferencesScaled)
scaledFit3 <- glm(mdl3, trainChickenScaled, family = "binomial")
coef(scaledFit3)
```

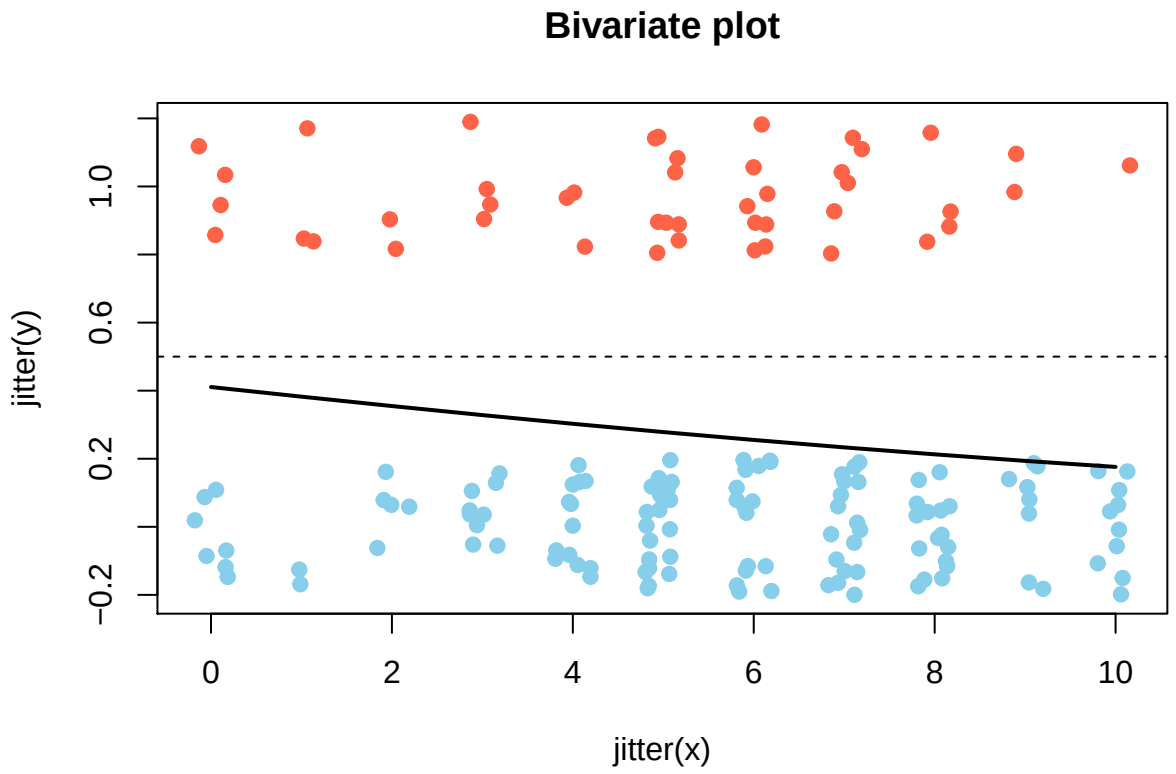
```
## (Intercept)   altruism2       trust       risk1
##  -1.0610630  -0.4133897  -0.2869665   0.2816014
```

Next, the bivariate plots are inspected. The first two show a negative relationship with dilemma. The lower the scores the more chance on defecting and vice versa. Risk1 shows a positive relationship, the higher the risk score, the higher the chance of daring. The strongest relation seem to be with altruism2. These plots are bivariate plots, so the true relationships in the model are different and a more complex combination of these.

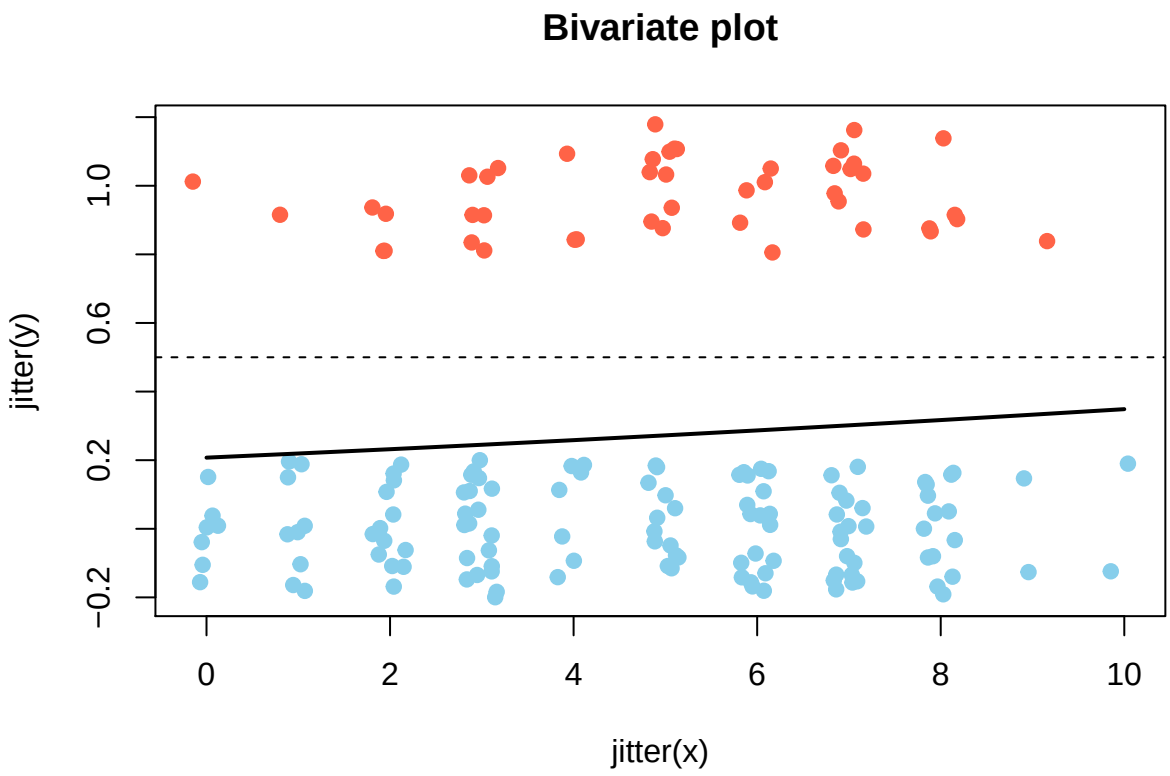
```
plotBivariate(altruism2, chickenBinary)
```



```
plotBivariate(trust, chickenBinary)
```

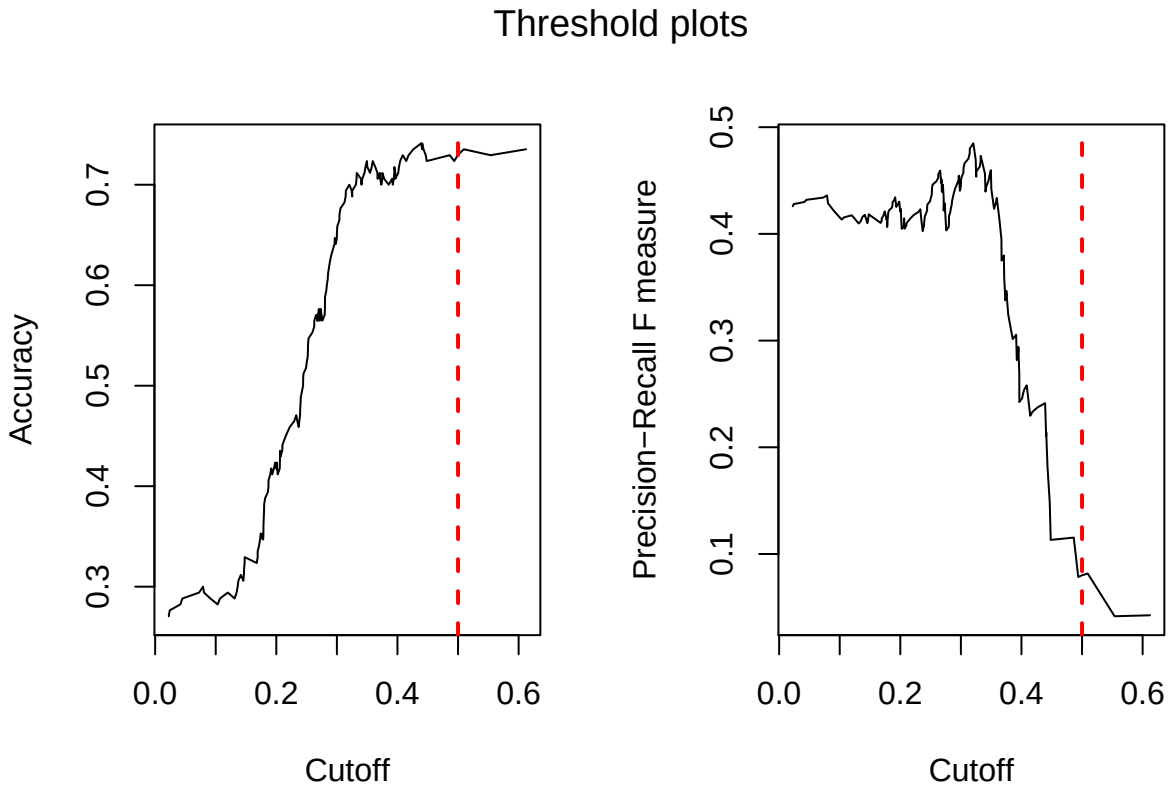



```
plotBivariate(risk1, chickenBinary)
```

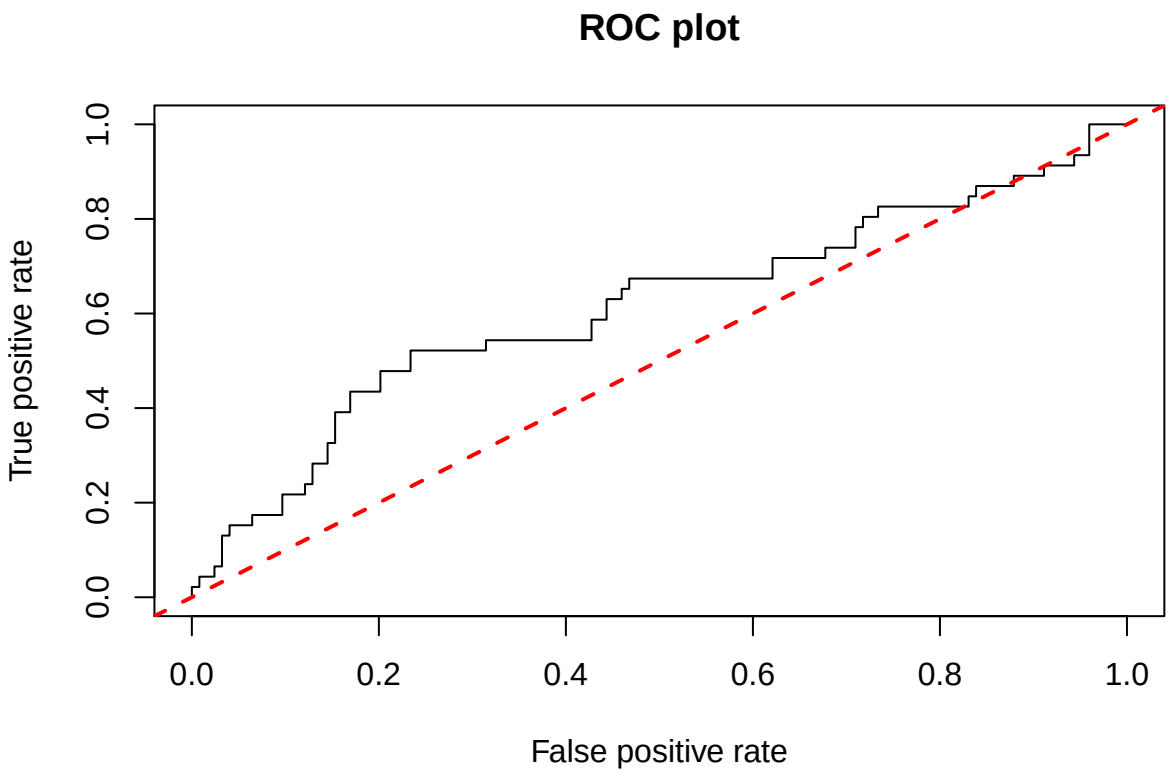


plotThreshold shows that the threshold of 0.50 (50 %) corresponds, more or less with the maximum accuracy score for all folds, however the F-score is very suboptimal, so if a precision, recall or combination might be preferred a lower threshold would improve performance. For instance, a cut-off at around 0.3 would decrease accuracy slightly but would be near optimal for the F-score, so this might be the preferred cut-off point if the measured goal was more than accuracy. Furthermore, the ROC plot shows that the AUC is 0.612, which is a poor result, but scores better than the random line of TP-rate and FP-rate of 0.5. So it seems that some patterns are learned considering precision or recall, but not so much in accuracy.

```
plotThreshold(trainChicken, foldid, mdl3)
```




```
plotThreshold(trainChicken, foldid, mdl3, "ROC")
```




```
## [1] 0.6125526
```

Lasso

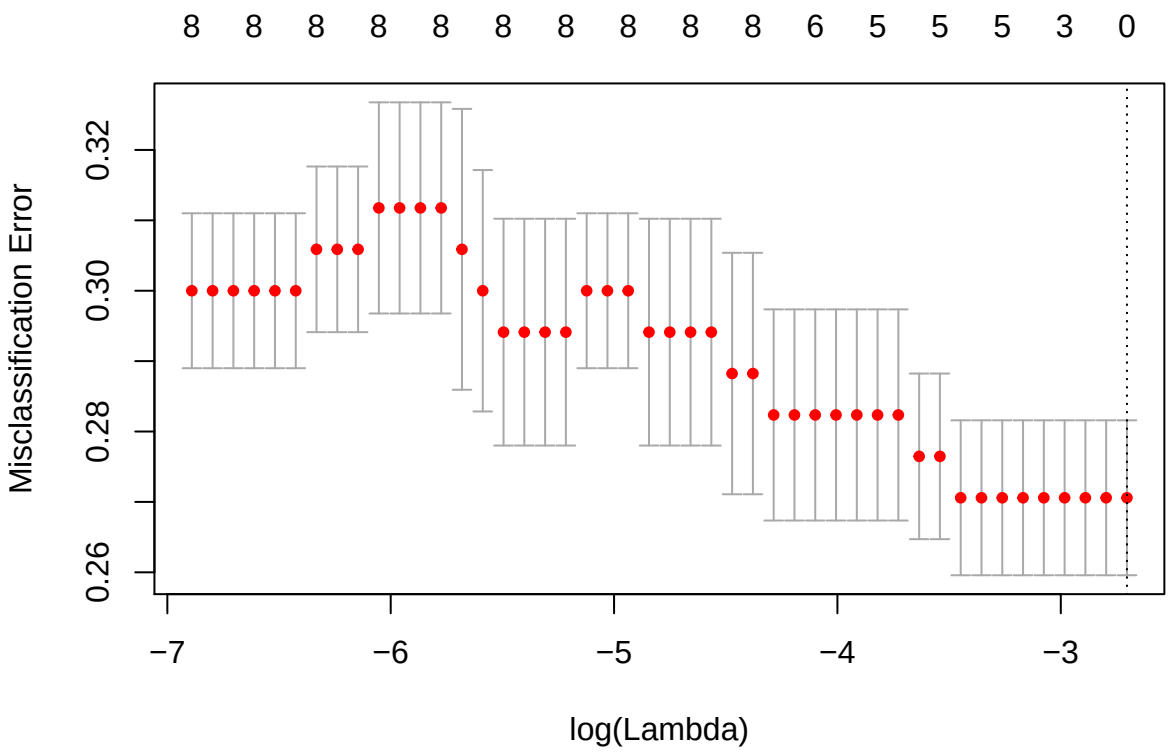
The lasso model is performed. The best performing model has a lambda of 0.067 and a corresponding accuracy of 0.729 (72.9%) and a standard deviation of 0.011 (1.1%). The accuracy is equal to the base model, however the standard deviation has decreased. As the plot shows, the optimal lambda has zero variables included, corresponding with the base model, just as the first cross-validation of best subset selection.

```
lassocv <- cv.glmnet(trainPreferences, chicken, alpha = 1, foldid = foldid, family = "binomial", type.m
resultsLasso(lassocv, measure = "acc")
```

```
## $glmnet.fit
##
## Call:  glmnet(x = trainPreferences, y = chicken, alpha = 1, family = "binomial")
##
##       Df      %Dev    Lambda
##  [1,]  0 7.606e-16 0.0669500
##  [2,]  1 3.359e-03 0.0610000
##  [3,]  3 7.924e-03 0.0555900
##  [4,]  3 1.261e-02 0.0506500
##  [5,]  3 1.653e-02 0.0461500
##  [6,]  4 2.042e-02 0.0420500
##  [7,]  5 2.433e-02 0.0383100
##  [8,]  5 2.856e-02 0.0349100
##  [9,]  5 3.209e-02 0.0318100
```

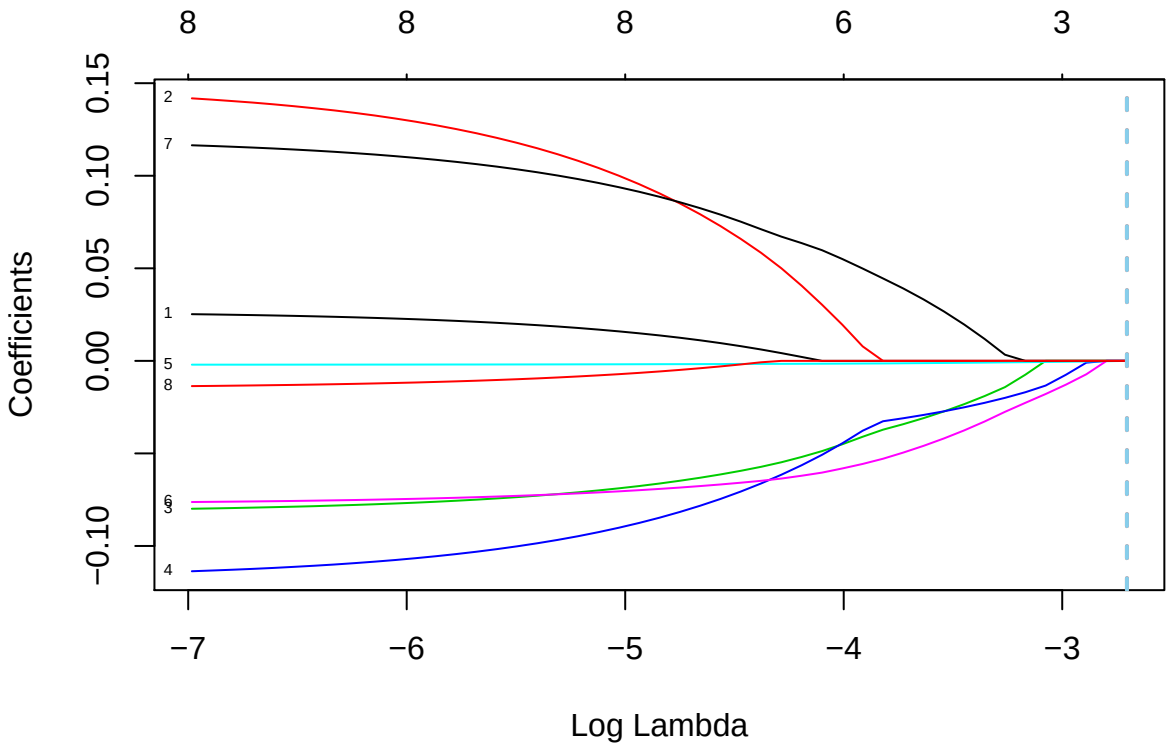
```
## [10,]  5 3.503e-02 0.0289800
## [11,]  5 3.749e-02 0.0264100
## [12,]  5 3.955e-02 0.0240600
## [13,]  5 4.127e-02 0.0219200
## [14,]  6 4.355e-02 0.0199800
## [15,]  6 4.593e-02 0.0182000
## [16,]  6 4.792e-02 0.0165800
## [17,]  7 4.969e-02 0.0151100
## [18,]  7 5.117e-02 0.0137700
## [19,]  8 5.255e-02 0.0125500
## [20,]  8 5.378e-02 0.0114300
## [21,]  8 5.482e-02 0.0104200
## [22,]  8 5.568e-02 0.0094900
## [23,]  8 5.641e-02 0.0086470
## [24,]  8 5.702e-02 0.0078790
## [25,]  8 5.752e-02 0.0071790
## [26,]  8 5.795e-02 0.0065410
## [27,]  8 5.831e-02 0.0059600
## [28,]  8 5.860e-02 0.0054310
## [29,]  8 5.885e-02 0.0049480
## [30,]  8 5.906e-02 0.0045090
## [31,]  8 5.923e-02 0.0041080
## [32,]  8 5.937e-02 0.0037430
## [33,]  8 5.949e-02 0.0034110
## [34,]  8 5.959e-02 0.0031080
## [35,]  8 5.968e-02 0.0028320
## [36,]  8 5.975e-02 0.0025800
## [37,]  8 5.981e-02 0.0023510
## [38,]  8 5.985e-02 0.0021420
## [39,]  8 5.989e-02 0.0019520
## [40,]  8 5.993e-02 0.0017780
## [41,]  8 5.995e-02 0.0016200
## [42,]  8 5.998e-02 0.0014760
## [43,]  8 6.000e-02 0.0013450
## [44,]  8 6.001e-02 0.0012260
## [45,]  8 6.003e-02 0.0011170
## [46,]  8 6.004e-02 0.0010180
## [47,]  8 6.005e-02 0.0009272
##
## $bestlambda
## [1] 0.06695276
##
## $bestnmean
## [1] 0.7294118
##
## $bestnsd
## [1] 0.01100487
##
## $lambda1se
## [1] 0.06695276
```

```
accLasso <- 1 - min(lassocv$cvm)
plot(lassocv)
```



As can be seen the best lambda includes 0 variables, which is evidence for a weak model.

```
plotShrinkage(trainPreferences, chicken, lassocv)
```




```
predict(lassocv, type = "coefficients", s = lassocv$lambda.min)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  -0.9916402
## negRecPooled  .
## posRec1       .
## posRec2       .
## altruism1     .
## altruism2     .
## trust         .
## risk1         .
## risk2         .
```

Boosting

The final model is boosting. First the parameters are set to train the model.

```
fitControl <- trainControl(method = "cv", number = 5)
gbmGrid <-  expand.grid(interaction.depth = c(1, 2, 4, 8), n.trees = (1:10)*200, n.minobsinnode = 10, sh
```

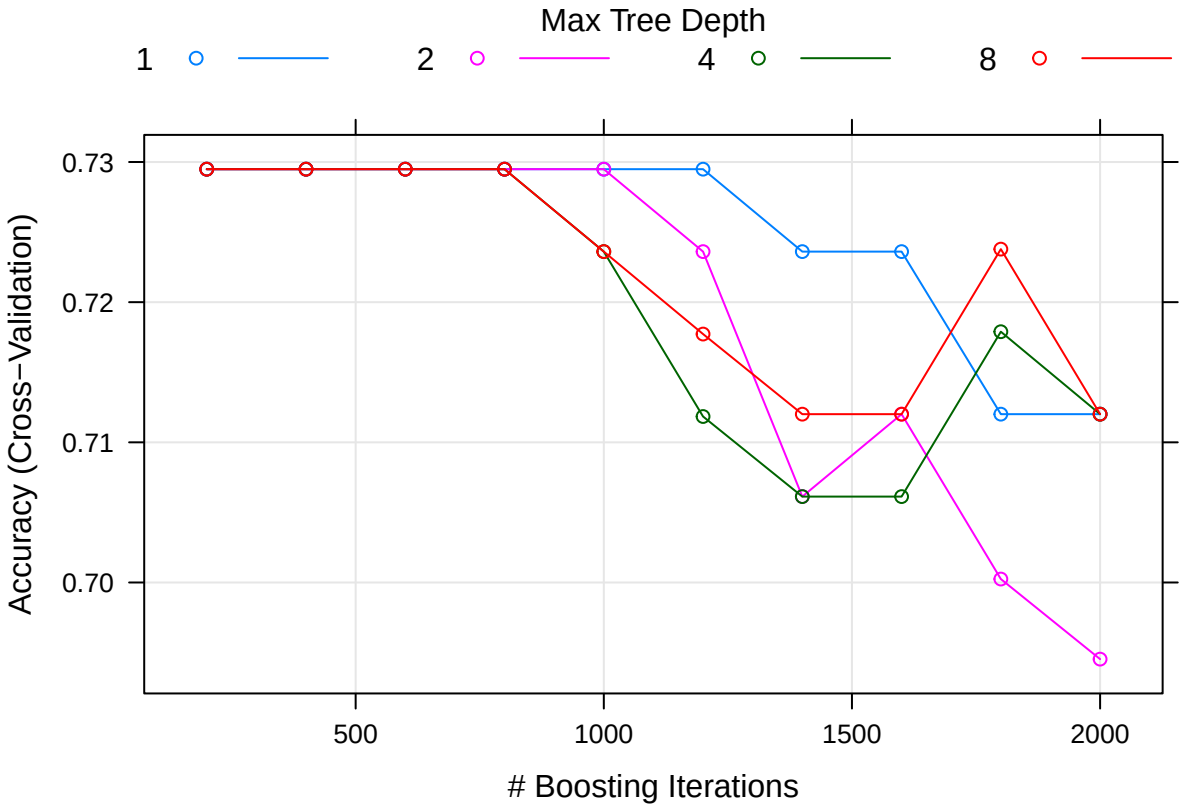
An accuracy of 0.729 (72.9%) is found, which is the same as the base model, the plot shows that nothing is learned and only worsens the model, so it seems like noise is learned.

```
set.seed(64)
gbmFit <- train(chicken ~ ., data = trainChicken, method = "gbm", verbose = FALSE, trControl = fitContro
gbmFit
```

```
## Stochastic Gradient Boosting
##
## 170 samples
##   8 predictor
##   2 classes: 'chicken', 'dare'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 135, 136, 136, 136, 137
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   200     0.7294882   0.000000000
##   1                   400     0.7294882   0.000000000
##   1                   600     0.7294882   0.000000000
##   1                   800     0.7294882   0.000000000
##   1                  1000     0.7294882   0.000000000
##   1                  1200     0.7294882   0.000000000
##   1                  1400     0.7236058  -0.011180124
##   1                  1600     0.7236058  -0.011180124
##   1                  1800     0.7120092  -0.032260679
##   1                  2000     0.7120092  -0.013121066
##   2                   200     0.7294882   0.000000000
##   2                   400     0.7294882   0.000000000
##   2                   600     0.7294882   0.000000000
##   2                   800     0.7294882   0.000000000
##   2                  1000     0.7294882   0.000000000
##   2                  1200     0.7236058  -0.011180124
##   2                  1400     0.7061268  -0.041467379
##   2                  1600     0.7120092  -0.013121066
##   2                  1800     0.7002445  -0.033537992
##   2                  2000     0.6945302  -0.043631720
##   4                   200     0.7294882   0.000000000
##   4                   400     0.7294882   0.000000000
##   4                   600     0.7294882   0.000000000
##   4                   800     0.7294882   0.000000000
##   4                  1000     0.7236058  -0.011180124
##   4                  1200     0.7118411  -0.030508475
##   4                  1400     0.7061268  -0.022357868
##   4                  1600     0.7061268  -0.022357868
##   4                  1800     0.7178915   0.068398164
##   4                  2000     0.7120092   0.057218040
##   8                   200     0.7294882   0.000000000
##   8                   400     0.7294882   0.000000000
##   8                   600     0.7294882   0.000000000
##   8                   800     0.7294882   0.000000000
##   8                  1000     0.7236058  -0.011180124
##   8                  1200     0.7177235  -0.021301775
##   8                  1400     0.7120092  -0.013121066
##   8                  1600     0.7120092   0.008698033
##   8                  1800     0.7237841   0.066215641
##   8                  2000     0.7120092   0.057218040
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.001
```

```
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 200,
##  interaction.depth = 1, shrinkage = 0.001 and n.minobsinnode = 10.
```

```
plot(gbmFit)
```




```
gbmFit$results[which.max(gbmFit$results$Accuracy),1:6]
```

```
##   shrinkage interaction.depth n.minobsinnode n.trees  Accuracy Kappa
## 1     0.001                 1             10     200 0.7294882     0
```

```
accBoost <- max(gbmFit$results$Accuracy)
```

Final models

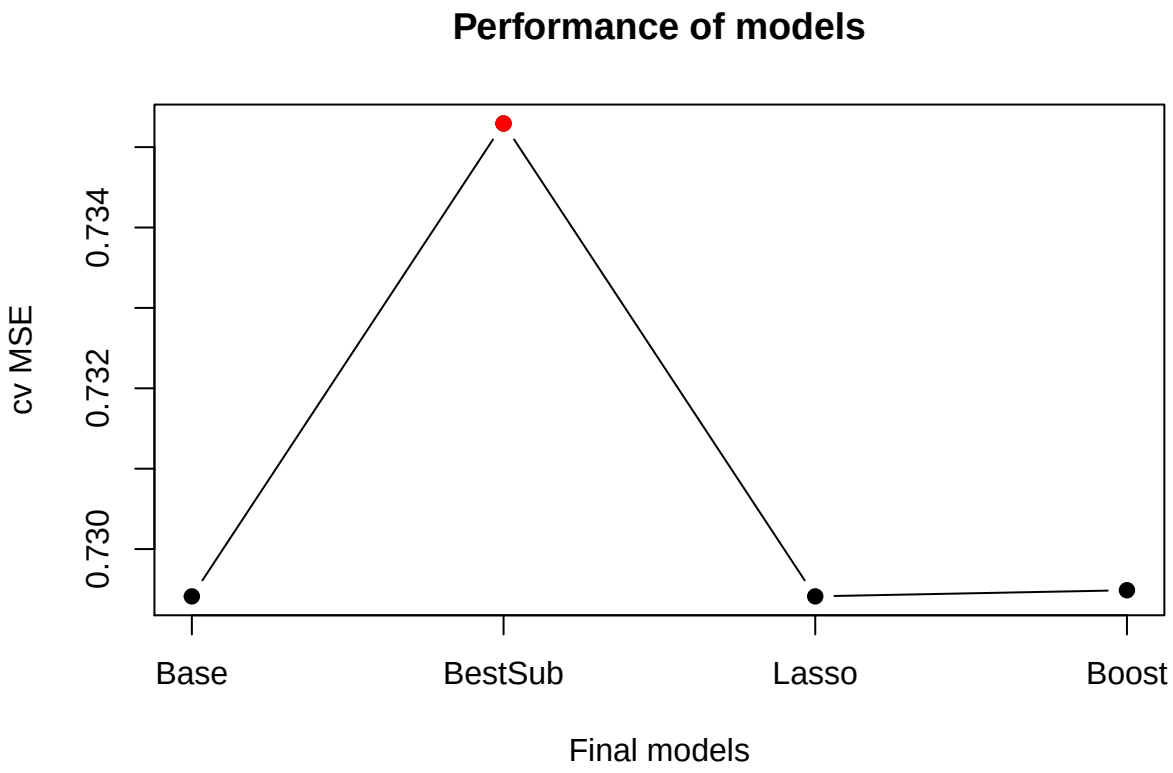
The final models are inspected. As can be seen the best performing model is the best subset selection method. All other methods do not perform better than the base model. The best subset selection method improves just a small 0.008 (0.8%). This shows that there is no strong evidence that this substantial better model than the base model and that it is a weak model.

```
finalModels <- data.frame(acc = c(accBase, accBestSub, accLasso, accBoost), row.names = c(c("Base", "Bes
finalModels
```

```
##               acc
## Base    0.7294118
## BestSub 0.7352941
```

```
## Lasso   0.7294118
## Boost   0.7294882
```

```
plotModels(finalModels$acc, xlab = "Final models", axislabels = rownames(finalModels), classification =
```




```
## [1] 0.7294118 0.7352941 0.7294118 0.7294882
```

Final predictions

Now the final predictions on test set are performed. The model is trained on all training data. The base model has an accuracy of 0.706 (70.6%) on the test set. The lasso model performs better and has an accuracy of 0.735 (73.5%). The final models performs 4% better on the test set compared to the base model and 1.9% in absolute percentage points.

```
ytrue <- qualtrics[test, "chicken"]

majorityIndex <- which.max(table(trainChicken$chicken))
majorityClass <- levels(trainChicken$chicken)[majorityIndex]
sum(ytrue == majorityClass)/length(ytrue)
```

```
## [1] 0.7058824
```

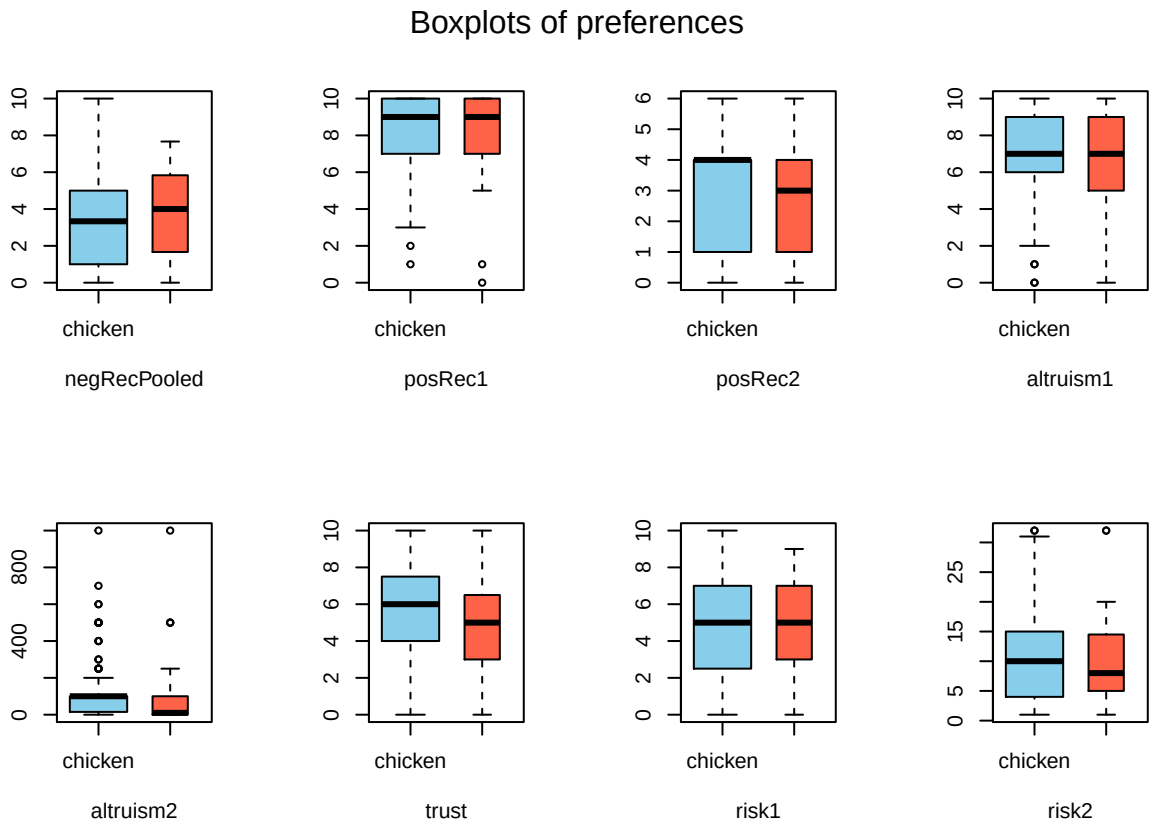
```
finalFit <- glm(mdl3, trainChicken, family = "binomial")
yhatProb <- predict(finalFit, qualtrics[test,], type = "response")
yhatClass <- ifelse(yhatProb < 0.5, "chicken", "dare")
sum(ytrue == yhatClass)/length(ytrue)
```

```
## [1] 0.7352941
```

Significance testing

Models are created to check for significance from less complex to more complex on all data, thus including train and test set. Starting with bivariate analysis, using Wilcoxon rank test and a non-directional test with a p-value threshold of 0.10. Wilcoxon rank test is chosen because most of these variables are asymmetric and mesokurtic or leptokurtic and this is a non-parametric method, which makes no assumptions based on normality and is less influenced by outliers. This shows a significant relation for altruism2 (0.02) and a weakly significant relation for trust (0.08), both variables were included in the final model. But these variables were not found using the lasso and in boosting. Risk1 is also insignificant (0.29).

```
qualtricsChicken <- select(qualtrics, -c(dilemma, offer, respond, public))
attach(qualtricsChicken)
chickenBinary <- ifelse(chicken == "chicken", 0, 1)
plotBoxes(qualtrics, chicken)
```




```
apply(preferences, 2, function(x) wilcox.test(x ~ chicken))
```

```
## $negRecPooled
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x by chicken
## W = 3562.5, p-value = 0.1221
## alternative hypothesis: true location shift is not equal to 0
##
##
## $posRec1
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x by chicken
## W = 4014, p-value = 0.7193
## alternative hypothesis: true location shift is not equal to 0
##
##
## $posRec2
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x by chicken
## W = 4539, p-value = 0.2851
## alternative hypothesis: true location shift is not equal to 0
##
##
## $altruism1
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x by chicken
## W = 4508.5, p-value = 0.3275
## alternative hypothesis: true location shift is not equal to 0
##
##
## $altruism2
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x by chicken
## W = 5024, p-value = 0.01699
## alternative hypothesis: true location shift is not equal to 0
##
##
## $trust
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x by chicken
## W = 4807, p-value = 0.07571
## alternative hypothesis: true location shift is not equal to 0
##
##
## $risk1
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x by chicken
## W = 3745, p-value = 0.2857
## alternative hypothesis: true location shift is not equal to 0
##
##
## $risk2
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x by chicken
## W = 4171, p-value = 0.9437
## alternative hypothesis: true location shift is not equal to 0
```

Now, the multivariate models are examined. The model with one variable altruism2, shows no significant relation between altruism2. This might be caused because this test uses a parametric method, or simply because there is no strong relation.

```
mdl0 <- chicken ~ 1
fit0 <- glm(mdl0, data = qualtricsChicken, family = binomial)

mdl1 <- chicken ~ altruism2
fit1 <- glm(mdl1, data = qualtricsChicken, family = binomial)
summary(fit1)
```

```
##
## Call:
## glm(formula = mdl1, family = binomial, data = qualtricsChicken)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8501  -0.8270  -0.8044   1.5448   2.1202
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.831909   0.192417  -4.323 1.54e-05 ***
## altruism2   -0.001304   0.001117  -1.168    0.243
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 239.78  on 203  degrees of freedom
## Residual deviance: 238.24  on 202  degrees of freedom
## AIC: 242.24
##
## Number of Fisher Scoring iterations: 4
```

Adding trust to this model is weakly significant with a p-value of 0.09.

```
mdl2 <- chicken ~ altruism2 + trust
fit2 <- glm(mdl2, data = qualtricsChicken, family = binomial)
summary(fit2)
```

```
##
## Call:
## glm(formula = mdl2, family = binomial, data = qualtricsChicken)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0456  -0.8352  -0.7189   1.3271   2.0735
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -0.3181930  0.3517388  -0.905    0.366
## altruism2   -0.0008457  0.0011140  -0.759    0.448
## trust       -0.1077434  0.0635503  -1.695    0.090 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 239.78  on 203  degrees of freedom
## Residual deviance: 235.33  on 201  degrees of freedom
## AIC: 241.33
##
## Number of Fisher Scoring iterations: 4
```

The final model shows no significant relation for any variables, trust is no longer weakly significant (0.0132).

```
finalModel <- chicken ~ altruism1 + trust + risk2
finalFit <- glm(finalModel, data = qualtricsChicken, family = binomial)
summary(finalFit)
```

```
##
## Call:
## glm(formula = finalModel, family = binomial, data = qualtricsChicken)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1352  -0.8260  -0.7226   1.3422   1.8801
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.040209   0.547230  -0.073    0.941
## altruism1   -0.052720   0.068559  -0.769    0.442
## trust       -0.100904   0.067019  -1.506    0.132
## risk2       -0.003582   0.022466  -0.159    0.873
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 239.78  on 203  degrees of freedom
## Residual deviance: 235.35  on 200  degrees of freedom
## AIC: 243.35
##
## Number of Fisher Scoring iterations: 4
```

Conducting an anova test confirms these findings and shows that adding altruism2 is insignificant, but adding trust to this model is weakly significant, adding risk2 actually lowers the AIC score. All in all, this shows that there is no strong relation between the variables and the choice made on the chicken game and that this is a weak model.

```
anova(fit0, fit1, fit2, finalFit, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: chicken ~ 1
## Model 2: chicken ~ altruism2
## Model 3: chicken ~ altruism2 + trust
## Model 4: chicken ~ altruism1 + trust + risk2
```

```
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       203     239.78
## 2       202     238.24  1  1.53875   0.2148
## 3       201     235.33  1  2.91067   0.0880 .
## 4       200     235.35  1 -0.02468
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

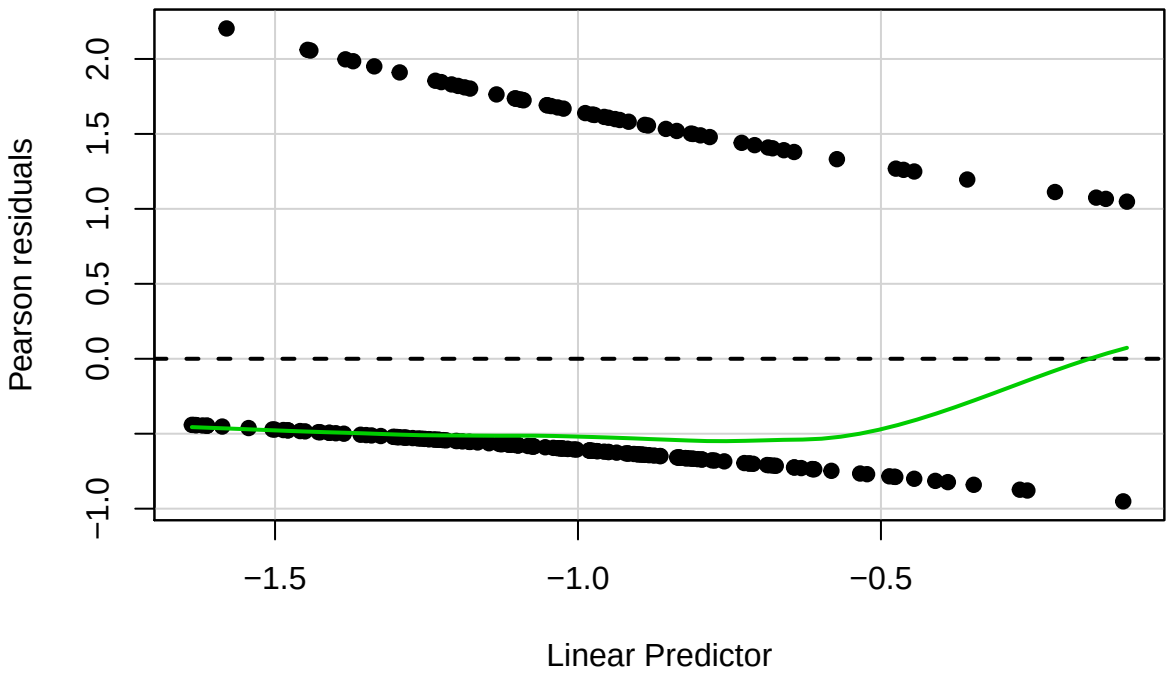
Diagnostics of the model

In this section the diagnostics of the models are tested and inspected. Beginning with a general overview plot of all the model.

Non-linearity

Next non-linearity is tested. There is some evidence of non-linearity on the end of the spectrum, however this seems to be mostly caused by the outliers, and there is no strong evidence that non-linearity would improve this model.

```
residualPlot(finalFit, pch = 19)
```



Correlation of error terms

The second test is correlation of error terms, however no evidence for correlation of error-terms is found in a Durbin Watson test.

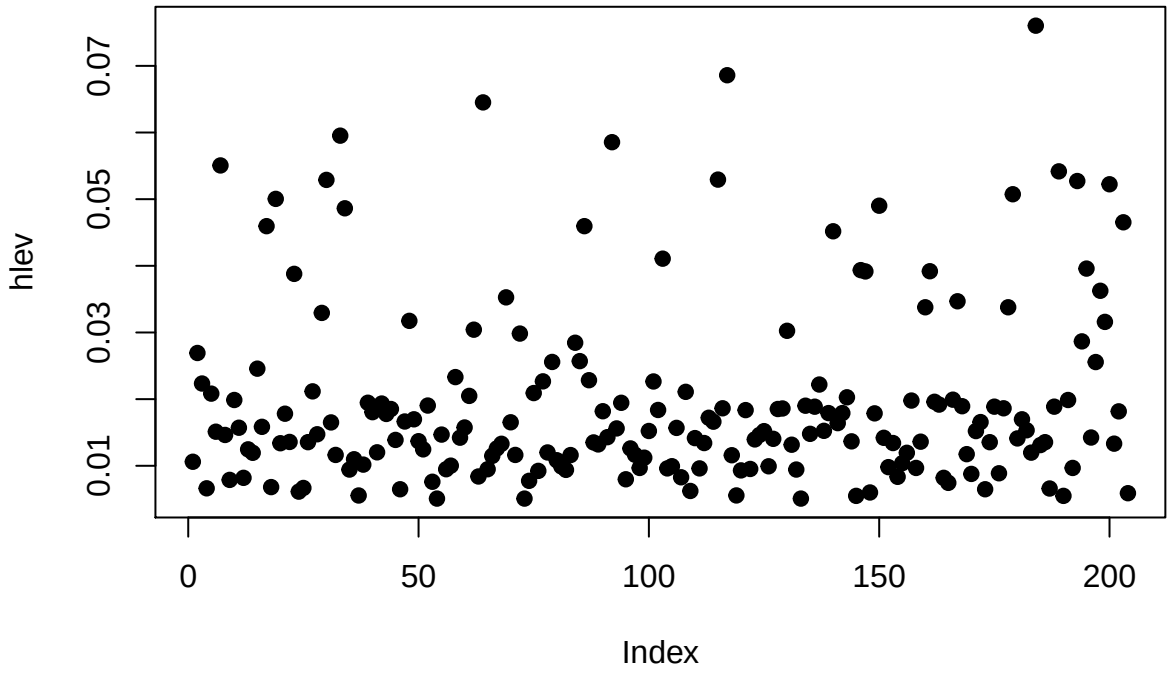
```
durbinWatsonTest(finalFit)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.0707376      1.844663   0.282
##  Alternative hypothesis: rho != 0
```

High leverage points

The high leverage plot gives no evidence for strong values.

```
hlev <- hatvalues(finalFit)
plot(hlev, pch = 19)
```



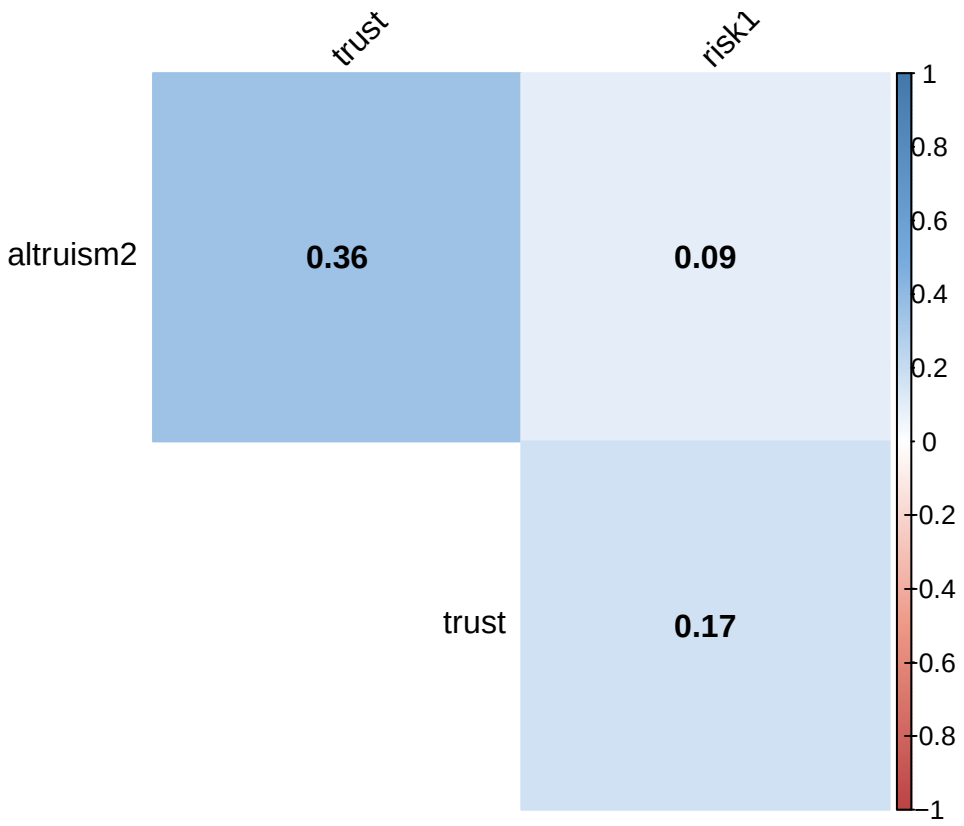
Collinearity

The last diagnostic of collinearity, as observed above there is high collinearity between variables and this possibly influences the coefficients of the variables. However there is no evidence for multicollinearity as the VIF shows.

```
vif(finalFit)
```

```
## altruism1     trust     risk2
##  1.156697  1.167780  1.028397
```

```
plotCor(data.frame(altruism2, trust, risk1))
```




```
## $cor
##           altruism2 trust risk1
## altruism2      1.00  0.36  0.09
## trust          0.36  1.00  0.17
## risk1          0.09  0.17  1.00
```