A Cross Validation R-function with Examples

M & W

24 September 2019

R-package xvalglms

We developed a package *xvalglms* with the R-function `xval.glm` for cross validation of generalized linear models. The package can be downloaded from a Github page. The code of the function is given in the following box.

```
library("xvalglms")
```

```
## Loading required package: foreach

## Loading required package: doParallel

## Loading required package: iterators

## Loading required package: parallel

## Loading required package: ggplot2

## Loading required package: gridExtra
```

The `xval.glm` function requires a few input statements:

- A data frame with the data;
- A list of models to be compared;
- `glm.family`: A family for generalized linear models, which defaults to the Gaussian family, but can be changed to binomial or poisson or any other family (see glm help function);
- `folds`: The number of folds, K, which defaults to 10;
- `repeats`: The number of repeats, i.e. cycles of the K-fold cross validation (defaults to 200);
- `loss`: loss function for the GLM (default = NULL equals RMSE);
- `numCore`: number of cores for use with parallel computation (default = NULL, equals no parallelization);
- `plots`: whether plots should be made or not (default is yes);
- `gray`: output greyscale plots;
- `seed`: a seed can be inserted to make the results reproducible.

Tutorial examplary cross validation analyses

In the following we show several data analysis procedures comparable to a one sample t-test, two sample t-test, simple regression, multiple regression, and logistic regression where we use cross validation as the tool for help making decisions about model selction.

Overall, the research questions we discuss can be translated into a comparison of two (or more) models. The translated question is then which of the two models gives the most accurate predictions. The model that provides the most accurate predictions is selected and interpreted.

For comparison, for every example we also show the standard analysis approach.

Is the mean equal to a prespecified number?

This question compares to a one sample t-test, where we see whether the mean differs from a prespecified number.

Example 1

Often we ask ourselves the question whether the mean of a population is equal to a prespecified number (μ_0).

Let us first load some data. Description

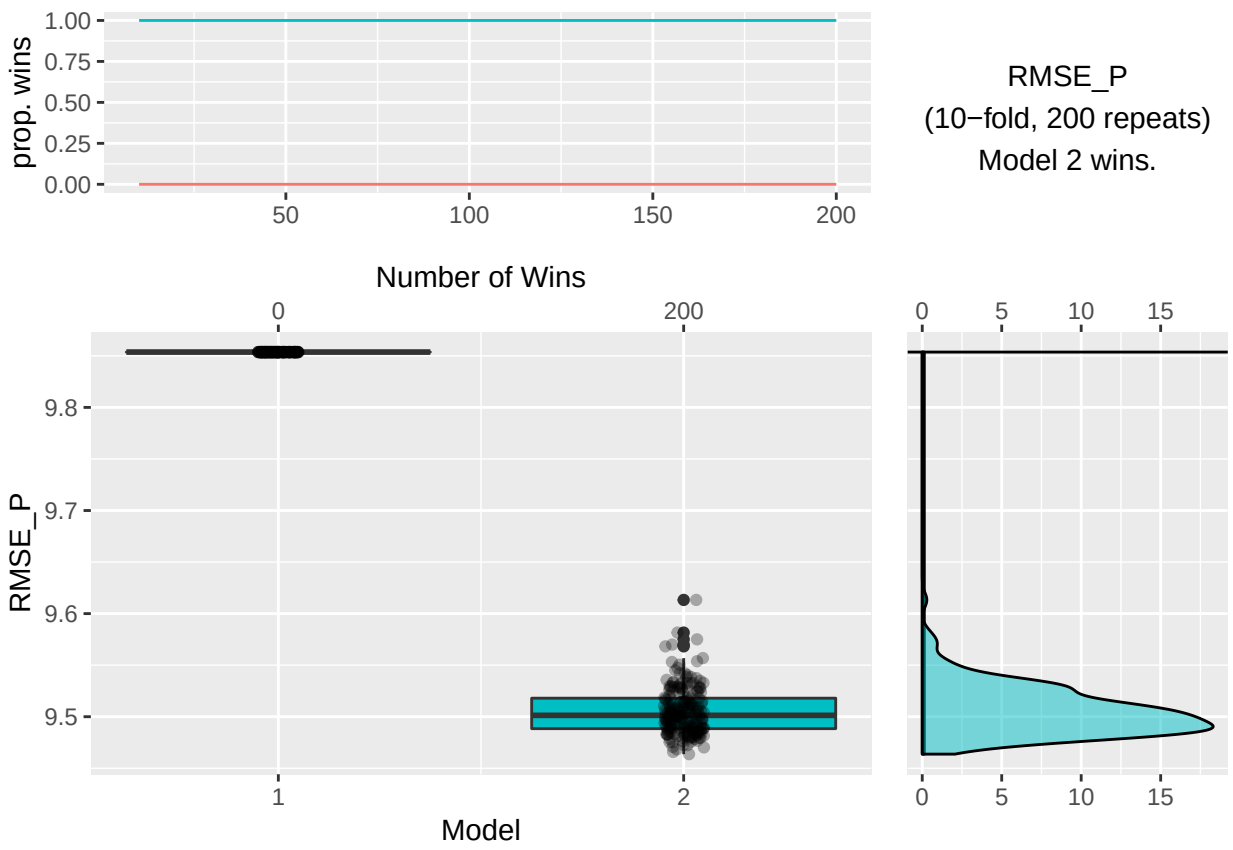
```
library(foreign)
mydat = read.spss("https://stats.idre.ucla.edu/stat/data/hsb2.sav",
    to.data.frame = TRUE)
summary(mydat$WRITE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   31.00   45.75   54.00   52.77   60.00   67.00
```

We test whether the mean in writing is equal to 50 (i.e. $\mu_0 = 50$) by rephrasing the question as follows: if we estimate the mean from the data and use this mean to make predictions is that more accurate than simply predicting 50.

```
mu0 = 50
# The two models
models = vector(mode = "list", length = 2)
models[[1]] = I(WRITE - 50) ~ 0
models[[2]] = I(WRITE - 50) ~ 1
output = xval.glm(data = mydat, models)
```

```
## Running Cross-validation...done [ 5 sec ]
```




```
## Results for (10-fold, 200 repeats)
##   Model:                                  |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] I(WRITE - 50) ~ 0                  |     0%   |   9.854 |   9.854 |   9.854 |
##   [ 2] I(WRITE - 50) ~ 1                  |   100%   |   9.472 |   9.505 |   9.557 |
```

We see that the predictions really become better when using the sample mean, compared to simply predict 50. Furthermore, in all 100 repeats the prediction error of the more complicated model (where we estimate the mean to make a prediction) wins from just predicting 50; we can see that at the top of the Figure where the number of 'wins' is reported.

Therefore, we better use the estimated mean to make a prediction. The outcome of the analysis is

```
result = glm(models[[2]], data = mydat)
summary(result)
```

```
##
## Call:
## glm(formula = models[[2]], data = mydat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -21.775   -7.025    1.225    7.225   14.225
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.7750     0.6702    4.14 5.12e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 89.84359)
##
##     Null deviance: 17879  on 199  degrees of freedom
## Residual deviance: 17879  on 199  degrees of freedom
## AIC: 1470.2
##
## Number of Fisher Scoring iterations: 2
```

where we see that the estimated mean is 50 plus the estimated intercept, which is 52.775. This latter number would be the predicted value for future cases.

Standard analysis approach

```
t.test(mydat$WRITE - 50)
```

```
##
##  One Sample t-test
##
## data:  mydat$WRITE - 50
## t = 4.1403, df = 199, p-value = 5.121e-05
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  1.453321 4.096679
## sample estimates:
## mean of x
##     2.775
```

Example 2

Another example from Howell, page 187-188. Nurcombe et al. (1984) reported on an intervention program for the mothers of low-birthweight (LBW) infants. These infants present special problems to their parents because they are unresponsive and unpredictable, in addition to being at risk for physical and developmental problems. One of the dependent variables in the study was the Psychomotor Development Index (PDI). This scale was first administered to all infants in the study when they were six months old. The question is whether these children are different from the normative population mean of 100, usually found with this index. We have data from 56 infants.

```
rm(models)
PDI = read.table("http://www.uvm.edu/~dhowell/methods8/DataFiles/Tab7-1.dat",
    header = TRUE)
summary(PDI)
```

```
##     PDIscore
##  Min.   : 83.0
##  1st Qu.: 92.0
##  Median :102.0
##  Mean   :104.1
```

```
##  3rd Qu.:116.0
##  Max.   :127.0
```

```
PDI$constant = 1
```

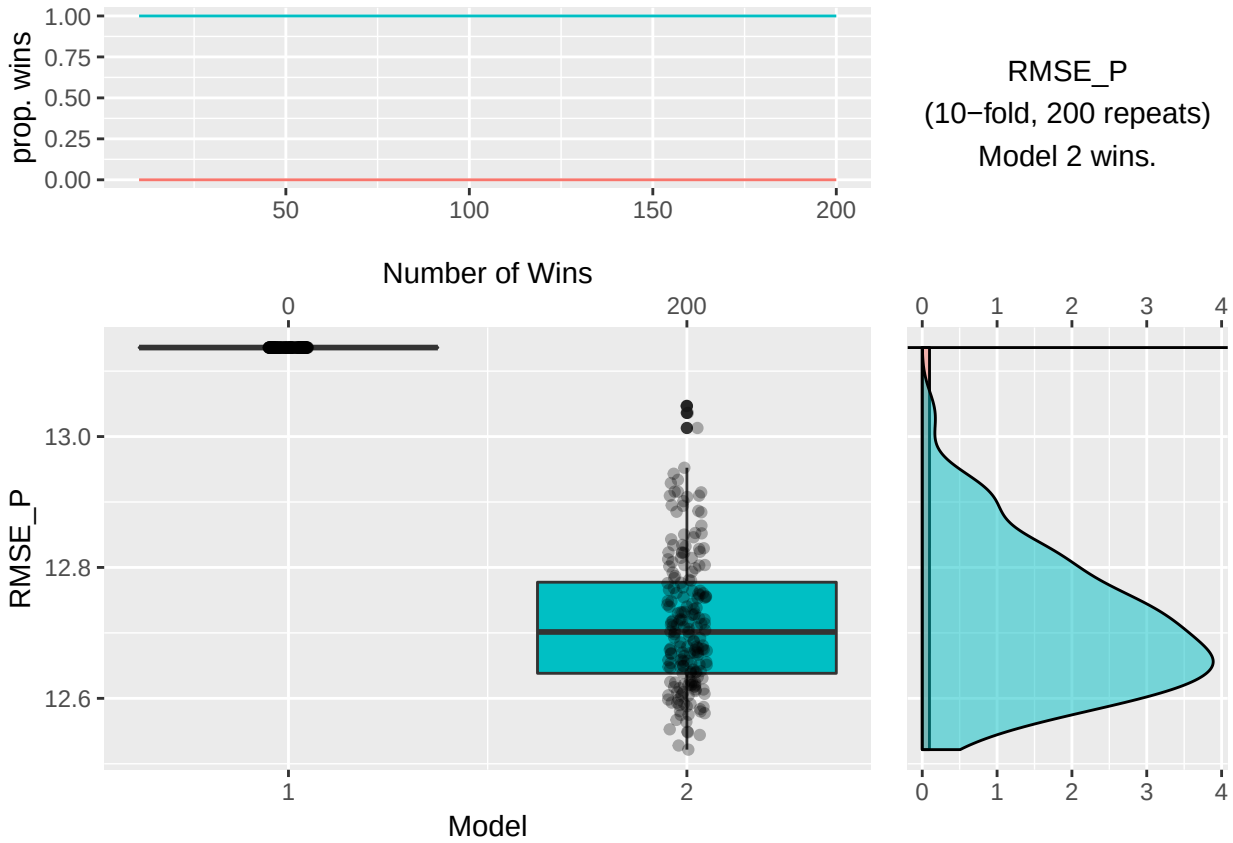
The observed mean is 104.125 and the standard deviation is 12.5843518. We rephrase the question as follows. Are the predictions obtained with the mean of the sample better than when we predict 100 everytime. In other words, does using the data help making better predictions. The Root Mean Square of prediction error for the prediction 100 is easily computed as

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - 100)^2}$$

which is 13.1359648, nevertheless we will also show the K-fold cross validation code for it.

```
# The two models
models = vector(mode = "list", length = 2)
models[[1]] = I(PDIscore - 100) ~ 0
models[[2]] = I(PDIscore - 100) ~ 1
output = xval.glm(data = PDI, models)
```

```
## Running Cross-validation...done [ 3.8 sec ]
```




```
## Results for (10-fold, 200 repeats)
```

```
##   Model:                                          |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] I(PDIscore - 100) ~ 0                      |     0%   |  13.136 |  13.136 |  13.136 |
##   [ 2] I(PDIscore - 100) ~ 1                      |   100%   |  12.553 |  12.715 |  12.934 |
```

The outcome of the analysis of the second model is

```
result = glm(models[[2]], data = PDI)
summary(result)
```

```
##
## Call:
## glm(formula = models[[2]], data = PDI)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -21.125  -12.125   -2.125   11.875   22.875
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.125      1.682   2.453   0.0174 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 158.3659)
##
##     Null deviance: 8710.1  on 55  degrees of freedom
## Residual deviance: 8710.1  on 55  degrees of freedom
## AIC: 445.55
##
## Number of Fisher Scoring iterations: 2
```

where we see that the estimated mean is 100 plus the estimated intercept, which is 104.125. This latter number would be the predicted value for future cases.

Standard analysis approach

```
t.test(PDI$PDIscore - 100)
```

```
##
##  One Sample t-test
##
## data:  PDI$PDIscore - 100
## t = 2.4529, df = 55, p-value = 0.01737
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.7548886 7.4951114
## sample estimates:
## mean of x
##     4.125
```

Is there a difference between two groups?

This analysis compares to a two sample t-test and can be easily generalized to a one-way ANOVA.

Example 1

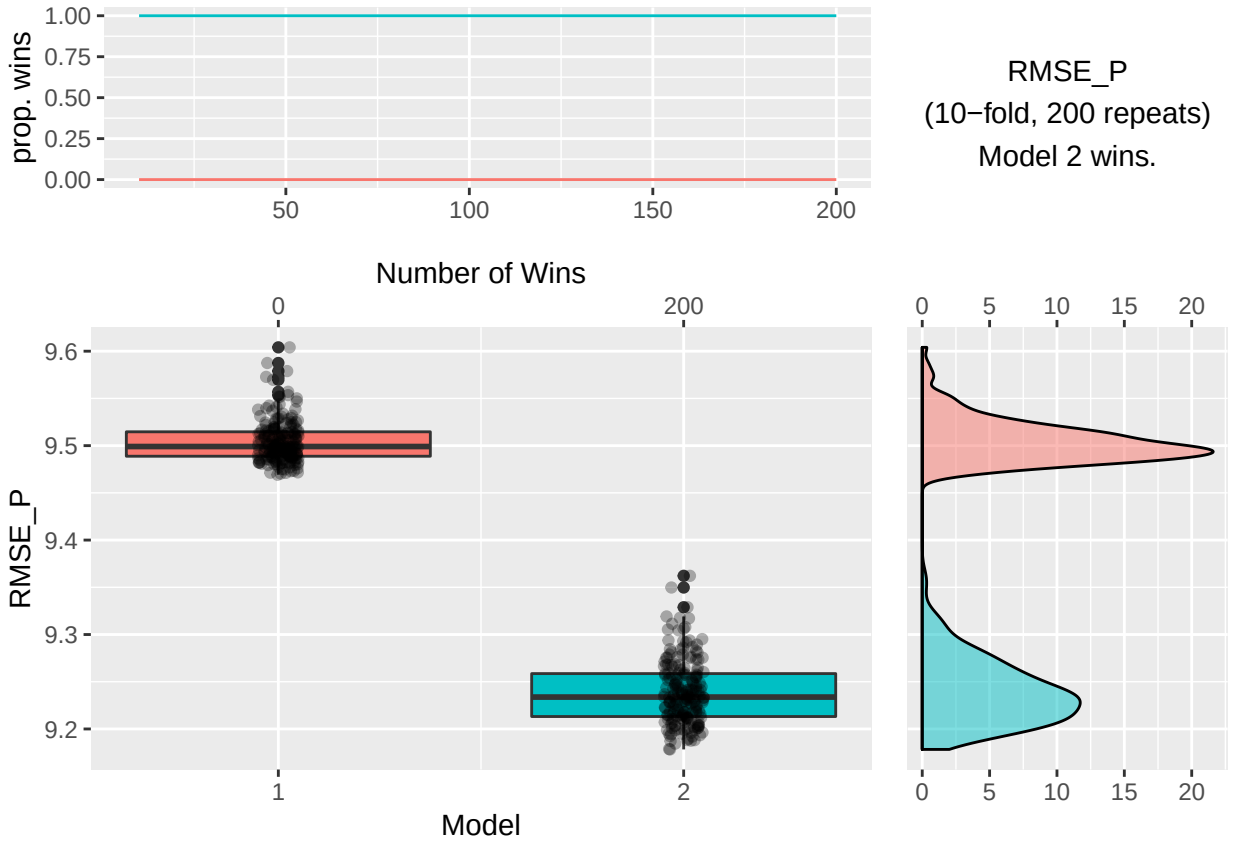
In the same data set we also have the variable gender.

Usually we test whether the mean of males is equal to the mean of females (one can wonder whether this is actually a sensible test as means are never exactly equal). We better ask ourselves the question whether the use of gender provides better predictions of writing compared to predictions without gender?

Therefore, we cross validate two models: The first only estimates an intercept (overall mean) so does not involve gender; The second uses also gender as a predictor.

```
# The two models
models = vector(mode = "list", length = 2)
models[[1]] = WRITE ~ 1
models[[2]] = WRITE ~ 1 + female2
output = xval.glm(data = mydat, models)
```

```
## Running Cross-validation...done [ 5.3 sec ]
```




```
## Results for (10-fold, 200 repeats)
##   Model:                                |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] WRITE ~ 1                        |     0%   |   9.473 |   9.504 |   9.557 |
##   [ 2] WRITE ~ 1 + female2              |   100%   |   9.188 |   9.239 |   9.317 |
```

The results show that with the model including gender the predictions are more accurate.

The outcome of the analysis is

```
result = glm(models[[2]], data = mydat)
summary(result)
```

```
##
## Call:
## glm(formula = models[[2]], data = mydat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -19.991   -6.371    1.879    7.009   16.879
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  50.1209     0.9628  52.057  < 2e-16 ***
## female2       4.8699     1.3042   3.734 0.000246 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 84.35687)
##
##     Null deviance: 17879  on 199  degrees of freedom
## Residual deviance: 16703  on 198  degrees of freedom
## AIC: 1458.6
##
## Number of Fisher Scoring iterations: 2
```

where we see that the estimated mean for men is 50.1208791 and that for women is 54.9908257. For the future we would predict the first mean if we see a man, and the second mean when we see a woman.

The only assumption we make in this analysis is that the criterion variable is at least an interval variable. No normality, no homoscedasticity and we do not need the concept of degrees of freedom!

Standard analysis approach

```
t.test(WRITE ~ female2, data = mydat)
```

```
##
##  Welch Two Sample t-test
##
## data:  WRITE by female2
## t = -3.6564, df = 169.71, p-value = 0.0003409
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -7.499159 -2.240734
## sample estimates:
## mean in group 0 mean in group 1
##        50.12088        54.99083
```

The null hypothesis that the means are equal in the population is rejected.

Example 2

Another example from Howell, page 211.

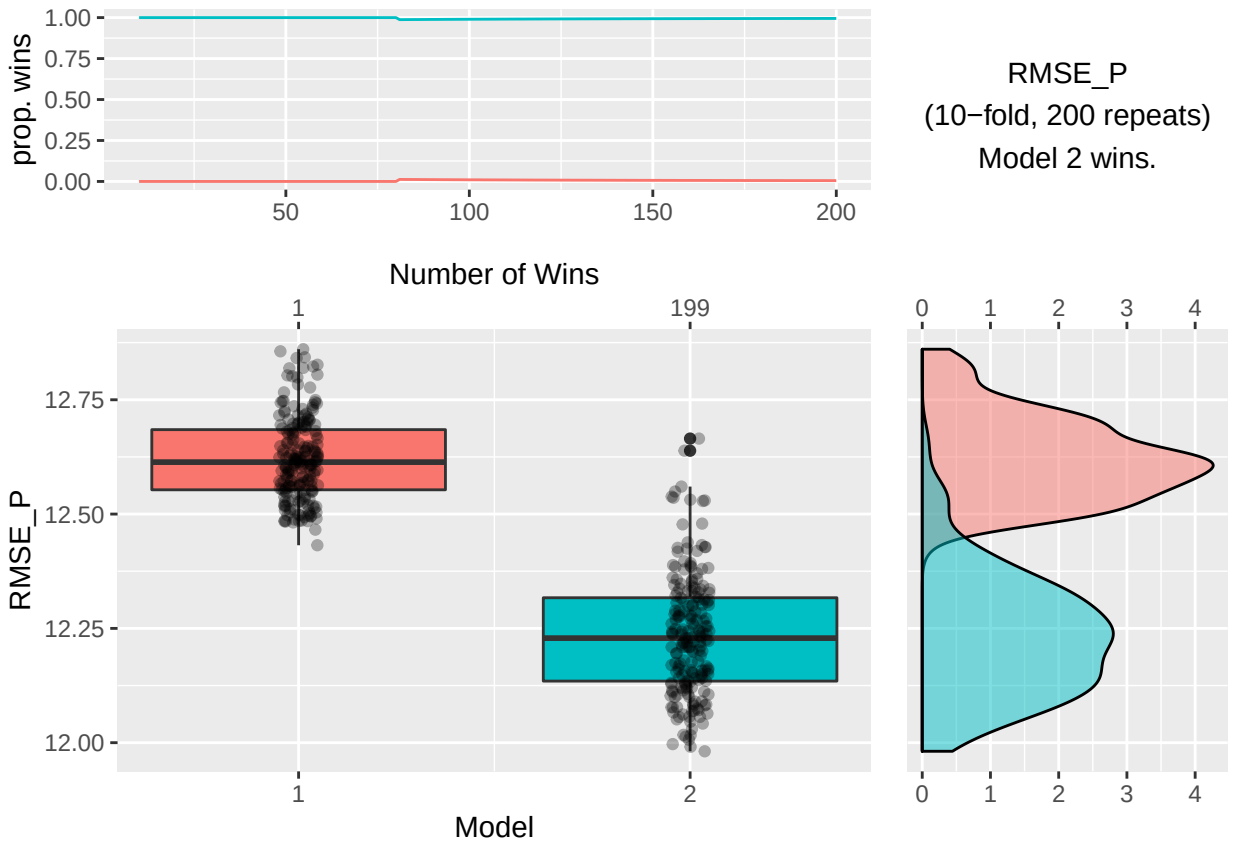
```
Arousal = read.spss("http://www.uvm.edu/~dhowell/methods8/DataFiles/Tab7-5.sav",
    to.data.frame = TRUE)
```

```
## re-encoding from CP1252
```

```
Arousal$Group = as.factor(Arousal$Group)

# The two models
models = vector(mode = "list", length = 2)
models[[1]] = Arousal ~ 1
models[[2]] = Arousal ~ 1 + Group
output = xval.glm(data = Arousal, models)
```

```
## Running Cross-validation...done [ 4.8 sec ]
```




```
## Results for (10-fold, 200 repeats)
##   Model:                                  |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] Arousal ~ 1                        |     0%   |  12.485 |  12.621 |  12.823 |
##   [ 2] Arousal ~ 1 + Group                |   100%   |  12.016 |  12.234 |  12.536 |
```

The second model gives better predictions and wins in all 100 repetitions. The estimated means for the two groups can be computed usin the oiutput of the model, i.e.

```
summary(glm(models[[2]], data = Arousal))
```

```
##
## Call:
## glm(formula = models[[2]], data = Arousal)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -32.003   -9.250   -0.702    9.397   30.100
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   24.000      2.032  11.812   <2e-16 ***
## Group2        -7.497      3.018  -2.484   0.0157 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 144.4847)
##
##     Null deviance: 9849.3  on 63  degrees of freedom
## Residual deviance: 8958.0  on 62  degrees of freedom
## AIC: 503.88
##
## Number of Fisher Scoring iterations: 2
```

Standard analysis approach

```
t.test(Arousal ~ Group, data = Arousal)
```

```
##
##  Welch Two Sample t-test
##
## data:  Arousal by Group
## t = 2.4917, df = 60.495, p-value = 0.01547
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   1.47935 13.51375
## sample estimates:
## mean in group 1 mean in group 2
##        24.00000        16.50345
```

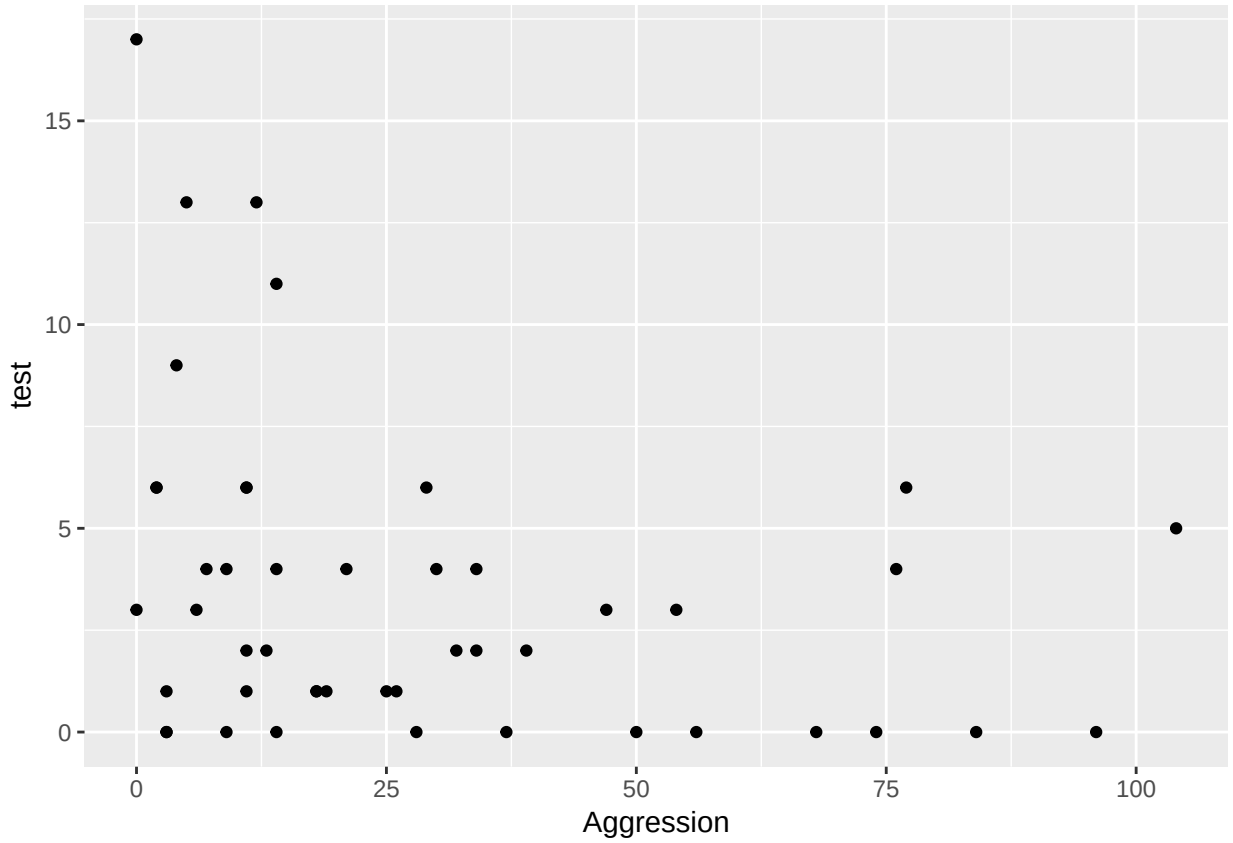
Is there a relationship between variable X and Y?

Univariate regression

A general goal of the study conducted by Margolin and Medina was to examine how children's information processing is related to a history of exposure to marital aggression. Data are collected for 47 children. The

variable `Aggression` is a measure of marital aggression that reflects physical, verbal, and emotional aggresion during the last year and the variable `test` is a child's score on a recall test.

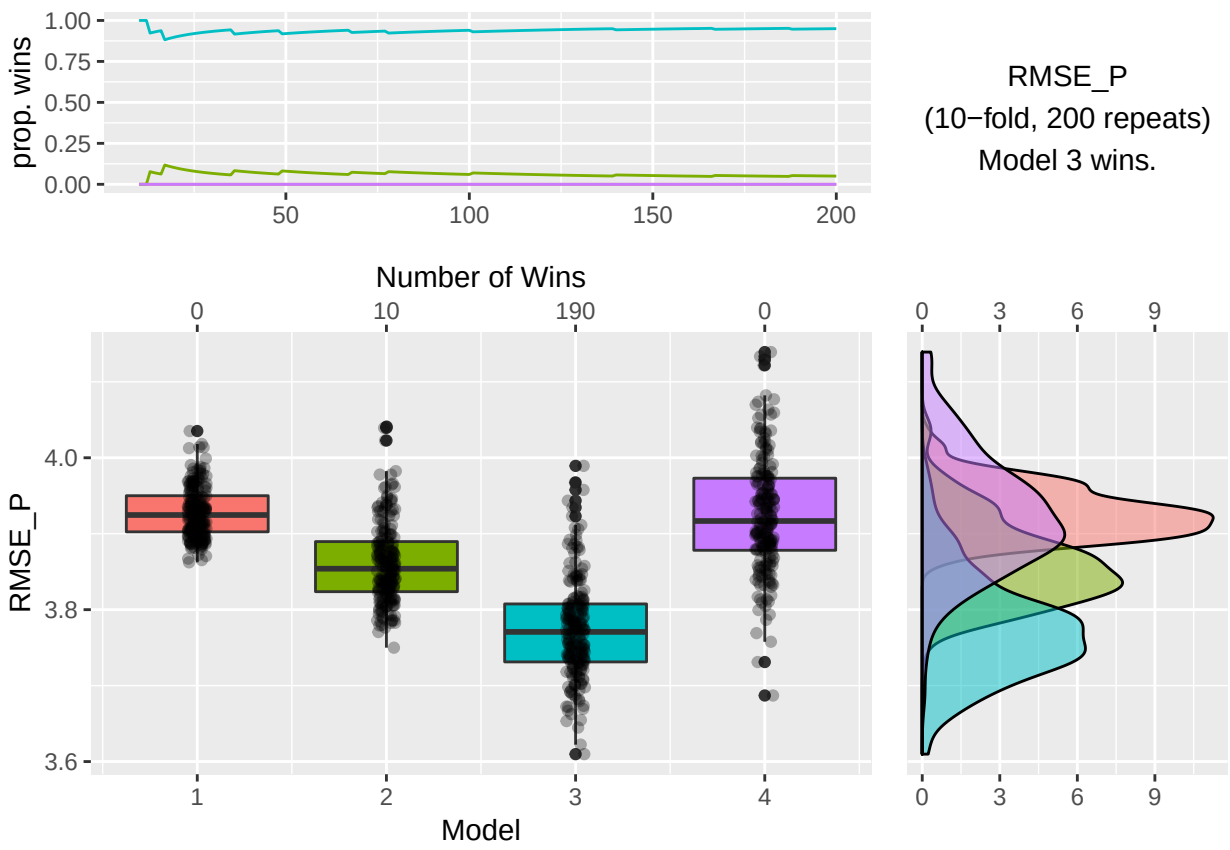
```
agdat = read.table("https://dornsife.usc.edu/assets/sites/239/docs/marital_agg_dat.txt",
    header = TRUE)
p <- ggplot(agdat, aes(Aggression, test))
p + geom_point()
```



We define four different models. The first says, we best predict with the overall mean. The second until fourth include agression as a predictor, but the functional forms differ. The second model defines a linear relationship, the third a quadratic, and the fourth a cubic polynomial.

```
# The four models
models = vector(mode = "list", length = 4)
models[[1]] = test ~ 1
models[[2]] = test ~ Aggression
models[[3]] = test ~ poly(Aggression, 2)
models[[4]] = test ~ poly(Aggression, 3)
output = xval.glm(data = agdat, models)
```

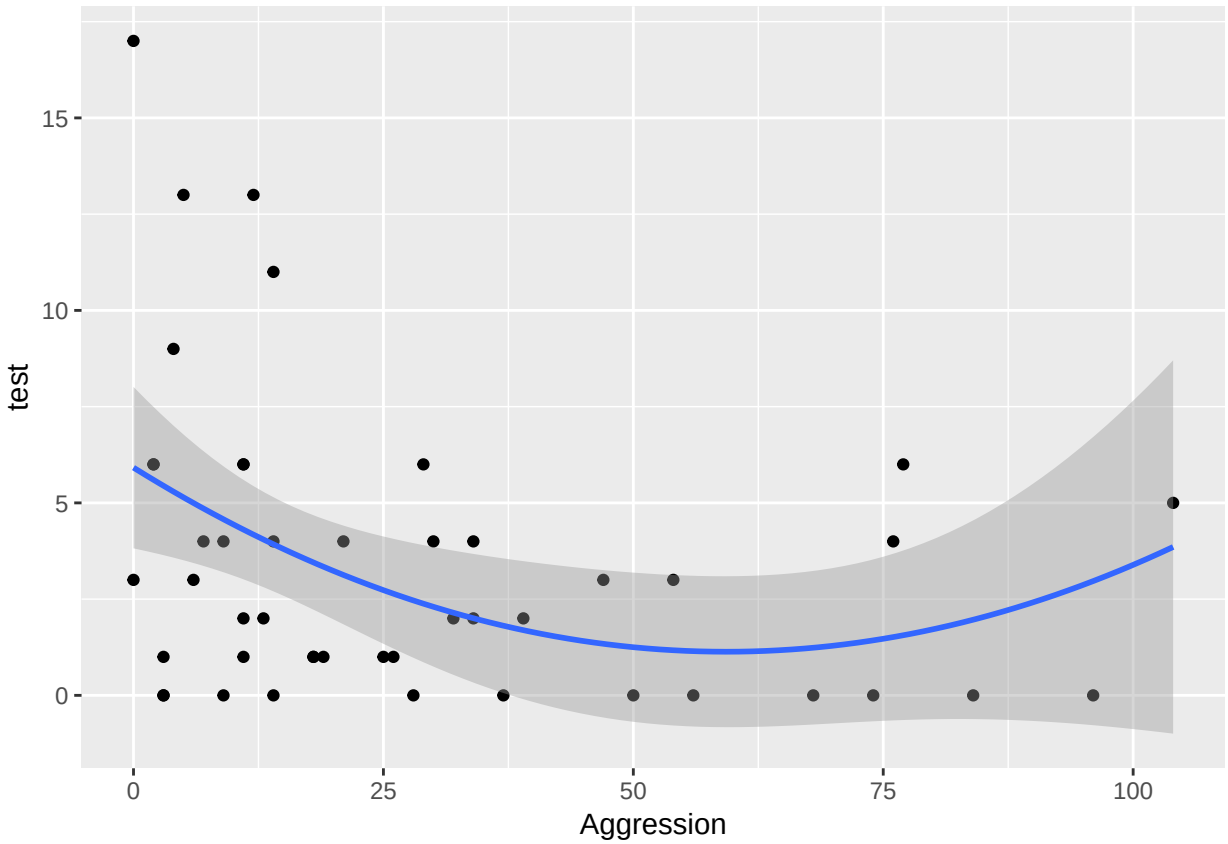
```
## Running Cross-validation...done [ 9.3 sec ]
```




```
## Results for (10-fold, 200 repeats)
##   Model:                                |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] test ~ 1                         |     0%   |   3.879 |   3.928 |   3.999 |
##   [ 2] test ~ Aggression                |     5%   |   3.782 |   3.861 |   3.978 |
##   [ 3] test ~ poly(Aggression, 2)       |    95%   |   3.662 |   3.774 |   3.923 |
##   [ 4] test ~ poly(Aggression, 3)       |     0%   |   3.793 |   3.925 |   4.077 |
```

The quadratic model makes the best predictions. Let us see the best fitted model, with the squared term of the predictor:

```
p <- ggplot(agdat, aes(Aggression, test))
p + geom_point() + geom_smooth(method = "lm",
    formula = y ~ poly(x, 2))
```



Other loss function

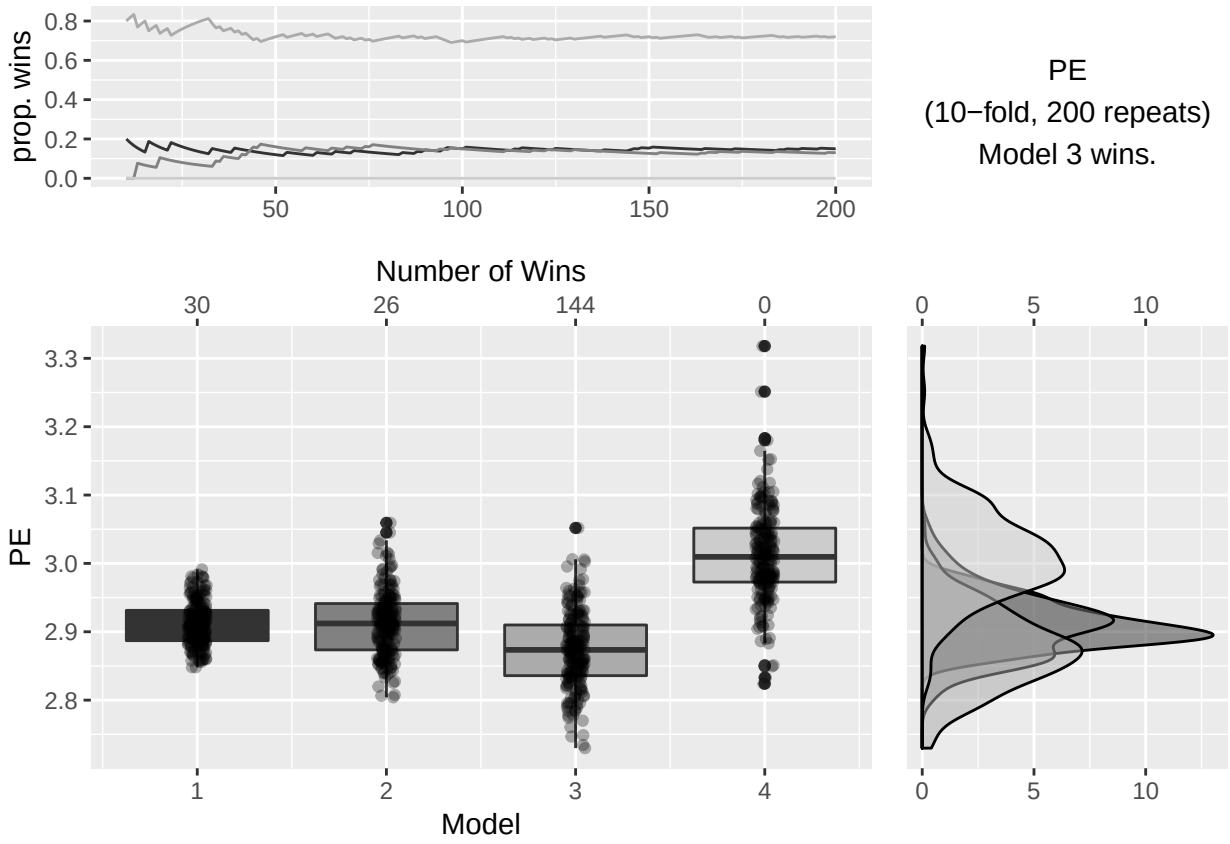
Here, instead of using the root mean squared error of prediction we look at the absolute value of the difference between the observed values and the predictions. Therefore we first have to define the loss function. This should be a function with two arguments, y and preds, the predictions.

```
abloss = function(y, preds) {
    mean(abs(y - preds))
}
```

Now we can run the four models from the previous section again, where prediction error is quantified in a different way, i.e. by the mean absolute error. We also make black and white plots.

```
output = xval.glm(data = agdat, models, loss = abloss,
    gray = TRUE)
```

```
## Running Cross-validation...done [ 9.3 sec ]
```

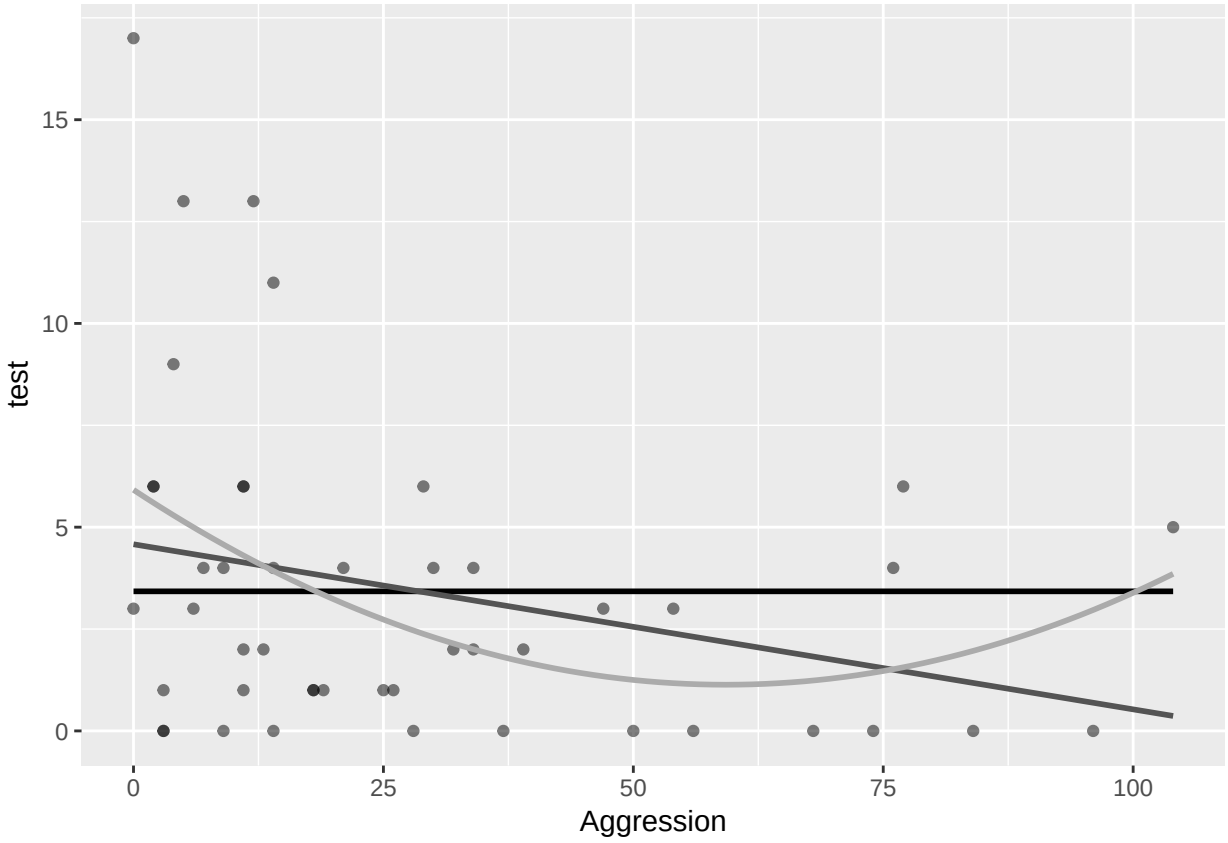



```
## Results for (10-fold, 200 repeats)
##   Model:                                |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] test ~ 1                         |    15%   |   2.859 |   2.911 |   2.979 |
##   [ 2] test ~ Aggression                |    13%   |   2.827 |   2.912 |   3.015 |
##   [ 3] test ~ poly(Aggression, 2)       |    72%   |   2.770 |   2.874 |   2.992 |
##   [ 4] test ~ poly(Aggression, 3)       |     0%   |   2.886 |   3.013 |   3.153 |
```

We see that the quadratic model still gives the best predictions. However, this model is not such a clear winner anymore. Evenmore, model 1 (the intercept only model), now outperforms model number 2 (the linear model). Looking at the plot of the different model estimates (using the intercept only, linear and quadratic model predictions), we can see why both models 1 and 2 seem to make approximately the same predicitions overall: model 2 is better in predicting high test scores for low aggression and low test scores for high aggression, while the intercept only model is better in predicting the low test scores for low aggression and the higher test scores for high aggression. Model 3 seems to do better overall, but still is relatively close to both models 1 and 2 in terms of prediction accuracy. Most likely, model 3 wins most of the time from model 2 due to the one participant with the highest aggression score (104) at the far right of the plot. But even with overlapping distributions, model 3 still has a lower prediction error in most repeats. This participant drives the winning of model 3 using the RMSEP more (since the prediction error is squared here) than in the analysis using the absolute loss function, hence explaining the difference in wins.

```
p <- ggplot(agdat, aes(Aggression, test))
p + geom_point(color = rgb(0, 0, 0, 0.5)) +
    geom_smooth(method = "lm", formula = y ~
        1, se = FALSE, color = gray(0)) +
    geom_smooth(method = "lm", formula = y ~
        poly(x, 1), se = FALSE, color = gray(0.33)) +
```

```
    geom_smooth(method = "lm", formula = y ~
        poly(x, 2), se = FALSE, color = gray(0.67))
```



Standard analysis approach

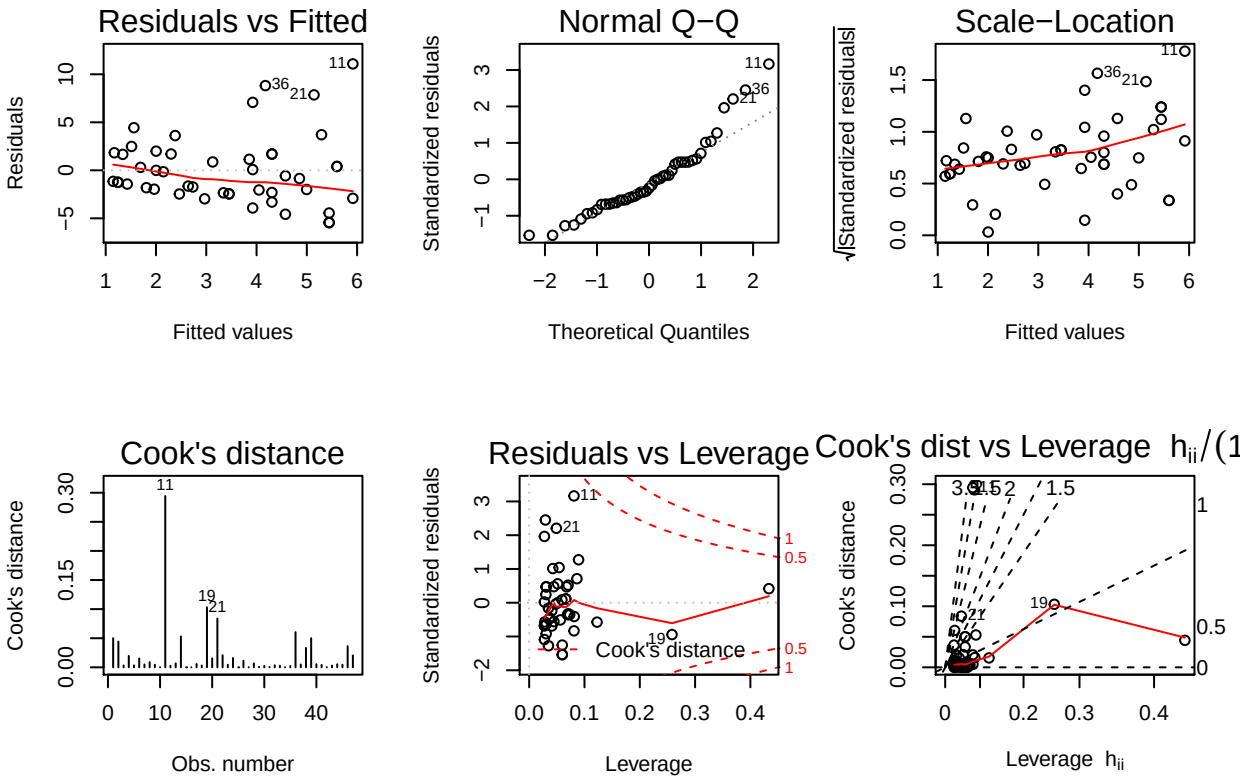
The standard analysis approach fits four rergession models and examines the change in R^2 and the corresponding incremental F-test to select a model. This approach assumes normally distributed error terms with zero mean and constant variance. In the following we show the results of the incremental F-tests and diagnostic plots considering the assumptions. As we can see the validity of the assumptions is questionable.

```
out1 = lm(models[[1]], data = agdat)
out2 = lm(models[[2]], data = agdat)
out3 = lm(models[[3]], data = agdat)
out4 = lm(models[[4]], data = agdat)
anova(out1, out2, out3, out4)
```

```
## Analysis of Variance Table
##
## Model 1: test ~ 1
## Model 2: test ~ Aggression
## Model 3: test ~ poly(Aggression, 2)
## Model 4: test ~ poly(Aggression, 3)
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     46 693.49
## 2     45 636.60  1    56.891 4.1844 0.04695 *
```

```
## 3     44 587.51  1    49.091 3.6107 0.06413 .
## 4     43 584.63  1     2.881 0.2119 0.64762
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mfrow = c(2, 3))
plot(out3, which = c(1, 2, 3, 4, 5, 6))
```



Moderated regression analysis

The data can be opbtained from: https://easy.dans.knaw.nl/ui/datasets/id/easy-dataset:100569.

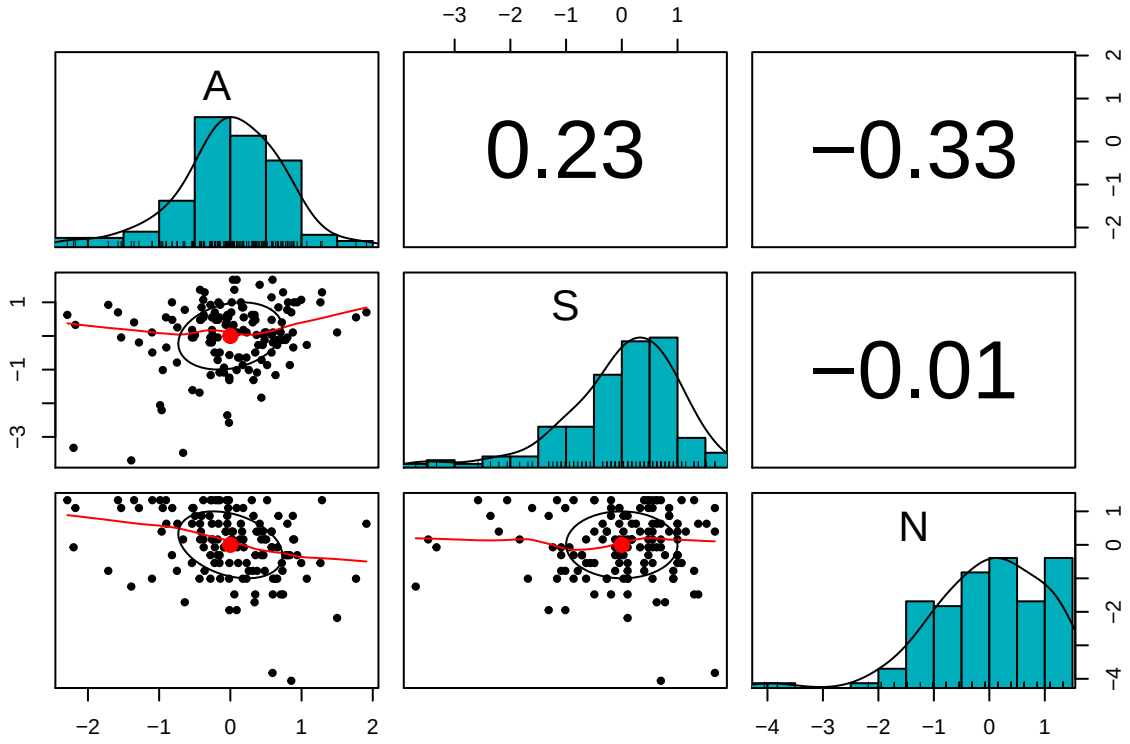
```
attidat = read.spss("~/surfdrive/Shared/CVtutorial/TMMSStudy1Data.sav",
    to.data.frame = TRUE, use.value.labels = F)
attidat = attidat[, c("wdefense", "ms", "Zsemean",
    "Zident")]
# rename varaibles A: positive Attitudes
# toward Muslims and multiculturalism
# (wdefense) M: mortality salience (ms)
# S: Self esteem (Zsemean) N: National
# identification (Zident)
colnames(attidat) = c("A", "M", "S", "N")

library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
pairs.panels(attidat[, -2], method = "pearson",
    hist.col = "#00AFBB", density = TRUE,
    ellipses = TRUE)
```



We see in these graphs that there are 5 subjects which have quite extreme scores on the Self esteem and National identification variables, with standardized scores far beyond 3. We will analyze the data with and without these subjects removed. Therefore we create a reduced data frame with these observations removed.

```
attidat.r = attidat[-c(3, 14, 39, 59, 138),
    ]  # four observations with extremely large observations
```

Original analysis

The authors report the following analysis

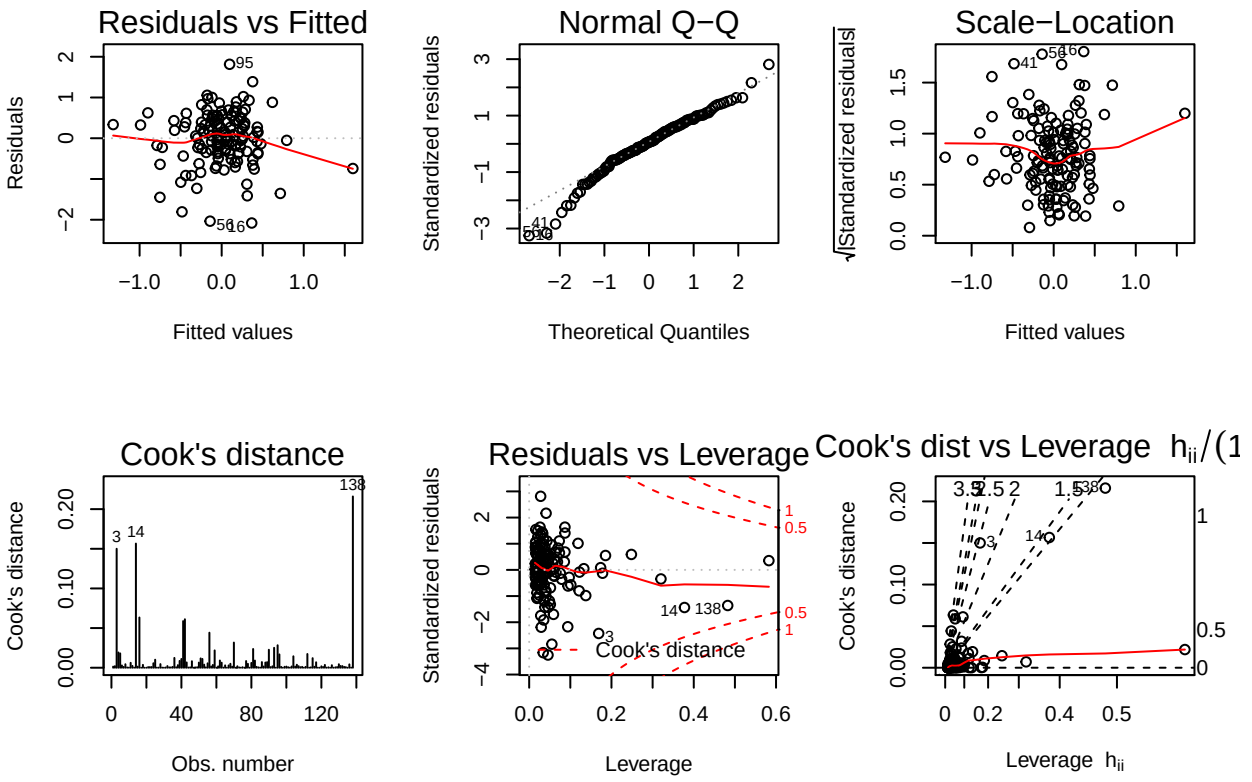
```
lm.out = lm(A ~ M * S * N, data = attidat)
summary(lm.out)
```

```
##
## Call:
## lm(formula = A ~ M * S * N, data = attidat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -2.08034 -0.32984  0.03915  0.42036  1.81501
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.004095   0.055851   0.073  0.94167
## M            0.018339   0.055851   0.328  0.74317
## S            0.150351   0.056193   2.676  0.00842 **
## N           -0.267092   0.058299  -4.581 1.07e-05 ***
## M:S          0.100792   0.056193   1.794  0.07519 .
## M:N         -0.011083   0.058299  -0.190  0.84952
## S:N         -0.011459   0.054073  -0.212  0.83250
## M:S:N        0.177794   0.054073   3.288  0.00130 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6546 on 130 degrees of freedom
## Multiple R-squared:  0.2398, Adjusted R-squared:  0.1989
## F-statistic: 5.859 on 7 and 130 DF,  p-value: 6.326e-06
```

Diagnostic plots for this analysis

```
par(mfrow = c(2, 3))
plot(lm.out, which = c(1, 2, 3, 4, 5, 6))
```



If we remove the five outliers from the data we obtain the following

```
lm.out = lm(A ~ M * S * N, data = attidat.r)
summary(lm.out)
```

```
##
## Call:
## lm(formula = A ~ M * S * N, data = attidat.r)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.12071 -0.32376 -0.00135  0.43379  1.83249
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03057    0.05623   0.544  0.58759
## M            0.01253    0.05623   0.223  0.82400
## S            0.08039    0.06752   1.191  0.23610
## N           -0.30339    0.06374  -4.760 5.26e-06 ***
## M:S          0.07405    0.06752   1.097  0.27491
## M:N          0.03319    0.06374   0.521  0.60353
## S:N          0.02651    0.07095   0.374  0.70924
## M:S:N        0.19557    0.07095   2.757  0.00672 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6423 on 125 degrees of freedom
## Multiple R-squared:  0.2054, Adjusted R-squared:  0.1609
## F-statistic: 4.617 on 7 and 125 DF,  p-value: 0.0001272
```

Cross validation analysis

First define the 15 models of interest.

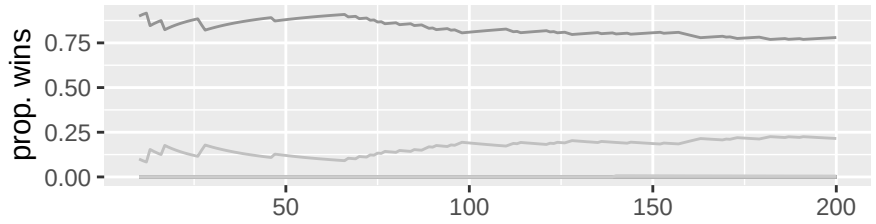
```
models = vector(mode = "list", length = 15)
models[[1]] = A ~ 1
models[[2]] = A ~ M
models[[3]] = A ~ N
models[[4]] = A ~ S
models[[5]] = A ~ M + S
models[[6]] = A ~ M + N
models[[7]] = A ~ S + N
models[[8]] = A ~ M + S + N
models[[9]] = A ~ M * S + N
models[[10]] = A ~ M + S * N
models[[11]] = A ~ M * N + S
models[[12]] = A ~ M * N + M * S
models[[13]] = A ~ M * N + N * S
models[[14]] = A ~ M * S + N * S
models[[15]] = A ~ M * S + N * S + M * N
models[[16]] = A ~ M * N * S
```

First the analysis on the complete data

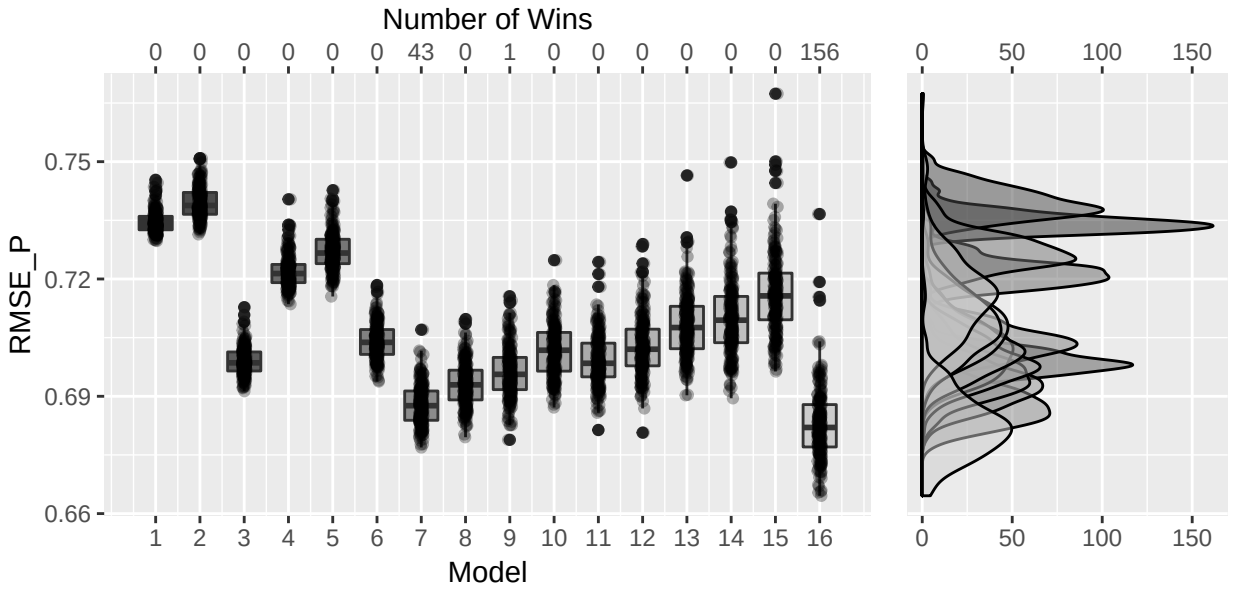
```
output = xval.glm(data = attidat, models,
    numCore = 14, gray = TRUE, seed = 123)
```

```
## Using 14 cores.
```

```
## Running Cross-validation...done [ 5.8 sec ]
```



RMSE_P
(10−fold, 200 repeats)
Model 16 wins.

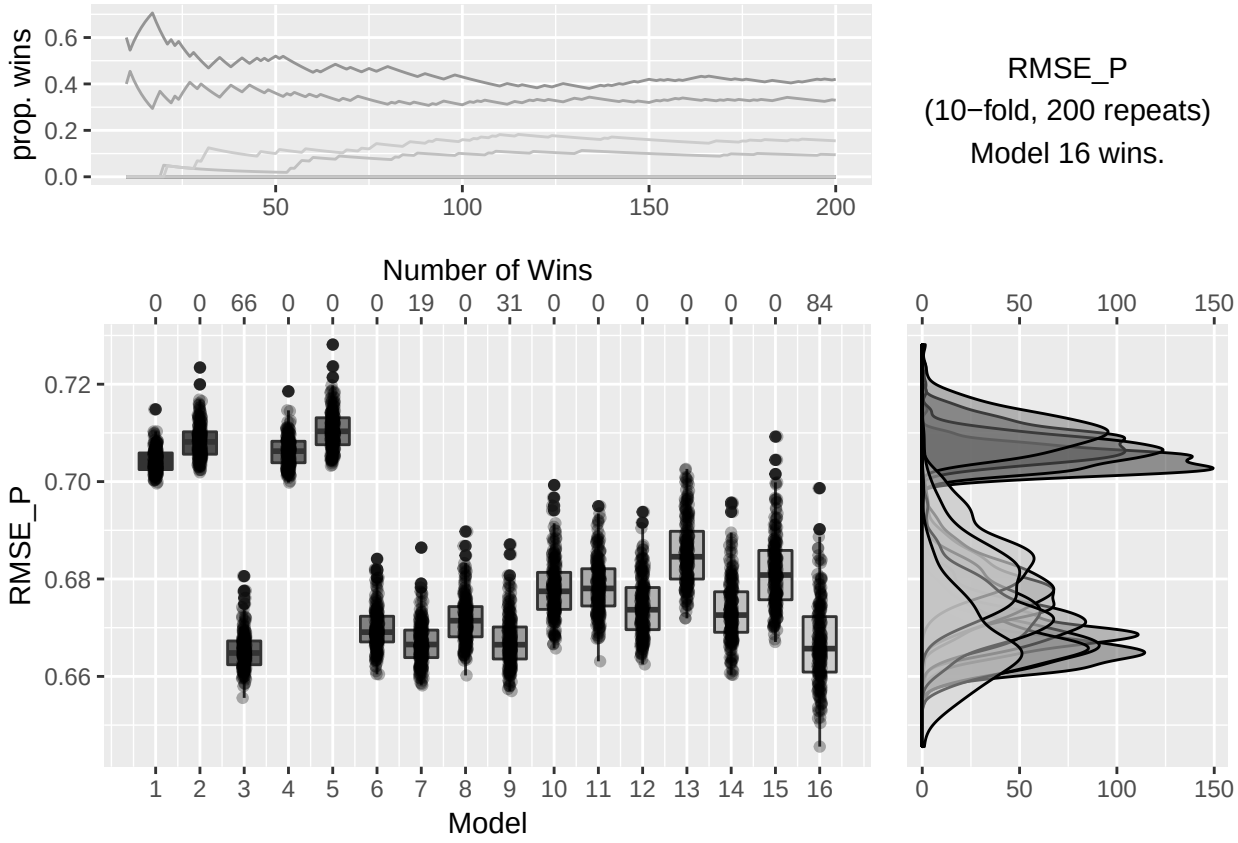



```
## Results for (10-fold, 200 repeats)
##   Model:                                |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] A ~ 1                            |     0%   |   0.730 |   0.735 |   0.742 |
##   [ 2] A ~ M                            |     0%   |   0.733 |   0.740 |   0.748 |
##   [ 3] A ~ N                            |     0%   |   0.693 |   0.699 |   0.707 |
##   [ 4] A ~ S                            |     0%   |   0.716 |   0.722 |   0.731 |
##   [ 5] A ~ M + S                        |     0%   |   0.719 |   0.727 |   0.738 |
##   [ 6] A ~ M + N                        |     0%   |   0.696 |   0.704 |   0.714 |
##   [ 7] A ~ S + N                        |    22%   |   0.679 |   0.688 |   0.698 |
##   [ 8] A ~ M + S + N                    |     0%   |   0.683 |   0.693 |   0.706 |
##   [ 9] A ~ M * S + N                    |     0%   |   0.684 |   0.696 |   0.711 |
##   [10] A ~ M + S * N                    |     0%   |   0.690 |   0.702 |   0.717 |
##   [11] A ~ M * N + S                    |     0%   |   0.688 |   0.699 |   0.712 |
##   [12] A ~ M * N + M * S                |     0%   |   0.690 |   0.703 |   0.721 |
##   [13] A ~ M * N + N * S                |     0%   |   0.695 |   0.708 |   0.726 |
##   [14] A ~ M * S + N * S                |     0%   |   0.695 |   0.710 |   0.733 |
##   [15] A ~ M * S + N * S + M * N        |     0%   |   0.699 |   0.716 |   0.745 |
##   [16] A ~ M * N * S                    |    78%   |   0.667 |   0.683 |   0.704 |
```

and a similar analysis on the data with the five persons removed

```
output = xval.glm(data = attidat.r, models,
    numCore = 14, gray = TRUE, seed = 123)
```

```
## Using 14 cores.
## Running Cross-validation...done [ 5.7 sec ]
```




```
## Results for (10-fold, 200 repeats)
##   Model:                                |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] A ~ 1                            |     0%   |   0.700 |   0.704 |   0.709 |
##   [ 2] A ~ M                            |     0%   |   0.703 |   0.708 |   0.716 |
##   [ 3] A ~ N                            |    33%   |   0.660 |   0.665 |   0.674 |
##   [ 4] A ~ S                            |     0%   |   0.701 |   0.706 |   0.712 |
##   [ 5] A ~ M + S                        |     0%   |   0.704 |   0.711 |   0.719 |
##   [ 6] A ~ M + N                        |     0%   |   0.663 |   0.670 |   0.680 |
##   [ 7] A ~ S + N                        |    10%   |   0.660 |   0.667 |   0.677 |
##   [ 8] A ~ M + S + N                    |     0%   |   0.664 |   0.672 |   0.682 |
##   [ 9] A ~ M * S + N                    |    16%   |   0.659 |   0.667 |   0.679 |
##   [10] A ~ M + S * N                    |     0%   |   0.668 |   0.678 |   0.691 |
##   [11] A ~ M * N + S                    |     0%   |   0.668 |   0.679 |   0.692 |
##   [12] A ~ M * N + M * S                |     0%   |   0.665 |   0.674 |   0.686 |
##   [13] A ~ M * N + N * S                |     0%   |   0.675 |   0.686 |   0.701 |
##   [14] A ~ M * S + N * S                |     0%   |   0.663 |   0.673 |   0.688 |
##   [15] A ~ M * S + N * S + M * N        |     0%   |   0.670 |   0.681 |   0.699 |
##   [16] A ~ M * N * S                    |    42%   |   0.653 |   0.667 |   0.685 |
```

```
ggsave("~/surfdrive/Shared/CVtutorial/manuscript/attitude.pdf",
    output$box.plot)
```

```
## Saving 6.5 x 4.5 in image
```

Standard analysis approach

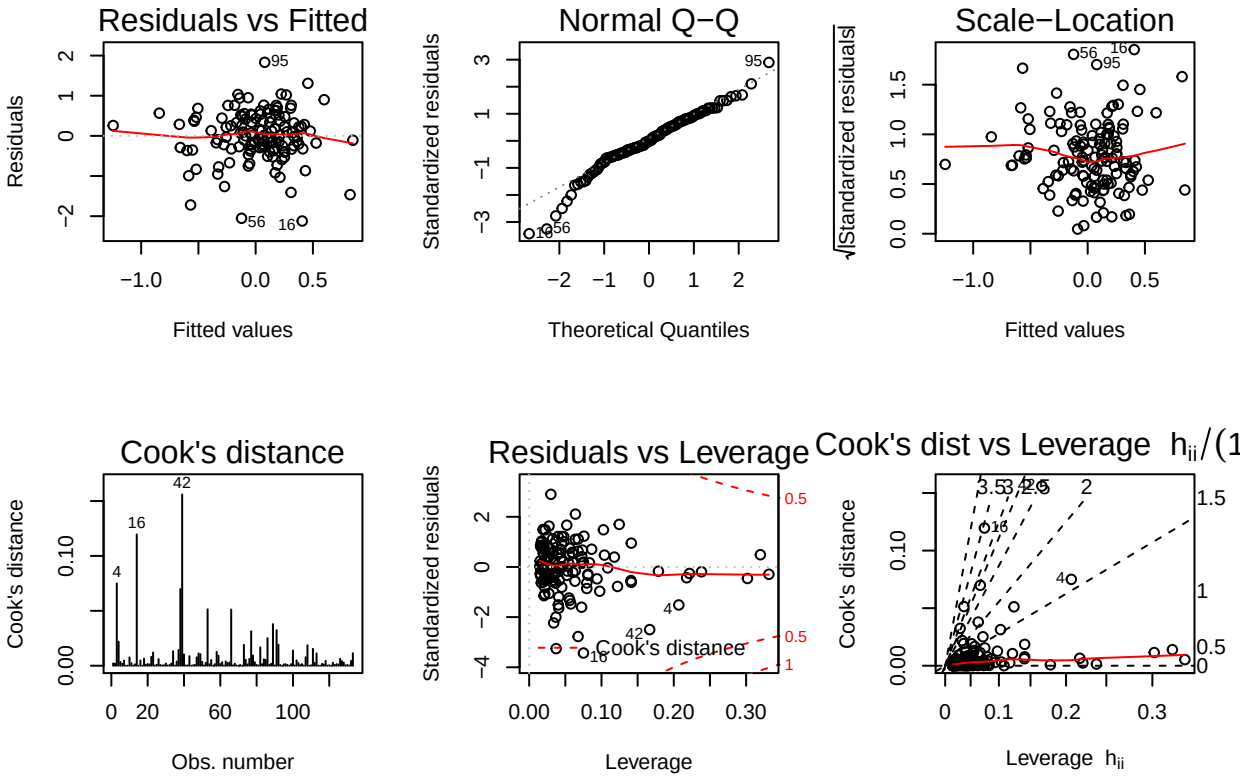
```
out1 = lm(models[[1]], data = attidat.r)
out2 = lm(models[[2]], data = attidat.r)
out3 = lm(models[[3]], data = attidat.r)
out4 = lm(models[[4]], data = attidat.r)
out5 = lm(models[[5]], data = attidat.r)
out6 = lm(models[[6]], data = attidat.r)
out7 = lm(models[[7]], data = attidat.r)
out8 = lm(models[[8]], data = attidat.r)
out9 = lm(models[[9]], data = attidat.r)
out10 = lm(models[[10]], data = attidat.r)
out11 = lm(models[[11]], data = attidat.r)
out12 = lm(models[[12]], data = attidat.r)
out13 = lm(models[[13]], data = attidat.r)
out14 = lm(models[[14]], data = attidat.r)
out15 = lm(models[[15]], data = attidat.r)
out16 = lm(models[[16]], data = attidat.r)
aovtab = anova(out1, out2, out3, out4, out5,
    out6, out7, out8, out9, out10, out11,
    out12, out13, out14, out15, out16)
```

A stepwise comparison of the models is given in the following table. In the column entitled "Against" we describe against which model the current model is compared.

Model	Res.Df	RSS	Against	Df	Sum of Sq	F	Pr(>F)
1	132	64.90					
2	131	64.66	1	1	0.25	0.50	0.4793
3	131	56.76	1	1	8.14	18.79	0.0000
4	131	64.50	1	1	0.41	0.83	0.3647
5	130	64.27	4	1	0.22	0.45	0.5029
6	130	56.65	3	1	0.11	0.26	0.6142
7	130	56.24	3	1	0.52	1.21	0.2728
8	129	56.14	7	1	0.09	0.21	0.6473
9	128	54.81	8	1	1.33	3.11	0.0802
10	128	56.14	8	1	0.00	0.01	0.9358
11	128	56.02	8	1	0.12	0.28	0.5950
12	127	54.71	9	1	0.10	0.23	0.6306
13	127	56.02	11	1	0.00	0.00	0.9780
14	127	54.80	9	1	0.01	0.03	0.8559
15	126	54.71	12	1	0.01	0.02	0.8942
16	125	51.57	15	1	3.13	7.60	0.0067

To verify the assumptions we make some diagnostic plots of the most complex model. Especially the QQ plot raises some concern anout the distributional assumptions of the residuals. If this assumpiton is not tenable the relability of the incremental Ftests and corresponding p-vales is questionable.

```
par(mfrow = c(2, 3))
plot(out16, which = c(1, 2, 3, 4, 5, 6))
```



Logistic Regression

Hastie and Tibsirani (1990) report data on the presence or absence of kyphosis, a postoperative spinal deformity. The predictor variable is the age of the patient in months. The data are available in the `gam`-package, where the response variable is a string variable. We first recode the response variable to a 0,1 variable, where 1 indicates presence of kyphosis. Then we fit three logistic regression models, a intercept only, a logistic regression with Age as predictor, and a logistic regression with Age and Age-squared as predictors.

In this case, where we have outcomes equal to zero or one and predicted values are probabilities, the squared loss equals the Brier Score.

Two hundred repeats of 10-fold cross validation are performed using the following code.

```
library(gam)
```

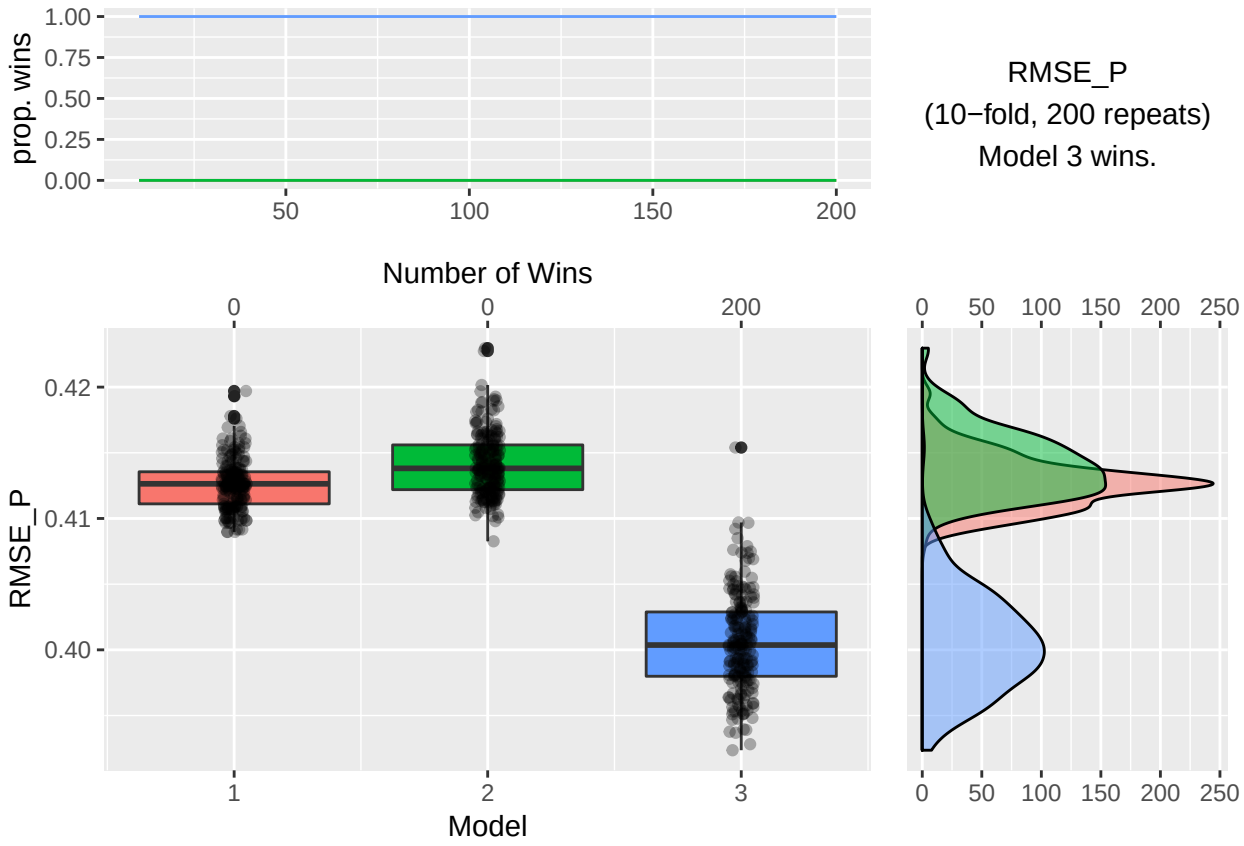
```
## Loading required package: splines
```

```
## Loaded gam 1.16.1
```

```
data(kyphosis)
kyphosis[, 1] = as.numeric(kyphosis[, 1] ==
    "present")
models = vector(mode = "list", length = 3)
models[[1]] = Kyphosis ~ 1
models[[2]] = Kyphosis ~ Age
models[[3]] = Kyphosis ~ poly(Age, 2)
```

```
output = xval.glm(data = kyphosis, models,
    glm.family = binomial)
```

```
## Running Cross-validation...done [ 8.3 sec ]
```




```
## Results for (10-fold, 200 repeats)
##   Model:                                |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] Kyphosis ~ 1                     |     0%   |   0.409 |   0.413 |   0.417 |
##   [ 2] Kyphosis ~ Age                   |     0%   |   0.410 |   0.414 |   0.419 |
##   [ 3] Kyphosis ~ poly(Age, 2)          |   100%   |   0.394 |   0.401 |   0.408 |
```

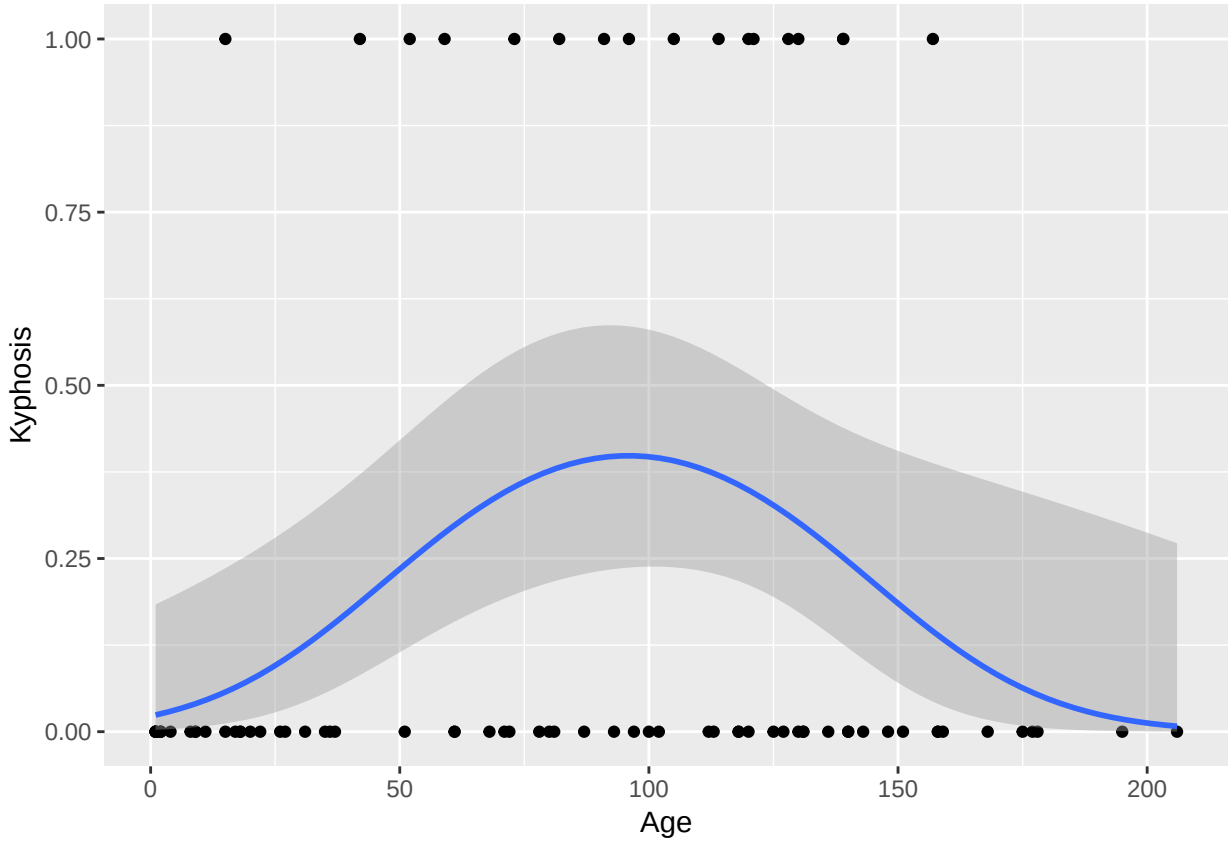
The quadratic model has a much lower prediction error and wins in all 100 repetitions. Let us interpret the model as follows

```
out = glm(models[[3]], data = kyphosis, family = binomial)
summary(out)
```

```
##
## Call:
## glm(formula = models[[3]], family = binomial, data = kyphosis)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0079  -0.8412  -0.4155  -0.2209   2.3920
```

```
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.6846     0.3822  -4.408 1.04e-05 ***
## poly(Age, 2)1   4.0616     3.6673   1.108   0.2681
## poly(Age, 2)2  -9.8118     3.9714  -2.471   0.0135 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 83.234  on 80  degrees of freedom
## Residual deviance: 72.739  on 78  degrees of freedom
## AIC: 78.739
##
## Number of Fisher Scoring iterations: 5
```

```
p <- ggplot(kyphosis, aes(Age, Kyphosis))
p + geom_point() + geom_smooth(method = "glm",
    method.args = list(family = "binomial"),
    formula = y ~ poly(x, 2))
```



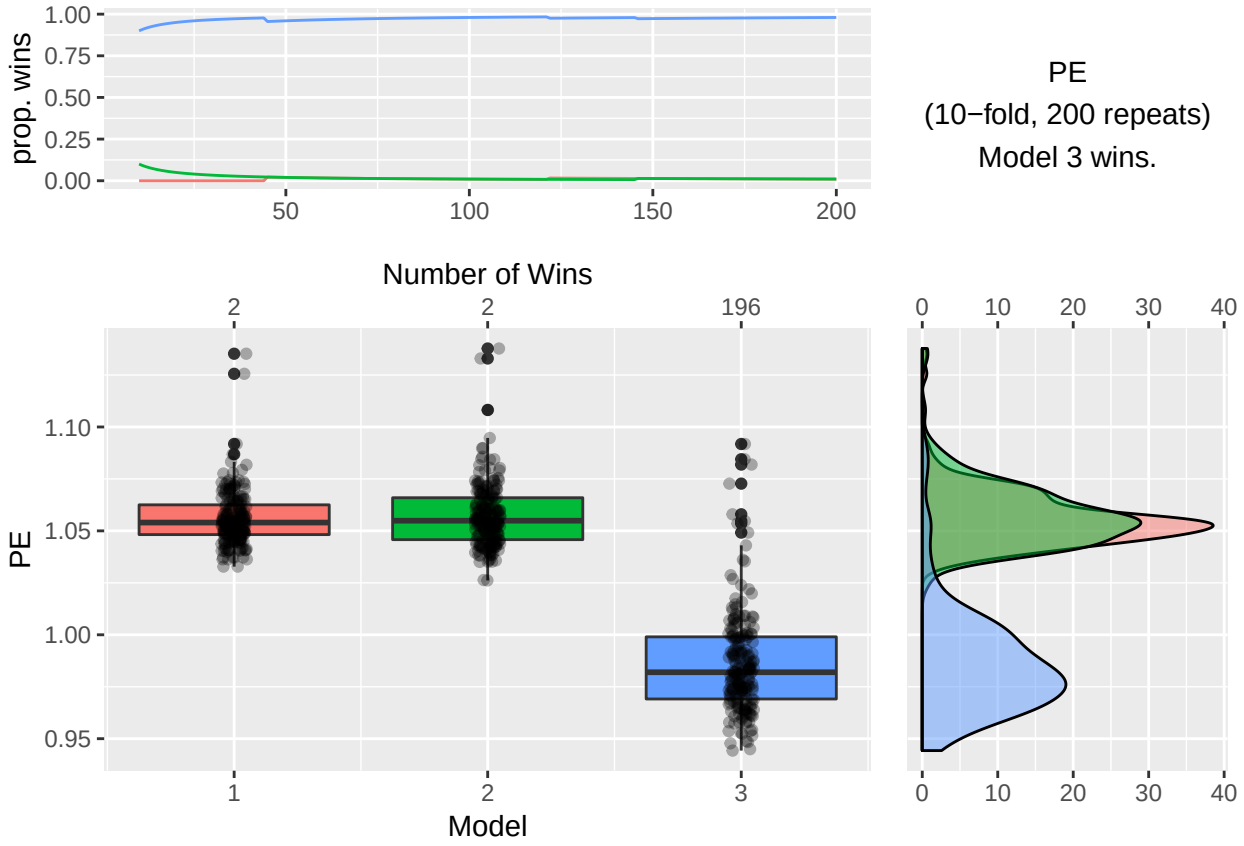
We see that this is a single peaked curve, first the probability goes up, later it goes down. Nowhere the probability becomes larger than 0.5, so for every person we would predict the absence of kyphosis. Around the age of 100 months there is a probability of about 40 percent that a patient has kyphosis. On the basis of age alone, we can however, not tell who that will be.

Changing the loss function

We will now use two different prediction loss functions. The first is the cross validated deviance, the second the misclassification rate.

```
dev = function(y, preds) {
    -2 * mean(y * log(preds) + (1 - y) *
        log(1 - preds))
}
output = xval.glm(data = kyphosis, models,
    glm.family = binomial, loss = dev)
```

```
## Running Cross-validation...done [ 8.5 sec ]
```



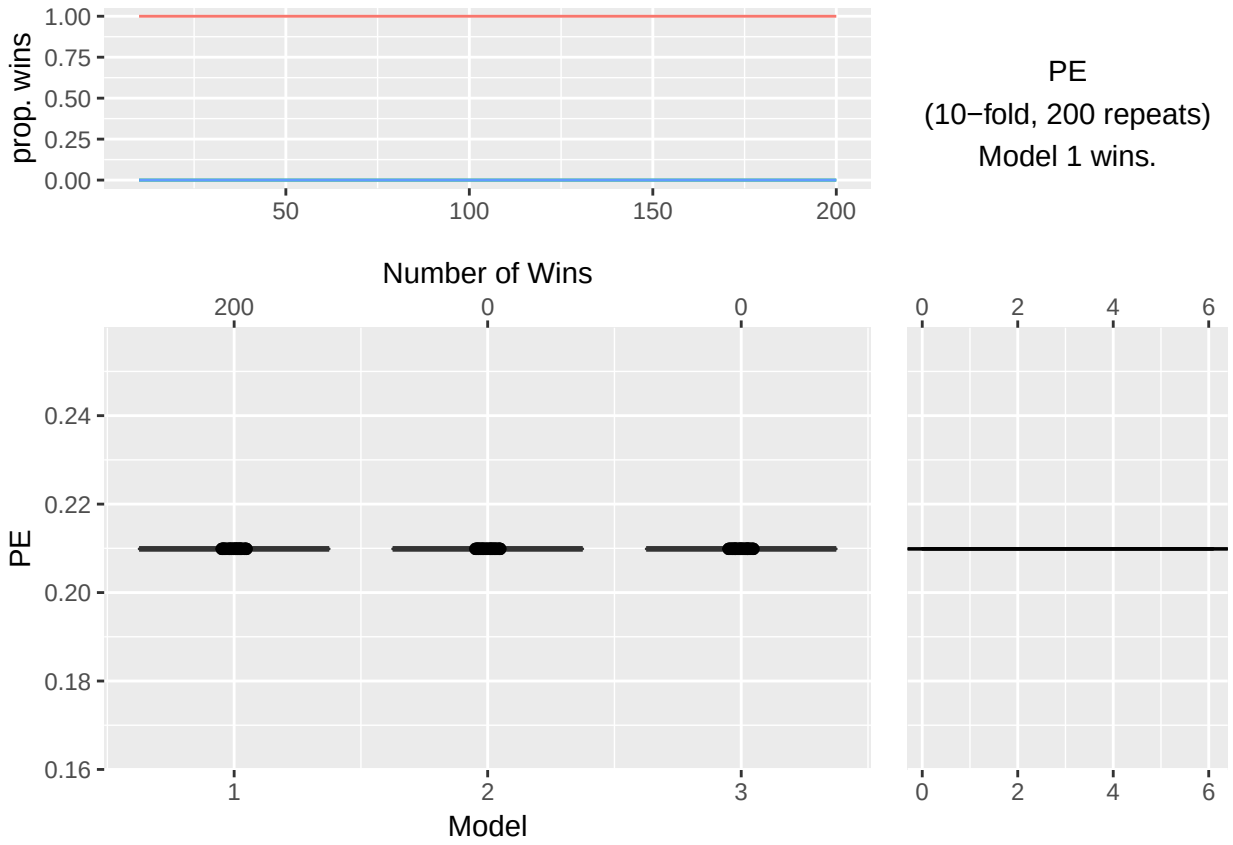

```
## Results for (10-fold, 200 repeats)
##   Model:                                |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] Kyphosis ~ 1                     |     1%   |   1.037 |   1.056 |   1.082 |
##   [ 2] Kyphosis ~ Age                   |     1%   |   1.036 |   1.057 |   1.090 |
##   [ 3] Kyphosis ~ poly(Age, 2)          |    98%   |   0.951 |   0.986 |   1.055 |
```

The results are roughly the same as with the Brier score (RMSEP).

Another often used loss function is the missclasification rate. In the following we first define this function and then use it in the cross validation.

```
mcr = function(y, preds) {
    1 - mean(y * (preds > 0.5) + (1 - y) *
        (preds < 0.5))
}
output = xval.glm(data = kyphosis, models,
    glm.family = binomial, loss = mcr)
```

```
## Running Cross-validation...done [ 8.4 sec ]
```




```
## Results for (10-fold, 200 repeats)
##   Model:                                  |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] Kyphosis ~ 1                       |   100%   |   0.210 |   0.210 |   0.210 |
##   [ 2] Kyphosis ~ Age                     |     0%   |   0.210 |   0.210 |   0.210 |
##   [ 3] Kyphosis ~ poly(Age, 2)            |     0%   |   0.210 |   0.210 |   0.210 |
```

We see that in this case the intercept only model performs best. Although most of the time all three models predict for every case absence of kyphosis in all repeats of all folds. These examples clearly show that the choice of loss function has a large impact on the results.

Standard analysis approach

```
out1 = glm(models[[1]], data = kyphosis,
    family = binomial)
out2 = glm(models[[2]], data = kyphosis,
    family = binomial)
out3 = glm(models[[3]], data = kyphosis,
    family = binomial)
anova(out1, out2, out3)
```

```
## Analysis of Deviance Table
##
## Model 1: Kyphosis ~ 1
## Model 2: Kyphosis ~ Age
## Model 3: Kyphosis ~ poly(Age, 2)
##   Resid. Df Resid. Dev Df Deviance
## 1        80     83.234
## 2        79     81.932  1   1.3020
## 3        78     72.739  1   9.1939
```

Comparing two theories

In Pollack et al. (2012) the authors investigate the effect of economic stress on intentions to disengage from entrepreneural activities. The participants in this study were 262 entrepreneurs who were members of a networking group for small-business owners, who responded to an online survey about recent performance of their business, and their emotional and cognitive responses to the economic climate.

The participants were asked a series of questions about how they felt their business was doing. Their responses were used to create an index of economic stress (**estress**, higher scores reflecting greater stress)

They were asked the extent to which they had various feelings related to their business, such as discouraged, hopeless, worthless, and the like, an aggregation of which was used to quantify business-related depressed affect (**affect**, higher scores reflecting more depressed affect).

Another measure is entrepreneurial self-efficacy: this measure indexes a person's confidence in his or her ability to succesfully engage in various entrepreneurship-related tasks such as setting and meeting goals, creating new products, managing risk, and making decisions (**ese**).

Finally, they were also asked a set of questions to quantify their intentions to withdraw from entrepreneuship in the next year (**withdraw**, higher scores indicative for greater withdrawal intentions). Moreover, we have a set of covariates: **sex** (0 = female; 1 = male), **age** in years, and **tenure** (length of time in business).

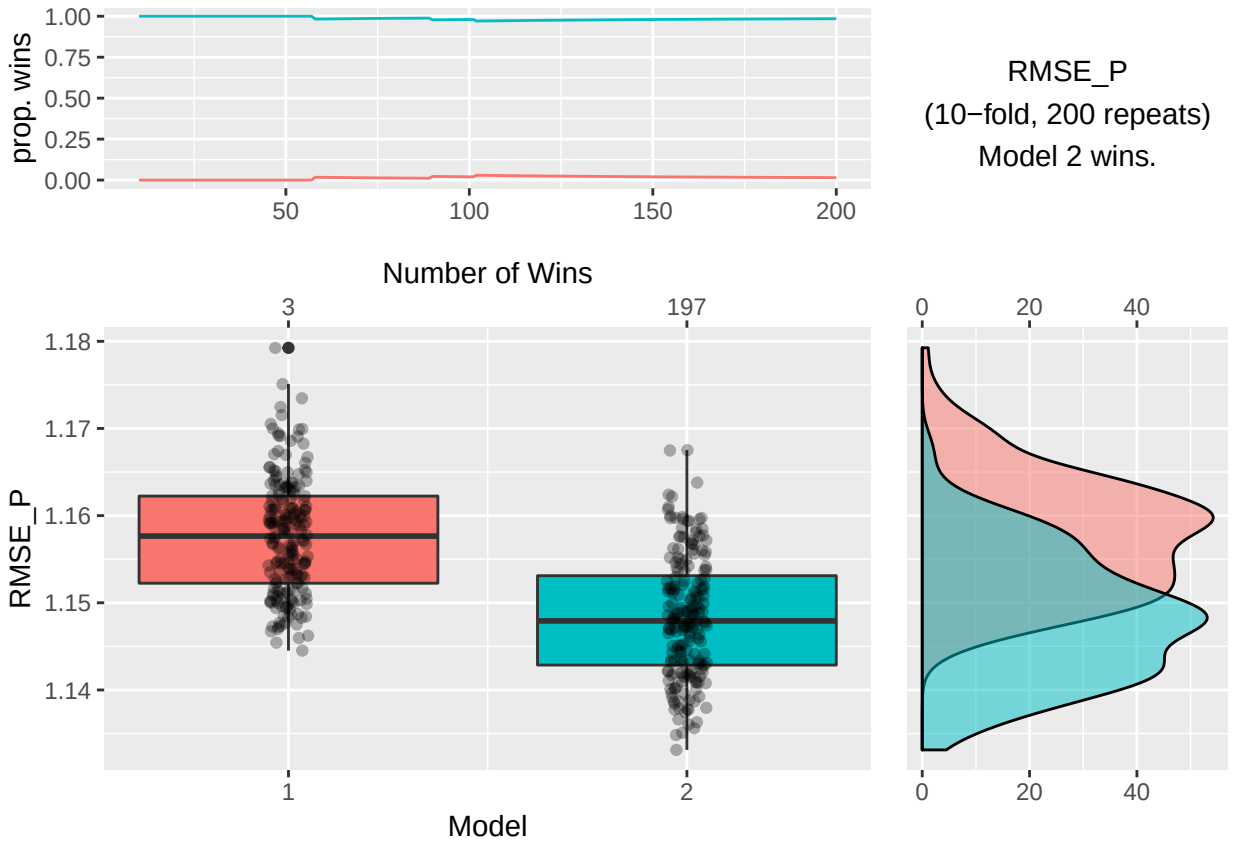
For these data there are two theories: - The first theory says that economic stress has an influence on withdrawal intentions but that this effect is mediated by business-related depressed affect. Taking the covariates into account this leads to a regression model with **withdraw** as response and **estress**, **affect**, **sex**, **age**, and **tenure** as predictors. - A second theory is that economic stress is not at all related to withdrawal intentions and that withdrawal intentions are just an effect of individual differences. That is, more confident persons have less depressed affect and therefore less intentions to withdraw. Taking the covariates into account this leads to a regression model with **withdraw** as response and **ese**, **affect**, **sex**, **age**, and **tenure** as predictors.

```
library(foreign)
ecdata = read.spss("~/surfdrive/Shared/CVtutorial/estress.sav",
    to.data.frame = TRUE)
```

```
## re-encoding from CP1252
```

```
models = vector(mode = "list", length = 2)
models[[1]] = withdraw ~ tenure + estress +
    affect + sex + age
models[[2]] = withdraw ~ tenure + affect +
    sex + age + ese
output = xval.glm(data = ecdata, models)
```

```
## Running Cross-validation...done [ 8.7 sec ]
```




```
## Results for (10-fold, 200 repeats)
##   Model:                                          |   Wins   |    2.5% |    mean |   97.5% |
##   [ 1] withdraw ~ tenure + estress + affect + s   |     2%   |   1.147 |   1.158 |   1.171 |
##   [ 2] withdraw ~ tenure + affect + sex + age +   |    98%   |   1.136 |   1.148 |   1.161 |
```

Standard analysis approach

This is a more difficult situation because the two models are not nested. Therefore, a comparison using statistical tests is not possible. Researchers could use an information criterion like the AIC or BIC.

The output object

The `xval.glm` function outputs an object with the following ingredients:

```
names(output)
```

```
##  [1] "models"     "glms"       "data"       "stab.plot"  "box.plot"
##  [6] "den.plot"   "win.matrix" "wins"       "summary"    "RMSEP"
```

In the following we will show one by one these for the analysis in the last section.

```
output$models
```

```
## [[1]]
## withdraw ~ tenure + estress + affect + sex + age
##
## [[2]]
## withdraw ~ tenure + affect + sex + age + ese
```

shows the models that were selected by the researcher to compare.

```
output$glms
```

```
## [[1]]
##
## Call:  glm(formula = models[[m]], family = glm.family, data = data)
##
## Coefficients:
## (Intercept)       tenure      estress       affect      sexmale
##    1.529987     0.001371    -0.080671     0.768182     0.112486
##         age
##   -0.003229
##
## Degrees of Freedom: 261 Total (i.e. Null);  256 Residual
## Null Deviance:       405.8
## Residual Deviance: 331.6     AIC: 819.3
##
## [[2]]
##
## Call:  glm(formula = models[[m]], family = glm.family, data = data)
##
## Coefficients:
## (Intercept)       tenure       affect      sexmale          age
##    2.609127    -0.001681     0.646385     0.103911    -0.005110
##         ese
##   -0.205351
##
## Degrees of Freedom: 261 Total (i.e. Null);  256 Residual
## Null Deviance:       405.8
## Residual Deviance: 325.5     AIC: 814.4
```

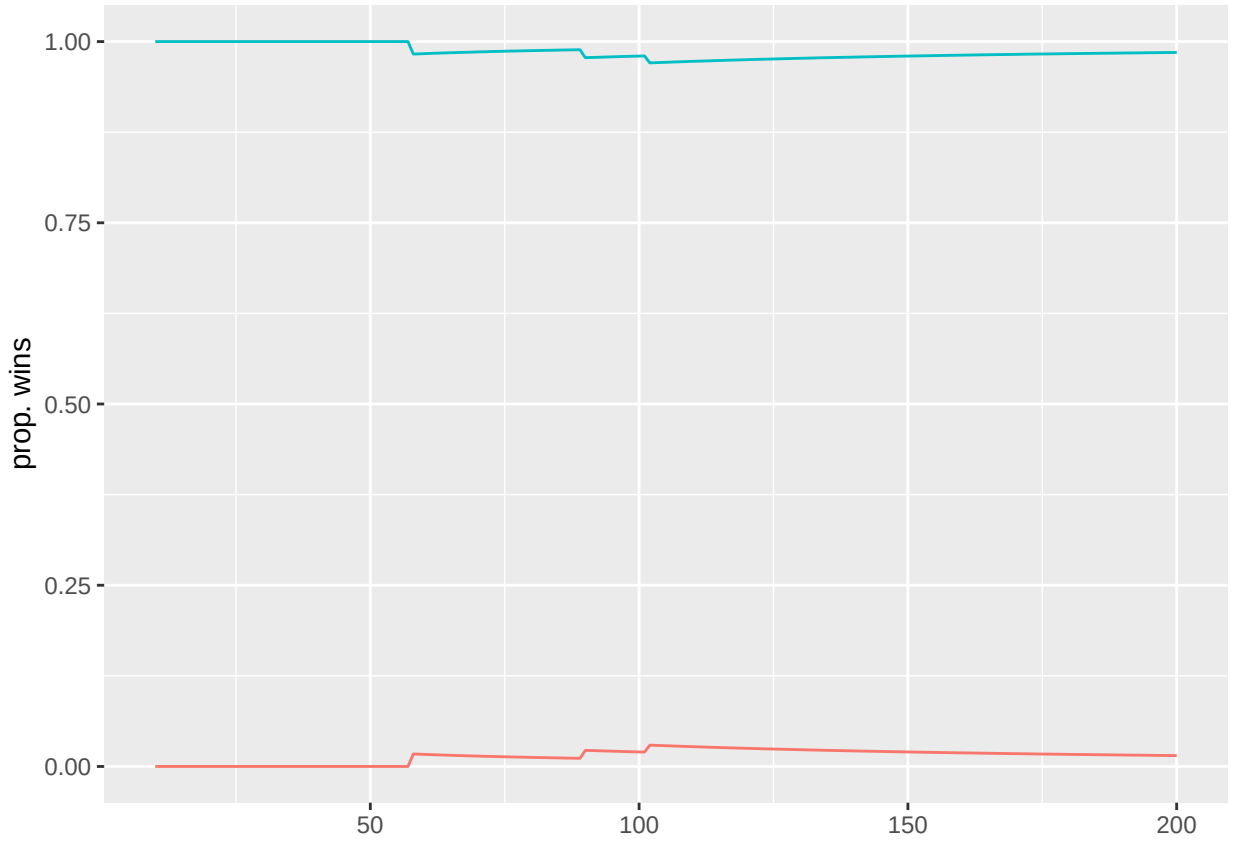
Shows the standard glm output for each of the requested models on the complete data set.

```
head(output$data)
```

```
##   tenure estress affect withdraw    sex age  ese
## 1   1.67     6.0   2.60     3.00   male  51 5.33
## 2   0.58     5.0   1.00     1.00 female  45 6.05
## 3   0.58     5.5   2.40     3.66   male  42 5.26
## 4   2.00     3.0   1.16     4.66   male  50 4.35
## 5   5.00     4.5   1.00     4.33   male  48 4.86
## 6   9.00     6.0   1.50     3.00   male  48 5.05
```

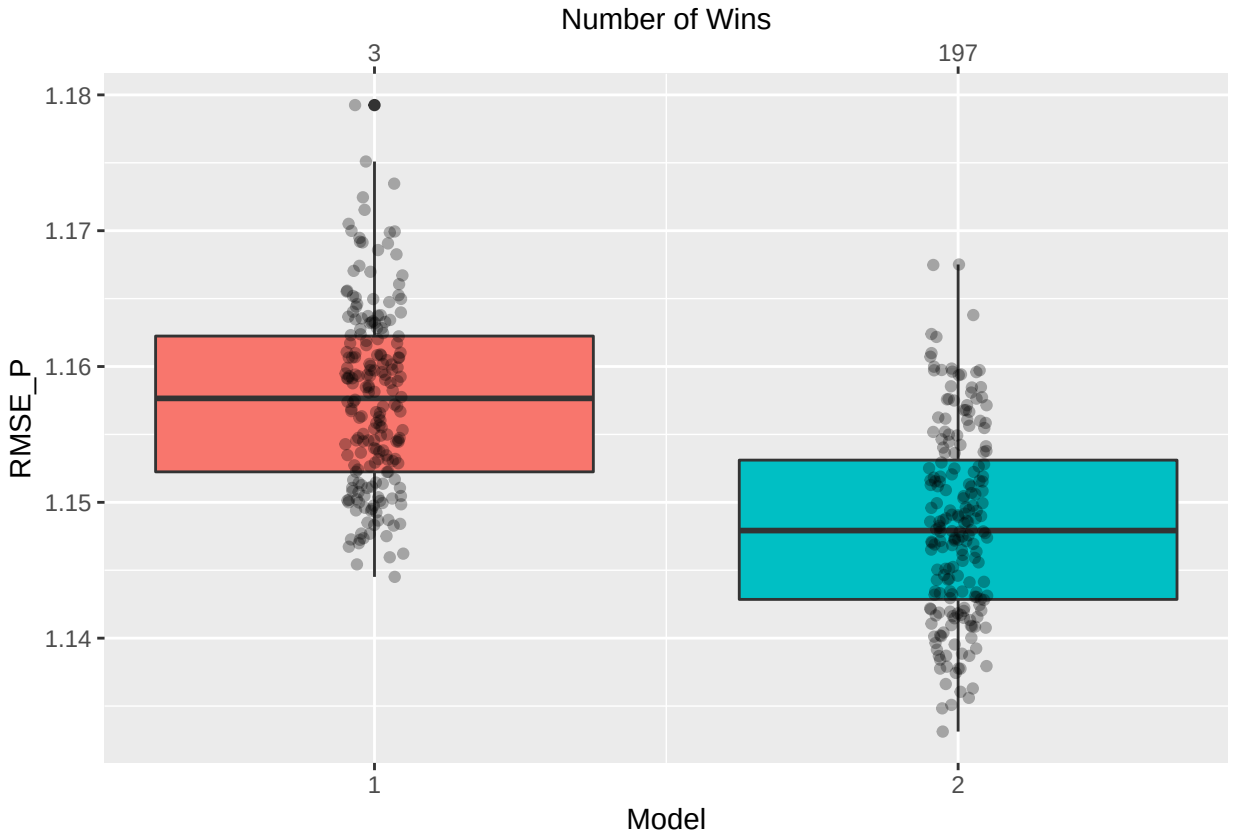
Gives the input data set.

```
output$stab.plot
```



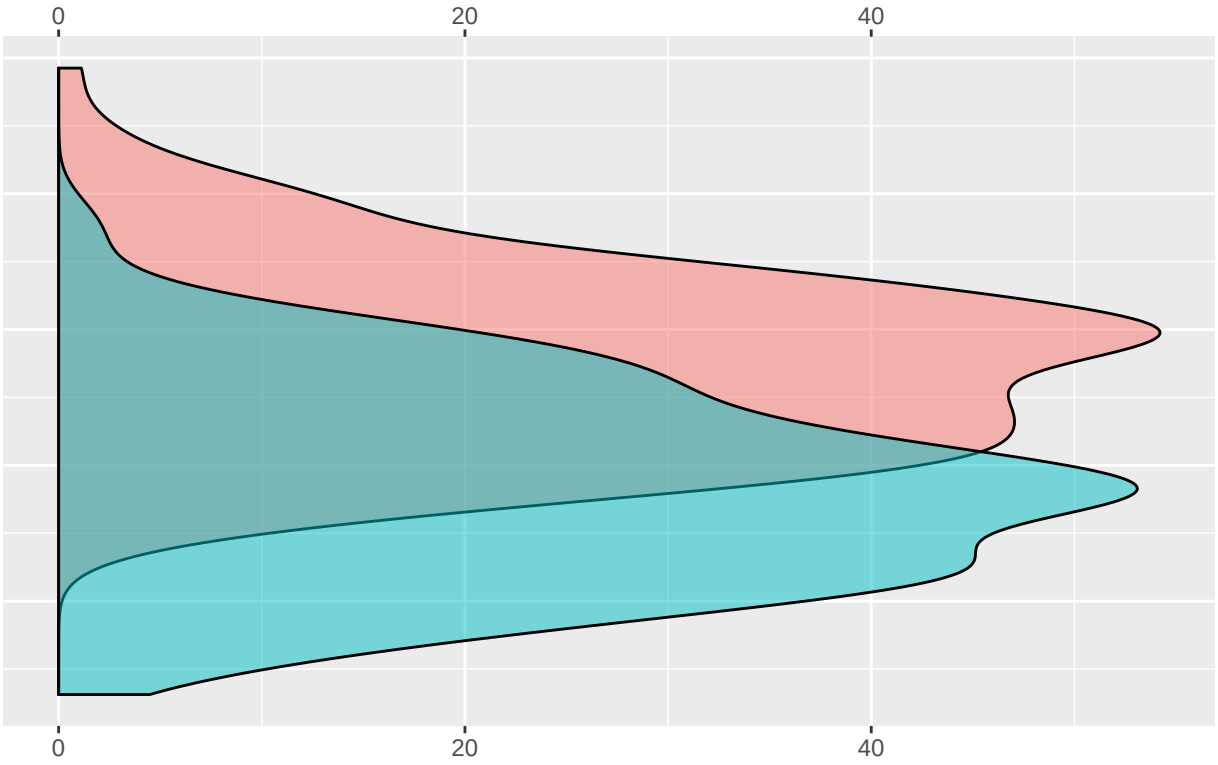
Shows the proportion of wins of each model over the repeated cross validations. Can be used in order to check whether more repetitions are needed for the cross validation.

```
output$box.plot
```



Gives the boxplot of the prediction error for each model over the 200 repetitions.

```
output$den.plot
```



Gives the density plot of the prediction error for each model.

```
head(output$win.matrix)
```

```
##      [,1] [,2]
## [1,]    0    1
## [2,]    0    1
## [3,]    0    1
## [4,]    0    1
## [5,]    0    1
## [6,]    0    1
```

Shows for each cycle of cross validation which model had the lowest prediction error

```
output$wins
```

```
## [1]   3 197
```

Shows the total number of wins for each model.

```
output$summary
```

```
## [1] "Results for (10-fold, 200 repeats)\n"
## [2] " Model:                                          |   Wins   |    2.5% |    mean |   97.5% |\n"
## [3] " [ 1] withdraw ~ tenure + estress + affect + s   |     2%   |   1.147 |   1.158 |   1.171 |\n"
## [4] " [ 2] withdraw ~ tenure + affect + sex + age +   |    98%   |   1.136 |   1.148 |   1.161 |\n"
```

Gives the screen output of the function.

```
head(output$RMSEP)
```

```
##   Model    RMSEP
## 1     1 1.149405
## 2     1 1.158800
## 3     1 1.164977
## 4     1 1.159299
## 5     1 1.157755
## 6     1 1.162759
```

Gives for each model, for every cross validation cycle the estimated prediction error (RMSEP)