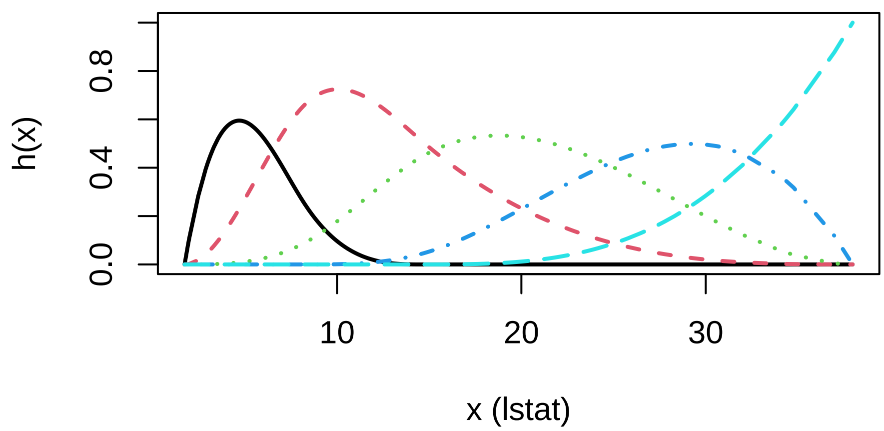
Answers to exercsies Session 5

Marjolein Fokkema

Exercise 1: Cubic and natural splines

a, b, c)

```
library("MASS")
data(Boston)
library("splines")
Boston <- Boston[order(Boston$lstat), ]
bs_x <- bs(Boston$lstat, df = 5)
matplot(Boston$lstat[order(Boston$lstat)], bs_x[order(Boston$lstat), ],
        type = "l", lwd = 2, xlab = "x (lstat)", ylab = "h(x)")
```




```
head(bs_x)
```

```
##               1           2            3 4 5
## [1,] 0.0000000 0.000000000 0.000000e+00 0 0
## [2,] 0.0828165 0.001247442 2.216045e-06 0 0
## [3,] 0.1074457 0.002148435 5.048215e-06 0 0
## [4,] 0.2824476 0.018017765 1.309221e-04 0 0
## [5,] 0.3928704 0.041199495 4.786657e-04 0 0
## [6,] 0.3952802 0.041885740 4.913730e-04 0 0
```

```
attr(bs_x, "knots")
```

```
## 33.33333% 66.66667%
##  8.316667 14.696667
```

```r
attr(bs_x, "degree")
```

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

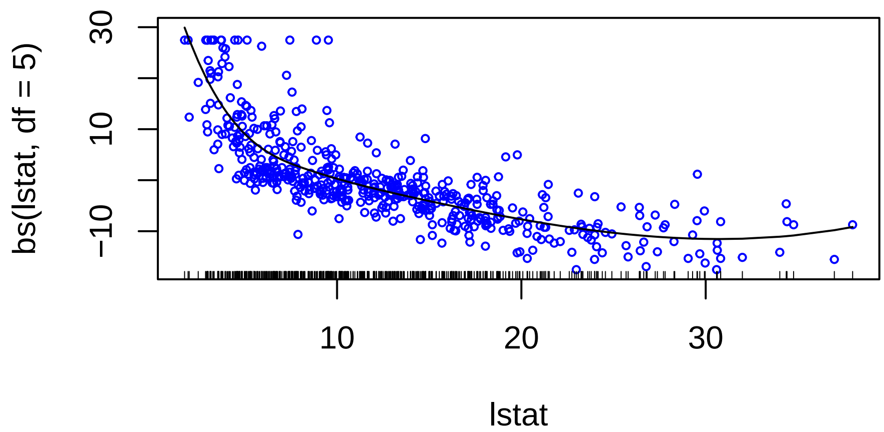
With `df = 5`, we obtain a design matrix of 5 columns. With a cubic spline, we use up 3 df for the first three expansions. Thus, with 5 df, we have 2 df 'left' to spend on the knots. Each knot introduces one additional basis function. Thus, with 5 df for a cubic spline, we can use 2 knots. Note that the knots are placed based on the univariate distribution of the predicton.

d)

```r
library("gam")
mod_df5 <- gam(medv ~ bs(lstat, df = 5), data = Boston)
summary(mod_df5)
```

```
##
## Call: gam(formula = medv ~ bs(lstat, df = 5), data = Boston)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1774  -3.1790  -0.7981   2.0964  26.6755
##
## (Dispersion Parameter for gaussian family taken to be 27.109)
##
##     Null Deviance: 42716.3 on 505 degrees of freedom
## Residual Deviance: 13554.52 on 500 degrees of freedom
## AIC: 3113.663
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                    Df Sum Sq Mean Sq F value    Pr(>F)
## bs(lstat, df = 5)   5  29162  5832.4  215.14 < 2.2e-16 ***
## Residuals         500  13554    27.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

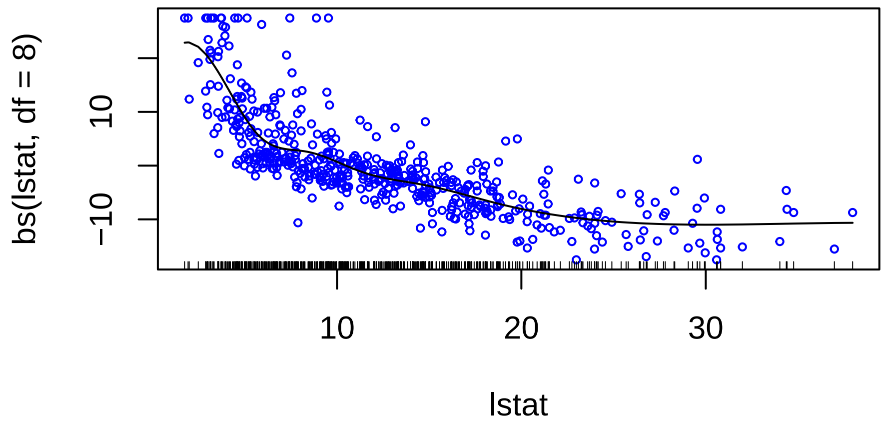
```r
plot(mod_df5, residuals = TRUE, col = "blue", cex = .5)
```




```
mod_df8 <- gam(medv ~ bs(lstat, df = 8), data = Boston)
summary(mod_df8)
```

```
##
## Call: gam(formula = medv ~ bs(lstat, df = 8), data = Boston)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -14.9627  -3.1253  -0.6612   2.0831  26.0972
##
## (Dispersion Parameter for gaussian family taken to be 26.7118)
##
##     Null Deviance: 42716.3 on 505 degrees of freedom
## Residual Deviance: 13275.77 on 497 degrees of freedom
## AIC: 3109.148
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                    Df Sum Sq Mean Sq F value    Pr(>F)
## bs(lstat, df = 8)   8  29441  3680.1  137.77 < 2.2e-16 ***
## Residuals         497  13276    26.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(mod_df8, residuals = TRUE, col = "blue", cex = .5)
```




```
BIC(mod_df5)
```

```
## [1] 3143.249
```

```
BIC(mod_df8)
```

```
## [1] 3151.414
```

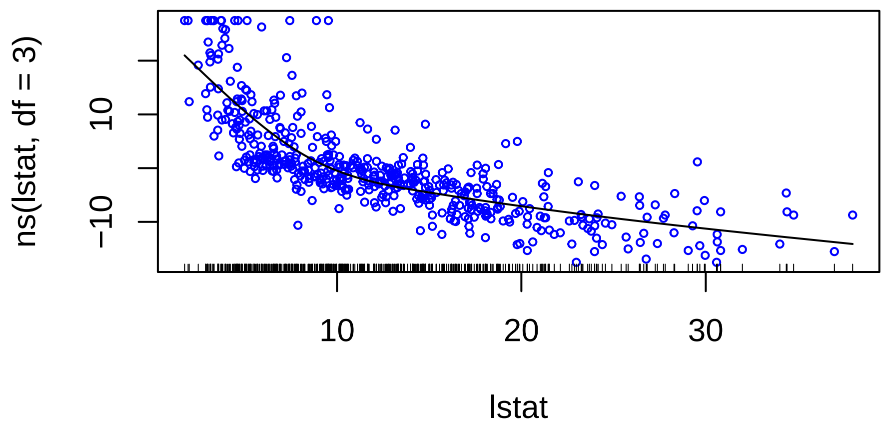
The 5 df cubic spline fits best according to BIC, the plots suggest similar: 8 df yields a slightly too wiggly function. Note that models with different (number of location of) knots are not nested, so we cannot use stattistical testing to compare the model fit.

e)

```
mod_ns3 <- gam(medv ~ ns(lstat, df = 3), data = Boston)
summary(mod_ns3)
```

```
##
## Call: gam(formula = medv ~ ns(lstat, df = 3), data = Boston)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -13.7595  -3.3628  -0.6468   2.3062  27.2857
##
## (Dispersion Parameter for gaussian family taken to be 28.4261)
##
##     Null Deviance: 42716.3 on 505 degrees of freedom
## Residual Deviance: 14269.9 on 502 degrees of freedom
## AIC: 3135.688
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                    Df Sum Sq Mean Sq F value    Pr(>F)
## ns(lstat, df = 3)   3  28446  9482.1  333.57 < 2.2e-16 ***
## Residuals         502  14270    28.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

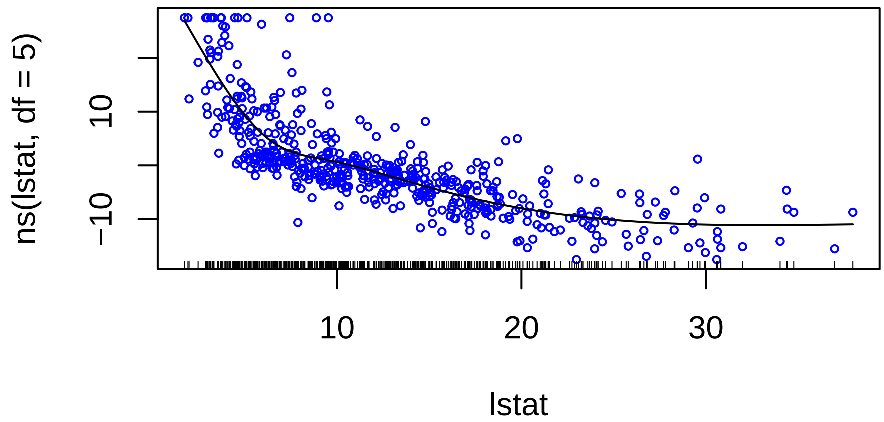
```r
plot(mod_ns3, residuals = TRUE, col = "blue", cex = .5)
```




```r
mod_ns5 <- gam(medv ~ ns(lstat, df = 5), data = Boston)
summary(mod_ns5)
```

```
##
## Call: gam(formula = medv ~ ns(lstat, df = 5), data = Boston)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -13.9811  -3.0266  -0.7252   2.1416  26.5111
##
## (Dispersion Parameter for gaussian family taken to be 26.9021)
##
##     Null Deviance: 42716.3 on 505 degrees of freedom
## Residual Deviance: 13451.03 on 500 degrees of freedom
## AIC: 3109.785
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                    Df Sum Sq Mean Sq F value    Pr(>F)
## ns(lstat, df = 5)   5  29265  5853.1  217.57 < 2.2e-16 ***
## Residuals         500  13451    26.9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
plot(mod_ns5, residuals = TRUE, col = "blue", cex = .5)
```




```
BIC(mod_ns3)
```

```
## [1] 3156.82
```

```
BIC(mod_ns5)
```

```
## [1] 3139.37
```

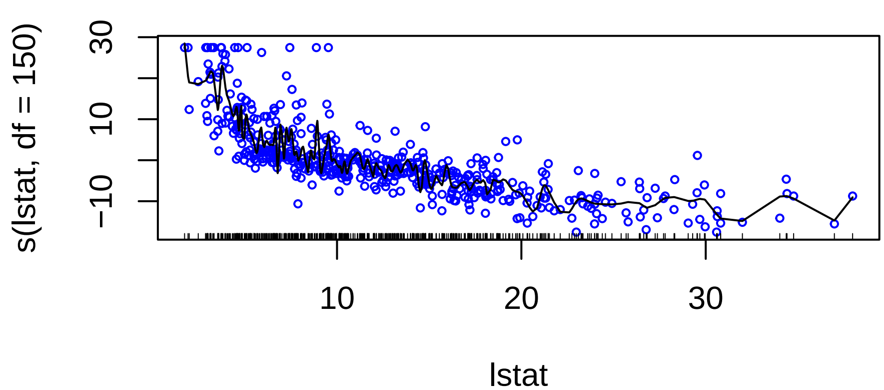
The lowest BIC value was obtained for the natural spline with 5 df. Visually, both the 3 and 5 df natural splines seem to provide a good fit to the data.

Exercise 2: Smoothing spline

```
mod_sc <- gam(medv ~ s(lstat, df = 150), data = Boston) # complex fit
summary(mod_sc)
```

```
##
## Call: gam(formula = medv ~ s(lstat, df = 150), data = Boston)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -14.0712  -2.7340  -0.4702   2.0649  21.9654
##
## (Dispersion Parameter for gaussian family taken to be 27.6915)
##
##     Null Deviance: 42716.3 on 505 degrees of freedom
## Residual Deviance: 10772.01 on 389 degrees of freedom
## AIC: 3219.4
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## s(lstat, df = 150)    1  23244 23243.9  839.39 < 2.2e-16 ***
## Residuals           389  10772    27.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                    Npar Df Npar F     Pr(F)
## (Intercept)
## s(lstat, df = 150)     115 2.7321 2.103e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(mod_sc, residuals = TRUE, cex = .5, col = "blue")
```

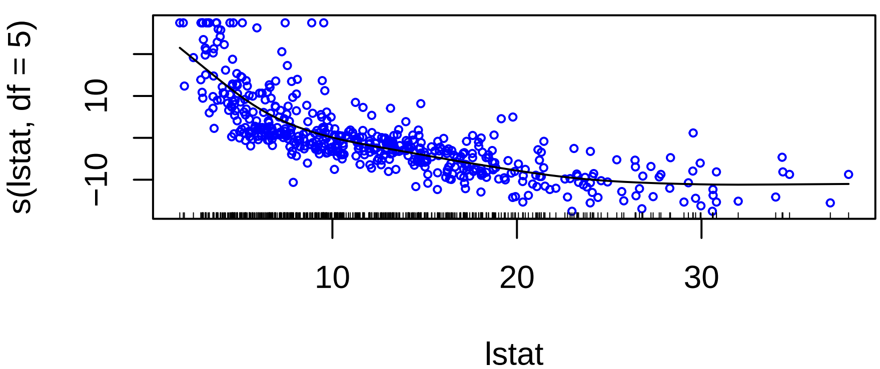


The results present both a parametric and non-parametric effect of `lstat`. The parametric effect represent the linear slope, thus using only 1 df. The non-parametric effects represent the non-linear effects. The non-linear part of the smoothing spline for `lstat` took up 115 degrees of freedom. Note that this is less than the requested degrees of freedom, because by default the knots are placed at a subset of the observations, for computational considerations. In addition, > 115 knots would be rarely needed to approximate a curve.

```
mod_ss <- gam(medv ~ s(lstat, df = 5), data = Boston) # more simple fit
summary(mod_ss)
```

```
##
## Call: gam(formula = medv ~ s(lstat, df = 5), data = Boston)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -13.6332  -3.2159  -0.6577   2.2051  26.8386
##
## (Dispersion Parameter for gaussian family taken to be 27.6492)
##
##     Null Deviance: 42716.3 on 505 degrees of freedom
## Residual Deviance: 13824.6 on 499.9999 degrees of freedom
## AIC: 3123.646
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                   Df Sum Sq Mean Sq F value    Pr(>F)
## s(lstat, df = 5)   1  23244 23243.9  840.67 < 2.2e-16 ***
## Residuals        500  13825    27.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df Npar F     Pr(F)
## (Intercept)
## s(lstat, df = 5)       4 51.065 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(mod_ss, residuals = TRUE, col = "blue", cex = .5)
```



With 5 df, the flexibility is much lower, and we obtain a much smoother fit.

```
BIC(mod_ss)
```

```
## [1] 3136.326
```

```
BIC(mod_sc)
```

```
## [1] 3232.079
```

```
BIC(mod_ns5)
```

```
## [1] 3139.37
```

According to the BIC, the more heavily penalized smoothing spline (i.e., with df = 5) has better fit than the less reularized smoothing spline. This is in accordance with what we can conclude from the visual inspection of the fitted smoothing splines. The smoothing spline with 5 df also outperforms the natural spline with 5 df from the previous exercise. Thus, the non-parametric smoothing spline approach appears to improve on the parametric natural and cubic spline approaches.

Exercise 3: Multiple predictors, binary outcome

```
detach("package:gam", unload=TRUE)
library("mgcv")
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-35. For overview type 'help("mgcv-package")'.
```

```
MASQ <- read.table("MASQ.txt")
set.seed(1)
train <- sample(1:nrow(MASQ), size = nrow(MASQ)*.8)
summary(MASQ)
```

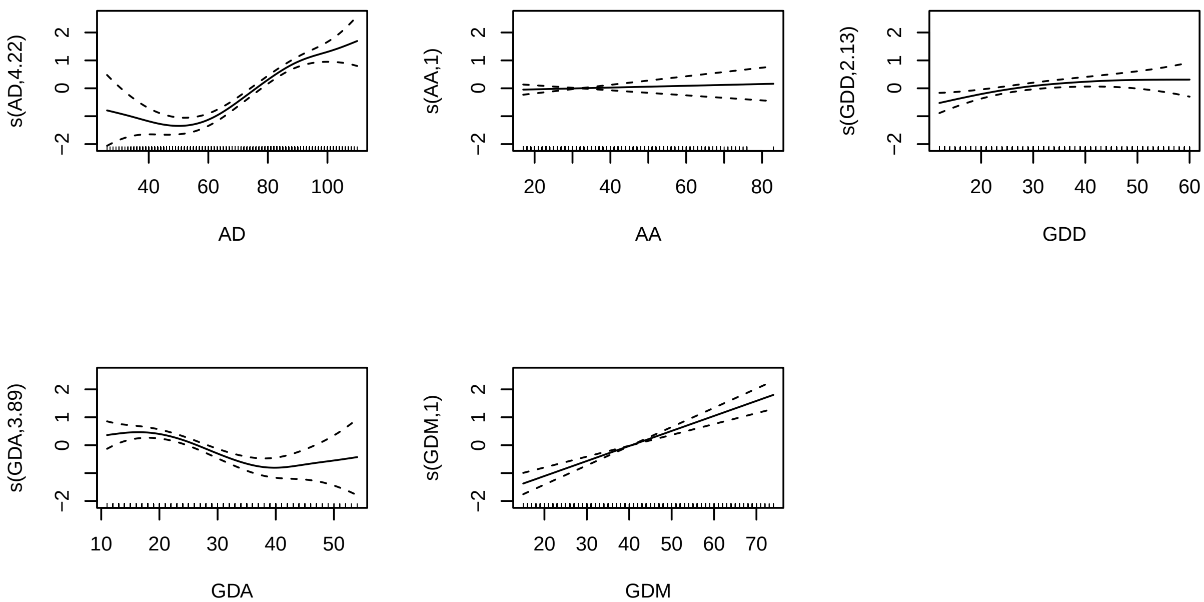
```
##     D_DEPDYS            AD                 AA              GDD
##  Min.   :0.0000   Min.   : 26.00   Min.   :17.00   Min.   :12.00
##  1st Qu.:0.0000   1st Qu.: 64.00   1st Qu.:22.00   1st Qu.:20.00
##  Median :0.0000   Median : 77.00   Median :28.00   Median :29.00
##  Mean   :0.4643   Mean   : 75.05   Mean   :32.01   Mean   :30.64
##  3rd Qu.:1.0000   3rd Qu.: 88.00   3rd Qu.:39.00   3rd Qu.:40.00
##  Max.   :1.0000   Max.   :110.00   Max.   :83.00   Max.   :60.00
##       GDA             GDM           leeftijd       geslacht
##  Min.   :11.0   Min.   :15.0   Min.   :17.0   Length:3597
##  1st Qu.:19.0   1st Qu.:31.0   1st Qu.:28.0   Class :character
##  Median :24.0   Median :40.0   Median :38.0   Mode  :character
##  Mean   :25.4   Mean   :40.6   Mean   :38.8
##  3rd Qu.:31.0   3rd Qu.:50.0   3rd Qu.:48.0
##  Max.   :54.0   Max.   :75.0   Max.   :91.0
##      D_TOT
##  Min.   :0.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :2.127
##  3rd Qu.:4.000
##  Max.   :7.000
```

```
GAM <- gam(D_DEPDYS ~ s(AD) + s(AA) + s(GDD) + s(GDA) + s(GDM),
           data = MASQ[train, ], method = "REML", family = "binomial")
summary(GAM)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## D_DEPDYS ~ s(AD) + s(AA) + s(GDD) + s(GDA) + s(GDM)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.25204    0.04854  -5.192 2.08e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df  Chi.sq  p-value
## s(AD)  4.221  5.213 156.700  < 2e-16 ***
```

```
## s(AA)  1.001  1.002   0.272   0.6026
## s(GDD) 2.132  2.725   9.315   0.0216 *
## s(GDA) 3.891  4.847  34.126 3.71e-06 ***
## s(GDM) 1.001  1.001  51.882  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.325   Deviance explained = 26.4%
## -REML = 1482.6  Scale est. = 1         n = 2877
```

```
par(mfrow = c(2, 3))
plot(GAM)
```



We compute the mean squared error and misclassification rate using predicted probabilities, for both training and test observations:

```
## Training data
GAM_preds_train <- predict(GAM, newdata = MASQ[train, ], type = "response")
mean((MASQ[train, "D_DEPDYS"] - GAM_preds_train)^2) ## Brier score
```

```
## [1] 0.1669075
```

```
tab_train <- prop.table(table(MASQ[train, "D_DEPDYS"], GAM_preds_train > .5)) ## confusion matrix
tab_train
```

```
##
##         FALSE      TRUE
##   0 0.4177963 0.1220021
##   1 0.1220021 0.3381995
```

```
## Test data
GAM_preds_test <- predict(GAM, newdata = MASQ[-train, ], type = "response")
mean((MASQ[-train, "D_DEPDYS"] - GAM_preds_test)^2) ## Brier score
```

```
## [1] 0.1690516
```

```r
tab_test <- prop.table(table(MASQ[-train, "D_DEPDYS"], GAM_preds_test > .5)) ## confusion matrix
tab_test
```

```
##
##         FALSE      TRUE
##   0 0.4013889 0.1180556
##   1 0.1263889 0.3541667
```

The Brier score and confusion matrices are quite similar between training and test data, indicating little overfitting.

Exercise 4: Fit an SVM

```
library("e1071")
```

```
## Warning: package 'e1071' was built under R version 4.1.1
```

```
cost <- c(.001, .01, .1, 1, 5, 10, 100)
set.seed(42)
names(MASQ)
```

```
## [1] "D_DEPDYS" "AD"        "AA"        "GDD"       "GDA"       "GDM"       "leeftijd"
## [8] "geslacht" "D_TOT"
```

```
MASQ <- MASQ[ , -9]
names(MASQ)
```

```
## [1] "D_DEPDYS" "AD"        "AA"        "GDD"       "GDA"       "GDM"       "leeftijd"
## [8] "geslacht"
```

```
MASQ$D_DEPDYS <- factor(MASQ$D_DEPDYS)
```

```
tune.out <- tune(svm, D_DEPDYS ~ ., data = MASQ[train, ], kernel = "linear",
                 ranges = list(cost = cost))
```

```
tune.out$best.parameters
```

```
##   cost
## 3  0.1
```

```
svmfit <- svm(D_DEPDYS ~ ., data = MASQ[train,], kernel = "linear",
              cost = 0.1)
```

```
tab_train <- table(MASQ[train, "D_DEPDYS"],
                   predict(svmfit, newdata = MASQ[train, ], probability = FALSE))
tab_train
```

```
##
##        0    1
##   0 1218  335
##   1  360  964
```

```
1 - sum(diag(prop.table(tab_train))) ## misclassification rate
```

```
## [1] 0.2415711
```