Nonlinear Modelling

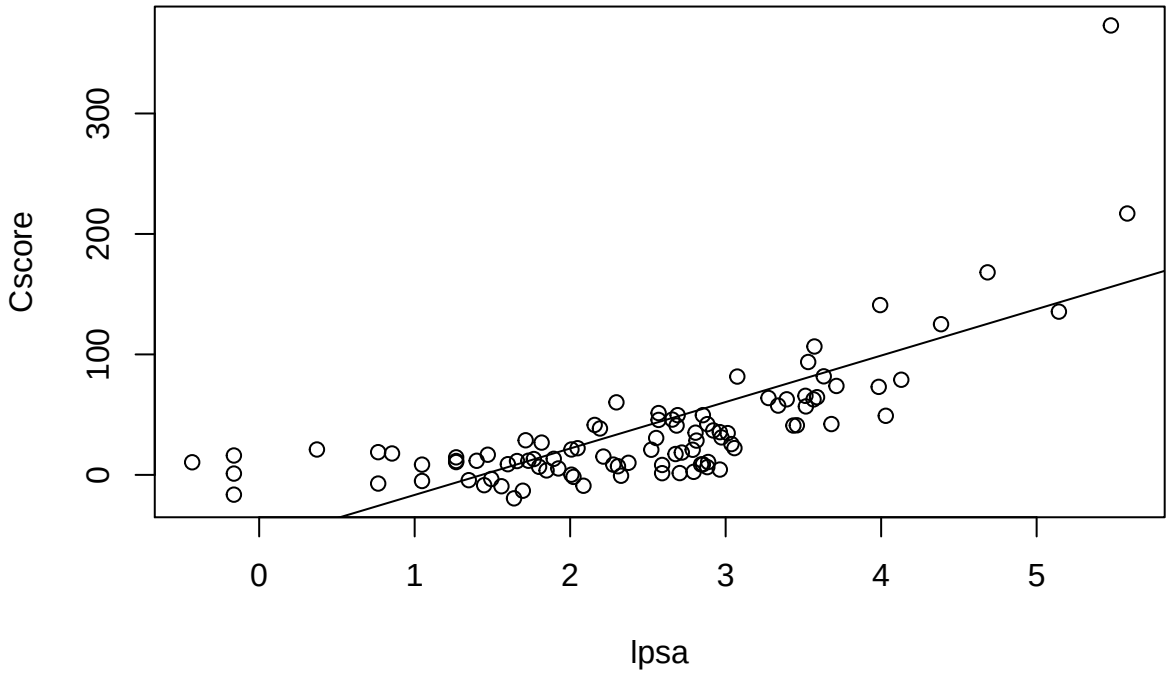
Thomas Vanpoucke, Boris Shilov

Exploring the association

First we randomise the rows and split the data set into a training and validation subset.

```
sample = sample.int(n = nrow(prostate), size = floor(.5*nrow(prostate)), replace = F)
train = prostate[sample,]
test = prostate[-sample,]
```

```
defaultfit = lm(Cscore ~ lpsa, data = prostate, subset = sample)
plot(Cscore ~ lpsa)
abline(defaultfit)
```




```
summary(defaultfit)
```

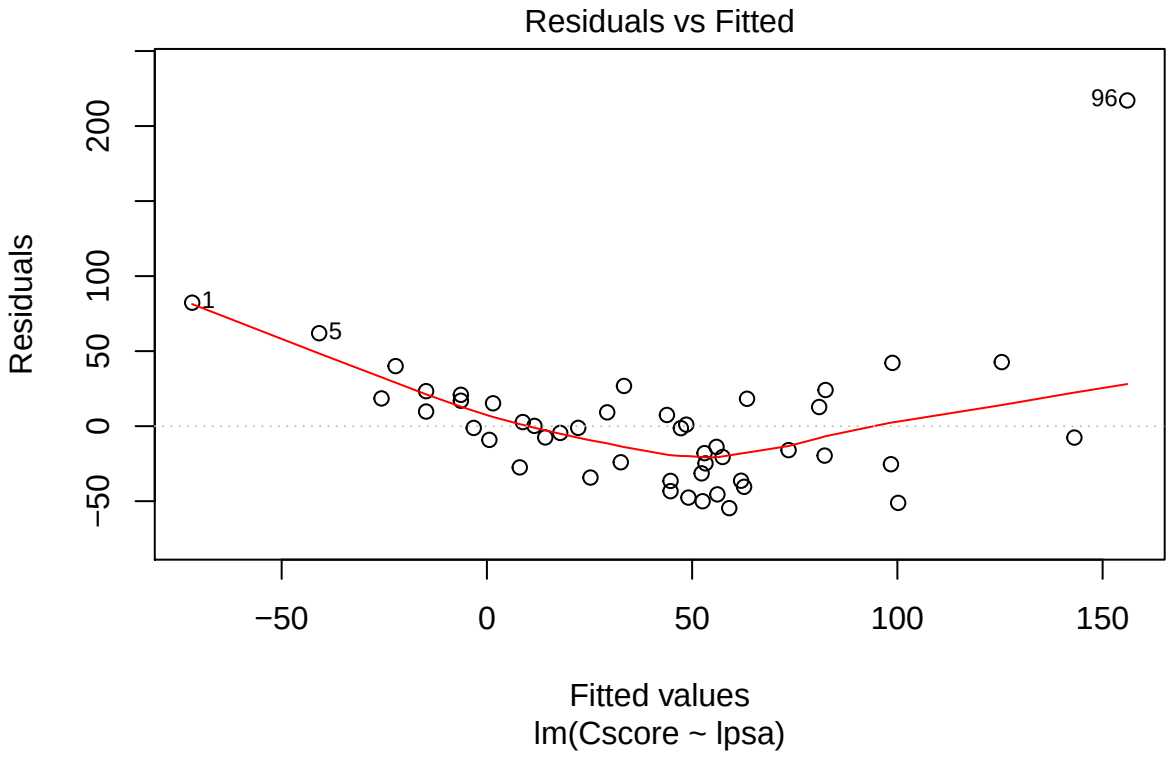
```
##
## Call:
## lm(formula = Cscore ~ lpsa, data = prostate, subset = sample)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -54.63 -25.90  -2.93  17.28 217.06
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -55.19      15.05   -3.67  0.00064 ***
## lpsa           38.56       5.48    7.04    8e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

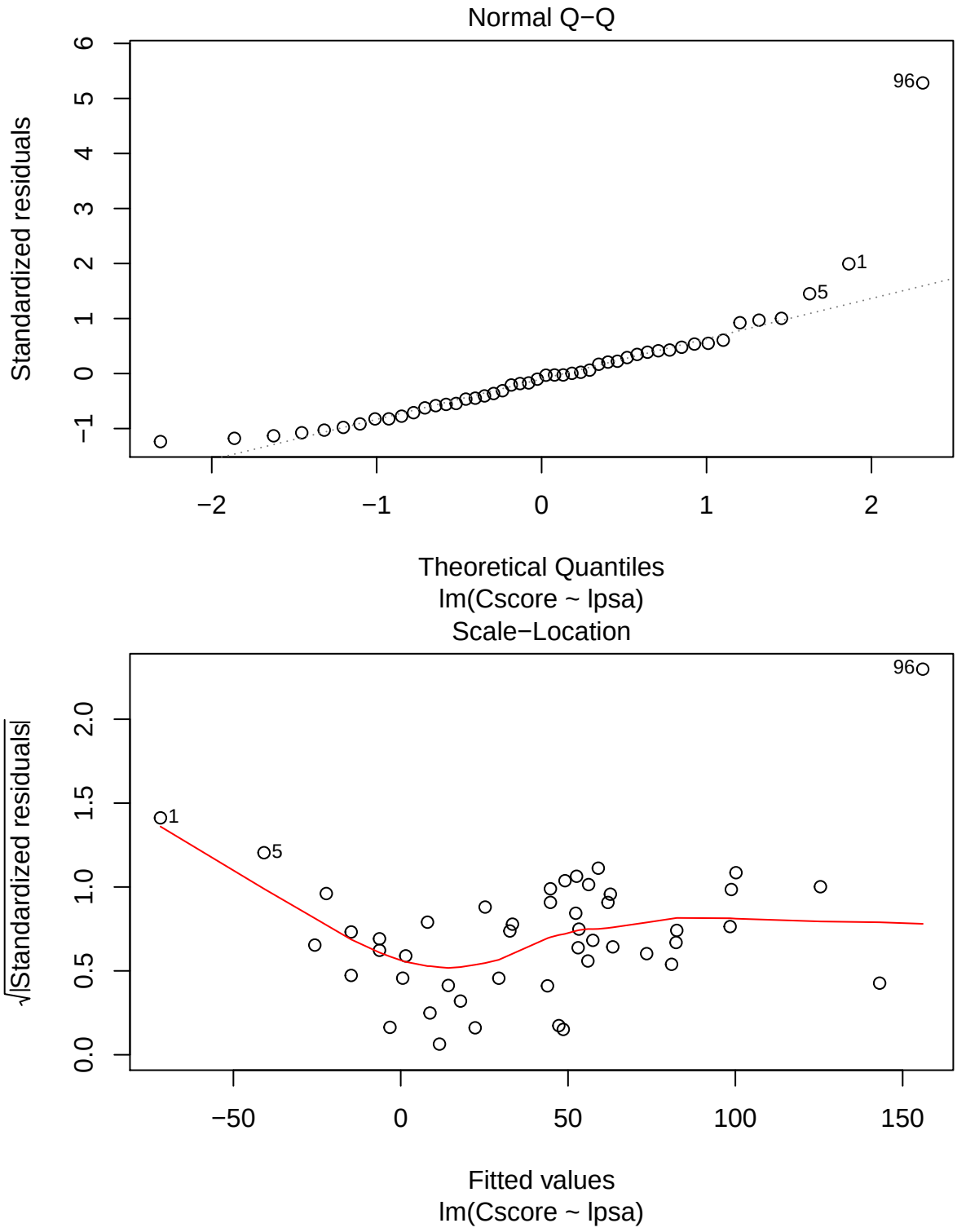
```
##
## Residual standard error: 44.7 on 46 degrees of freedom
## Multiple R-squared:  0.519,  Adjusted R-squared:  0.508
## F-statistic: 49.5 on 1 and 46 DF,  p-value: 7.97e-09
```

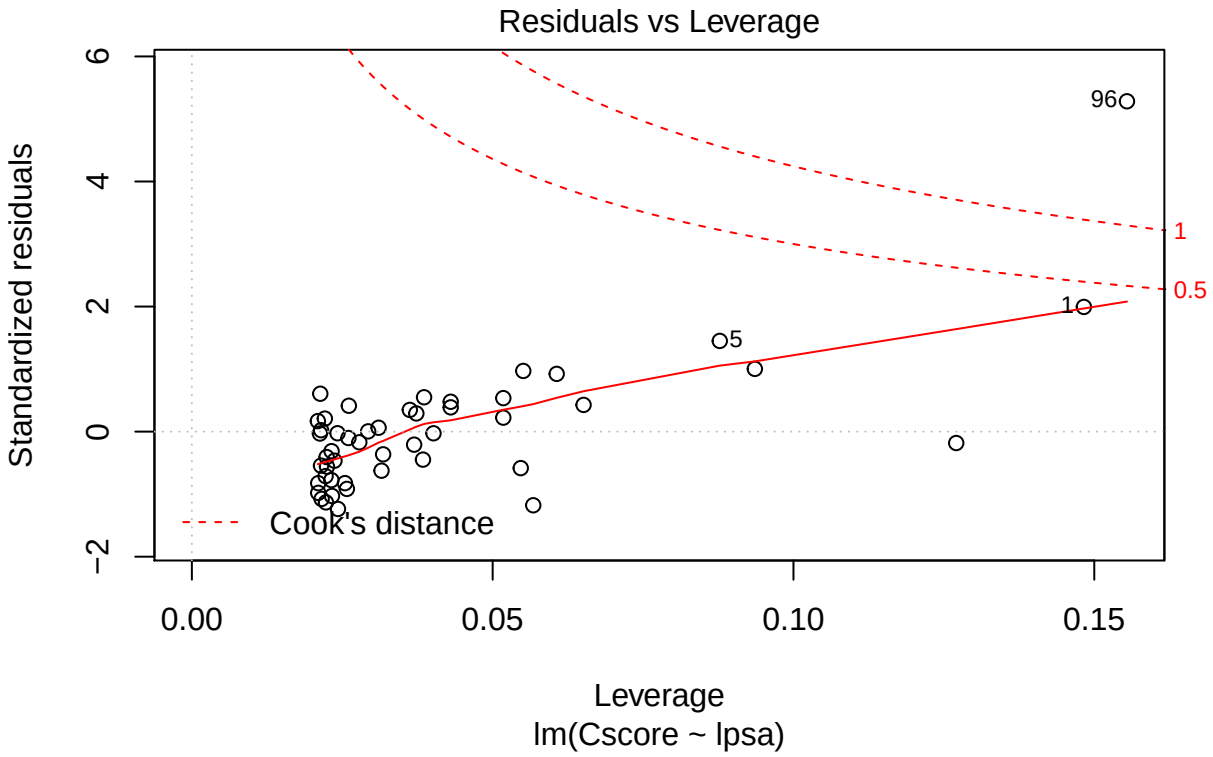
It can be seen that the trend between Cscore and lpsa does not appear perfectly linear, just by visual inspection. Some summary statistics confirm that a simple linear model only explains about half of the variance in this association (R^2adj = 50,8%).

Having confirmed our suspicion that there may be some non-linearity involved, we can turn to more powerful diagnostics to determine more conclusively what is happening.

```
plot(defaultfit)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Cscore ~ lpsa)
Scale−Location
√|Standardized residuals|
Fitted values
lm(Cscore ~ lpsa)
96
1
5



The plot of residuals against fitted values clearly resembles a parabola more than a straight line expected of a dataset with good linearity. Thus we strongly suspect that there is a large deviation from our assumption of linearity.

The Q-Q plot appears much less concerning, but there is still a notable parabolic relationship. The tails of the plot are clearly edging upwards, especially in the top half, where many data points of concern are located. Data point number 96 is particularly problematic. The Q-Q plot thus clearly indicates a small departure from the assumption of normality.

The scale-location plot again demonstrates a strong parabolic pattern, although there is a good spread of points - but again the problematic data points at the higher end are highlighted. Thus we strongly suspect that the residuals are not spread equally along the predictor ranges, and thus the assumption of homoscedasticity is violated as well.

The last diagnostic plot, residuals against leverage, shows that data point 96 is indeed a very influential case, and thus we would probably alter our model significantly if we exluded it from analysis. Data point 1 is also relatively influential cases.

Thus we can conclude rather confidently that most of the assumptions underlying linearity do not hold up very well for this association.

Nonlinear modelling

Polynomial regression

The first model we constructed was a polynomial regression. We will do this up to the 9th degree polynomial, since this equals ten degrees of freedom.

```
poly_list = list()
MSE_list = list()
```

```
for (i in 1:9){
  model = lm(Cscore~poly(lpsa,i), data=prostate, subset=sample)
  poly_list = c(poly_list, list(model))
  mean = mean((Cscore-predict(model,prostate))[-sample]^2)
  MSE_list = c(MSE_list, list(mean))
}
```

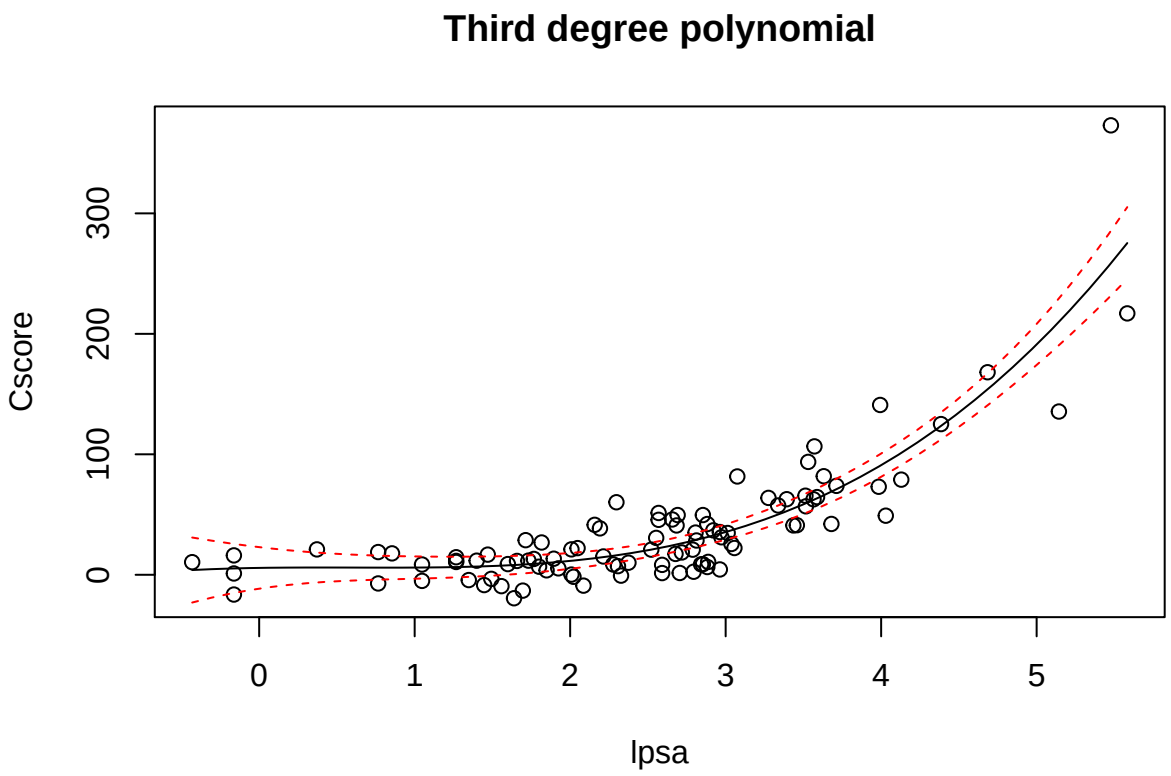
As can be seen, the model that has the best test MSE is a third degree polynomial model. We can further confirm this using an ANOVA method for picking the best model.

```
eval(parse(text=paste("anova(",paste("poly_list[[",1:length(poly_list),"]]",sep="",collapse=","),")")))
```

```
## Analysis of Variance Table
##
## Model 1: Cscore ~ poly(lpsa, i)
## Model 2: Cscore ~ poly(lpsa, i)
## Model 3: Cscore ~ poly(lpsa, i)
## Model 4: Cscore ~ poly(lpsa, i)
## Model 5: Cscore ~ poly(lpsa, i)
## Model 6: Cscore ~ poly(lpsa, i)
## Model 7: Cscore ~ poly(lpsa, i)
## Model 8: Cscore ~ poly(lpsa, i)
## Model 9: Cscore ~ poly(lpsa, i)
##   Res.Df   RSS Df Sum of Sq     F Pr(>F)
## 1     46 91956
## 2     45 42249  1     49707 87.79  2e-11 ***
## 3     44 37013  1      5237  9.25 0.0043 **
## 4     43 35259  1      1753  3.10 0.0865 .
## 5     42 31186  1      4073  7.19 0.0108 *
## 6     41 27178  1      4008  7.08 0.0114 *
## 7     40 23216  1      3963  7.00 0.0118 *
## 8     39 22183  1      1032  1.82 0.1849
## 9     38 21515  1       668  1.18 0.2843
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model 3 is a significant improvement over Model 2, whereas Model 4 gives no significant improvement under a 0.05 significance threshold. Thus a second or third degree polynomial model provides a good fit to the data, and using our previous test MSE result we can pick the third degree polynomial as the best model with some confidence.

```
best_poly_model = lm(Cscore~poly(lpsa, 3), data = prostate)

lpsalims = range(lpsa)
lpsa.grid = seq(lpsalims[1], lpsalims[2], length.out = 50)
prediction = predict(best_poly_model, newdata = list(lpsa = lpsa.grid), se = T)
SE.bands = cbind(prediction$fit+2*prediction$se.fit, prediction$fit-2*prediction$se.fit)

plot(lpsa, Cscore, xlim=lpsalims)
title("Third degree polynomial")
lines(lpsa.grid, prediction$fit)
matlines(lpsa.grid, SE.bands, lwd=1, col="red", lty=2)
```



Cubic splines

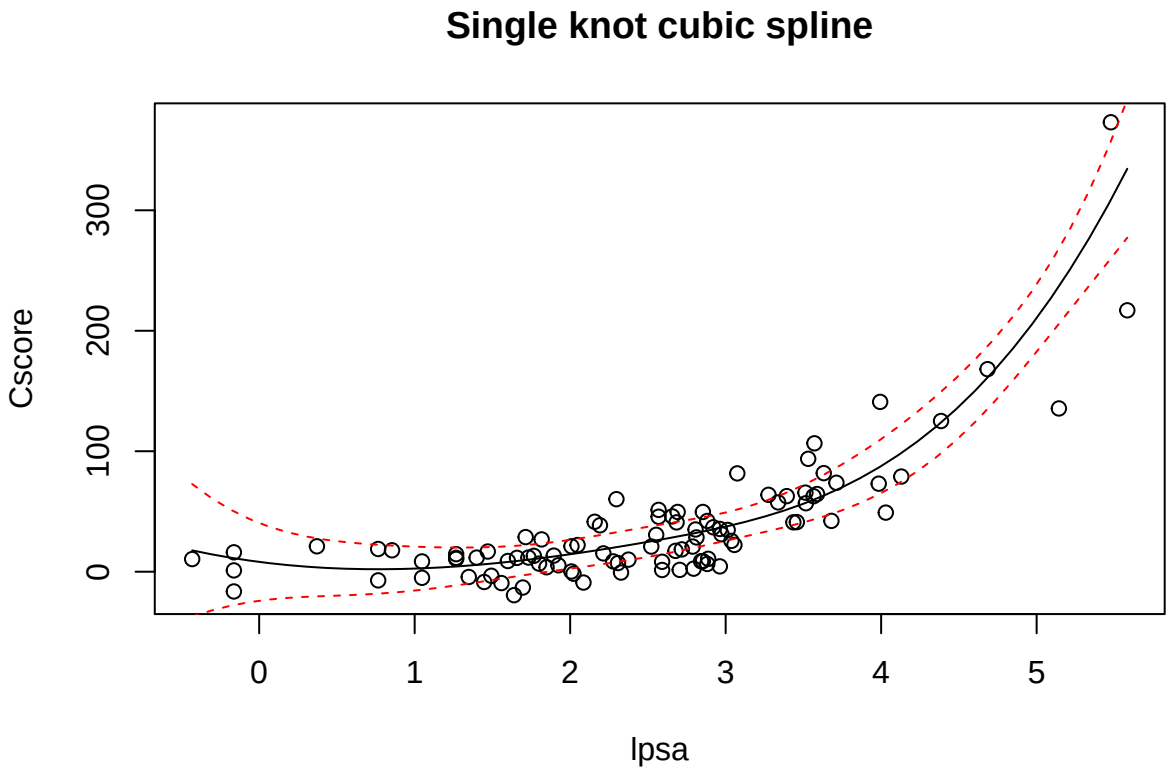
The next regression model we construct is a cubic spline. Here, we start from four degrees of freedom, since a cubic spline has three polynomial terms (first, second and third order) and at least one knot. We let the knots be put at uniform quantiles.

```
library(splines)

cubic_list = c()
MSE_list = c()
for (i in 4:10){
  model = lm(Cscore~bs(lpsa, df = i),data=prostate, subset=sample)
  cubic_list = c(cubic_list, model)
  mean = mean((Cscore - predict(model, prostate))[-sample]^2)
  MSE_list = c(MSE_list, list(mean))
}
best_cubic_model = cubic_list[[which.min(MSE_list)]]
best_cubic_model
```

```
##       (Intercept) bs(lpsa, df = i)1 bs(lpsa, df = i)2 bs(lpsa, df = i)3
##            17.694           -28.134            -0.375            64.644
## bs(lpsa, df = i)4
##           316.774
```

```
best_cubic = lm(Cscore~bs(lpsa, df = 4),data=prostate, subset=sample)

lpsalims = range(lpsa)
lpsa.grid = seq(lpsalims[1], lpsalims[2], length.out = 50)
prediction = predict(best_cubic, newdata = list(lpsa = lpsa.grid), se = T)
```

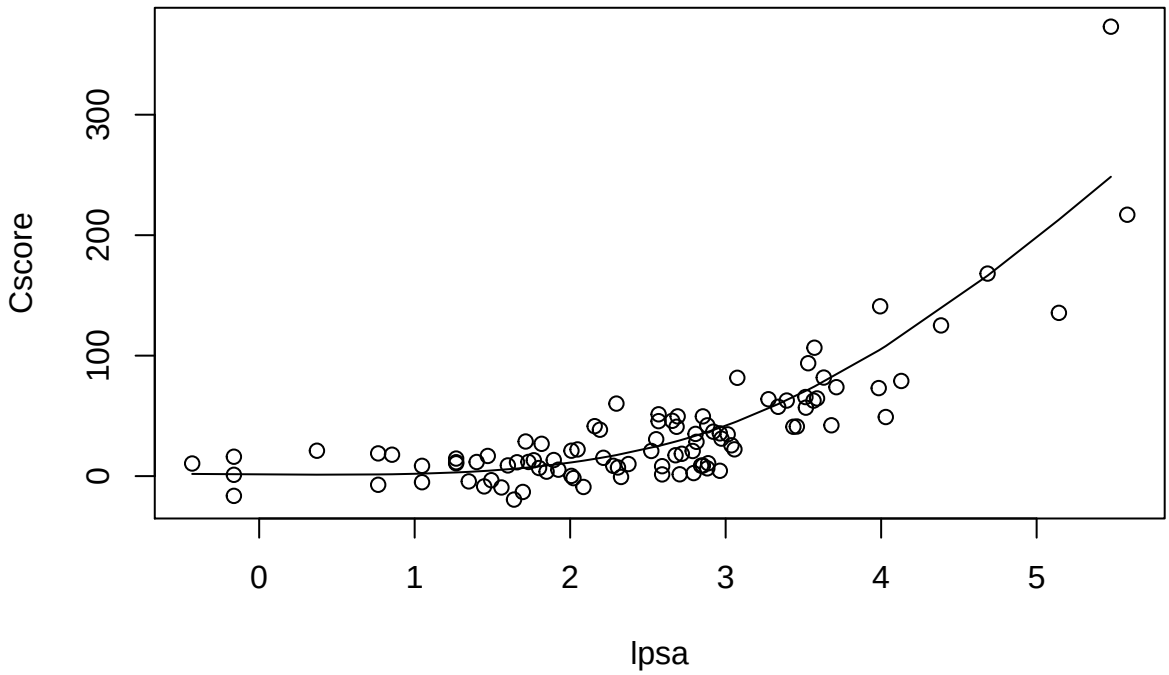
```
SE.bands = cbind(prediction$fit+2*prediction$se.fit, prediction$fit-2*prediction$se.fit)

plot(lpsa, Cscore, xlim=lpsalims)
title("Single knot cubic spline")
lines(lpsa.grid, prediction$fit)
matlines(lpsa.grid, SE.bands, lwd=1, col="red", lty=2)
```



Smoothing Splines

The last model we construct is a smoothing spline. We can use the built in leave-out-one cross validation to arrive at a good model with a degrees of freedom value within the desired range.

```
plot(lpsa, Cscore)
fit2 = smooth.spline(lpsa[sample], Cscore[sample], cv=T)
lpsalims = range(lpsa)
lpsa.grid = seq(lpsalims[1], lpsalims[2], length.out = 50)
prediction = predict(fit2, newdata = list(lpsa = lpsa.grid), se=T, data = prostate)
lines(prediction)
```




```
fit2$df
```

```
## [1] 3.54
```

```
predictedVals = predict(fit2, test$lpsa)$y
errors = test$Cscore - predictedVals
smooth.mse = mean(errors^2)
smooth.mse
```

```
## [1] 297
```

Model comparison

We compare all the models to see which one gives us the best test MSE.

```
linear.mse = mean((Cscore-predict(defaultfit, prostate))[-sample]^2)
poly.mse = mean((Cscore-predict(best_poly_model, prostate))[-sample]^2)
cubic.mse =  mean((Cscore-predict(best_cubic, prostate))[-sample]^2)

linear.mse
```

```
## [1] 851
```

```
poly.mse
```

```
## [1] 260
```

```
cubic.mse
```

```
## [1] 487
```

```
smooth.mse
```

```
## [1] 297
```

These are the test mean square errors for the linear, polynomial, cubic spline and smooth spline models, respectively. The nonlinear models all have better performance than the linear regression, and the polynomial

regression is still better than the cubic spline with a test MSE error of 260. Even the cubic spline, which is the worst performing of these nonlinear models, has almost halved the test MSE.

We surprised that the polynomial regression is the best model, since it is a cubic model and the cubic spline also uses a cubic model, in this case with one knot. We posit that the cause of this difference is that the knot adds undesired variance.

General additive model

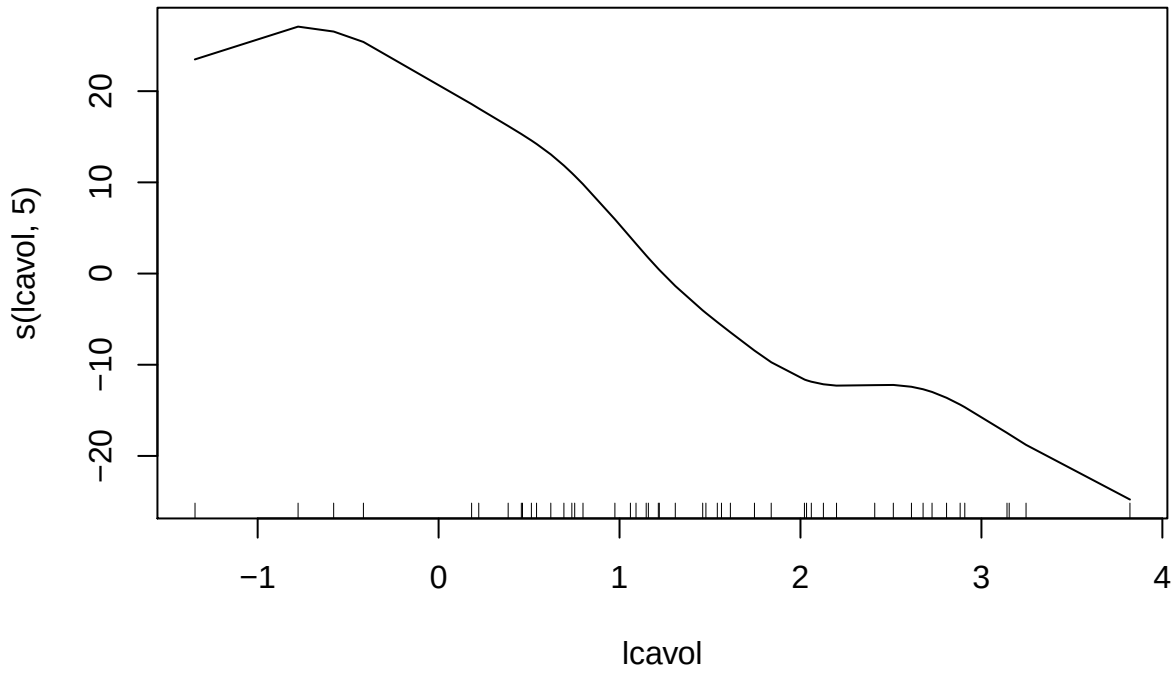
All variables

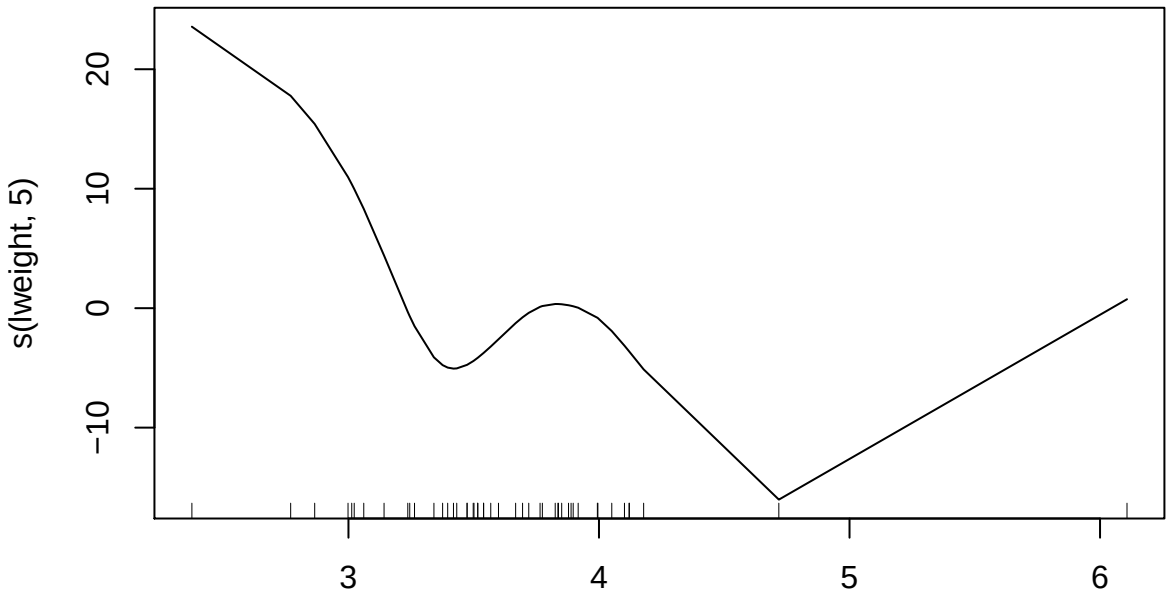
```
library(gam)
```

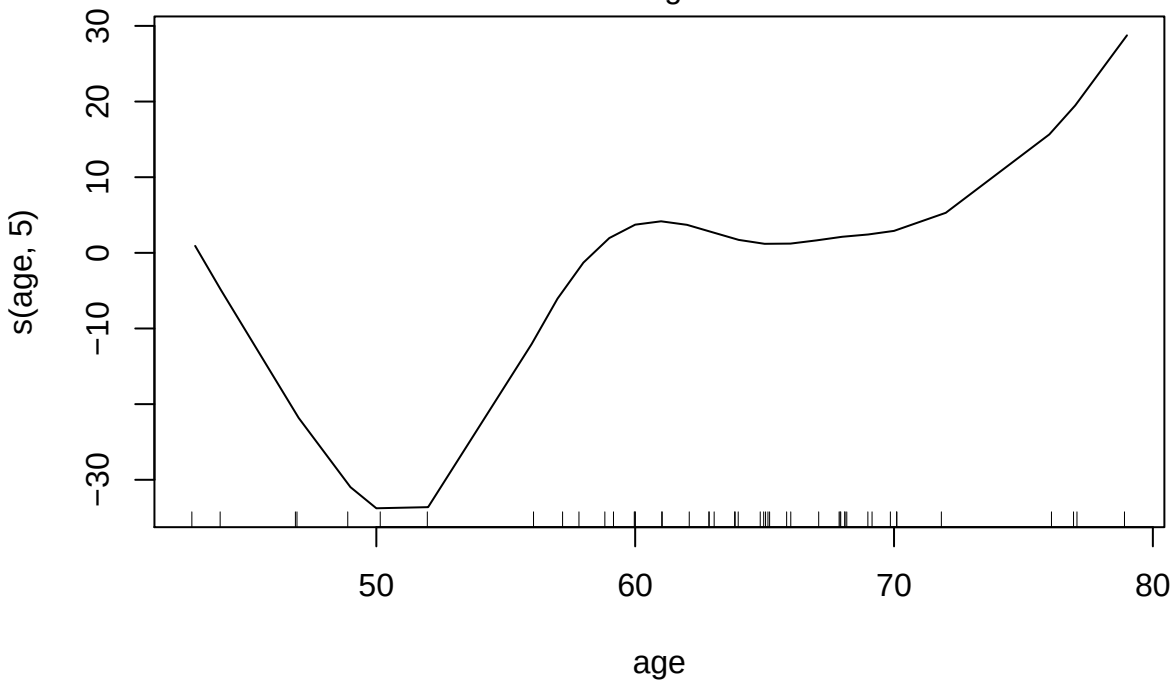
```
## Loading required package: foreach
```

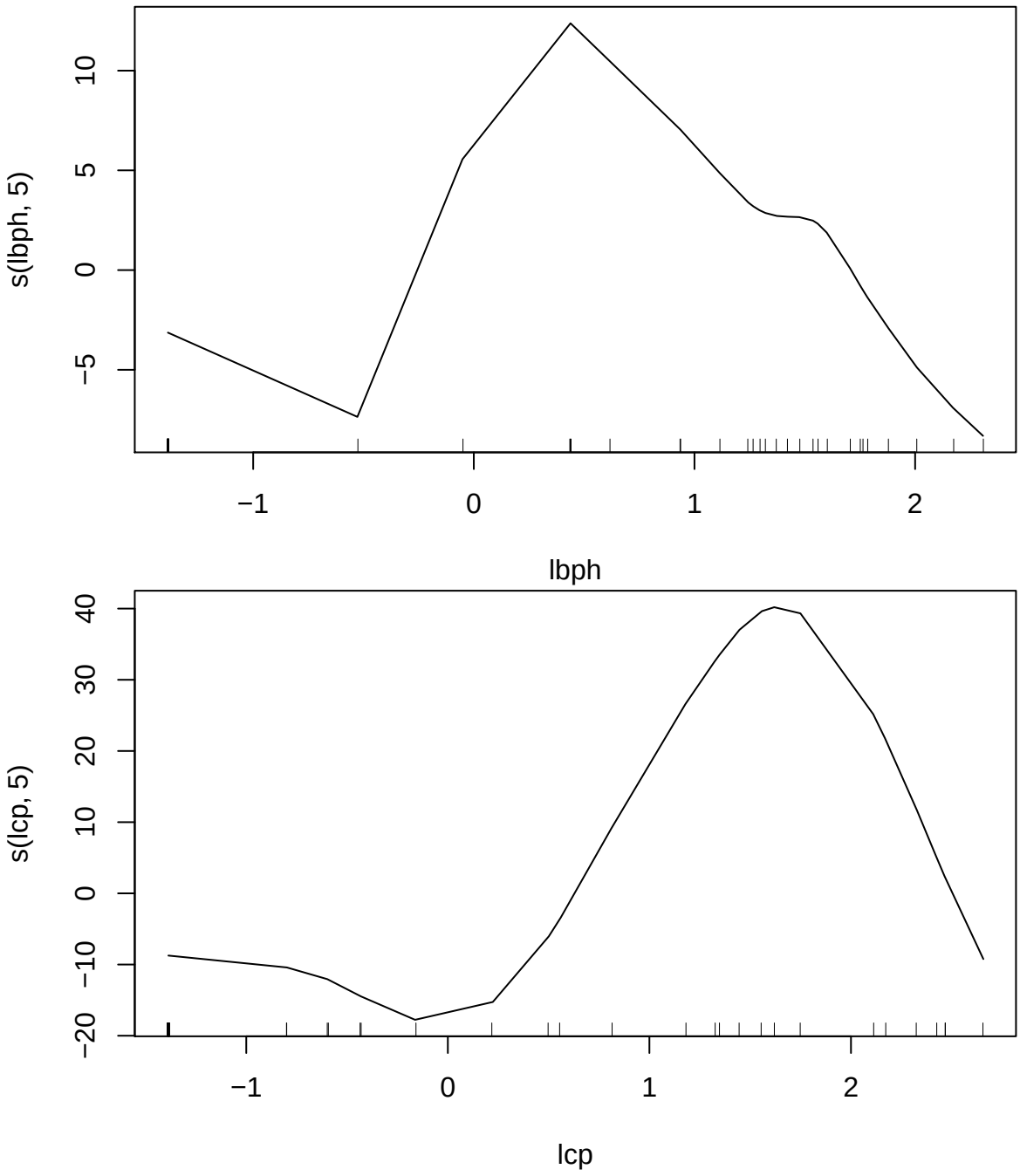
```
## Loaded gam 1.15
```

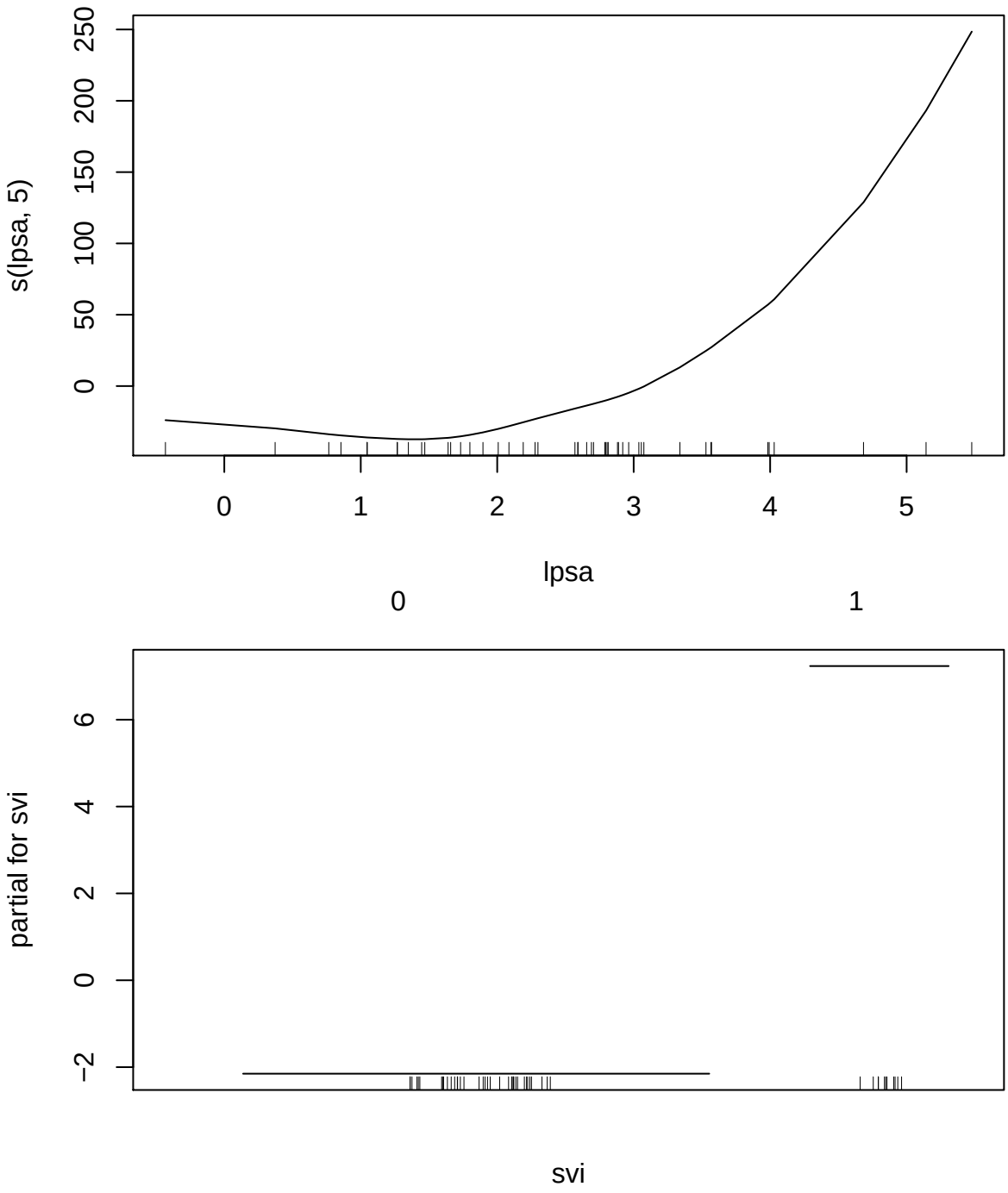
```
gam.all = gam(Cscore~s(lcavol,5)+s(lweight,5)+s(age,5)+s(lbph,5)+s(lcp,5)+s(lpsa,5)+svi, data = train)
plot(gam.all)
```




s(lweight, 5)
20
10
0
−10
3
4
5
6
lweight


s(age, 5)
30
20
10
0
−10
−30
50
60
70
80
age


s(lbph, 5)
10
5
0
−5
−1
0
1
2
lbph
s(lcp, 5)
40
30
20
10
0
−10
−20
−1
0
1
2
lcp




```
gam.all.mse = mean((Cscore-predict(gam.all, prostate))[-sample]^2)
gam.all.mse
```

```
## [1] 570
```

```
summary(gam.all)
```

```
##
## Call: gam(formula = Cscore ~ s(lcavol, 5) + s(lweight, 5) + s(age,
##     5) + s(lbph, 5) + s(lcp, 5) + s(lpsa, 5) + svi, data = train)
## Deviance Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -51.53  -8.59  -1.88   3.80  46.89
##
## (Dispersion Parameter for gaussian family taken to be 764)
##
##     Null Deviance: 191001 on 47 degrees of freedom
## Residual Deviance: 12218 on 16 degrees of freedom
## AIC: 468
##
## Number of Local Scoring Iterations: 3
##
## Anova for Parametric Effects
##               Df Sum Sq Mean Sq F value  Pr(>F)
## s(lcavol, 5)   1  43188   43188   56.56 1.2e-06 ***
## s(lweight, 5)  1   3273    3273    4.29  0.0550 .
## s(age, 5)      1   3301    3301    4.32  0.0540 .
## s(lbph, 5)     1   2824    2824    3.70  0.0724 .
## s(lcp, 5)      1  12967   12967   16.98  0.0008 ***
## s(lpsa, 5)     1  46397   46397   60.76 7.8e-07 ***
## svi            1    304     304    0.40  0.5367
## Residuals     16  12218     764
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##               Npar Df Npar F   Pr(F)
## (Intercept)
## s(lcavol, 5)        4   0.33   0.854
## s(lweight, 5)       4   0.86   0.510
## s(age, 5)           4   1.74   0.190
## s(lbph, 5)          4   0.44   0.775
## s(lcp, 5)           4   2.56   0.079 .
## s(lpsa, 5)          4  15.22 2.6e-05 ***
## svi
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
gam.all.mse
```

```
## [1] 570
```

If we look at the summary of the full GAM model, we can see that the predictor variables are all significant or almost significant, excepting svi, because it is a categorical variable. This was to be expected as the model we use is quite flexible.

We also looked at the test MSE, which is 570 for this model. This is much lower than the forward selected model from our previous work, which had a test MSE of 803. Thus it is a better model.

We can also see that the model is not as smooth near the edges for some of the predictors. This is because of the sparseness of the data at these extreme values.

The model is thus an improvement on our previous work.

Backward selection

```
getAIC <- function(data, nbloops){
namelist = c()
for (i in 1:nbloops){
  name = paste(data,i,sep ="")
  namelist = c(namelist,name)
}
aiclist = lapply(namelist, function(x)get(x))
aiclist
}

gam.5 = gam(Cscore~s(lcavol,5)+s(lweight,5)+s(lcp,5)+s(lpsa,5)+svi, data = train)$aic
gam.4.1 = gam(Cscore~+s(lweight,5)+s(lcp,5)+s(lpsa,5)+svi, data = train)$aic
gam.4.2 = gam(Cscore~s(lcavol,5)+s(lcp,5)+s(lpsa,5)+svi, data = train)$aic
gam.4.3 = gam(Cscore~s(lcavol,5)+s(lweight,5)+s(lpsa,5)+svi, data = train)$aic
gam.4.4 = gam(Cscore~s(lcavol,5)+s(lweight,5)+s(lcp,5)+svi, data = train)$aic
gam.4.5 = gam(Cscore~s(lcavol,5)+s(lweight,5)+s(lcp,5)+s(lpsa,5), data = train)$aic

aic_4 = getAIC("gam.4.",5)
gam.4 = gam.4.2
gam.3.1 = gam(Cscore~s(lcp,5)+s(lpsa,5)+svi, data = train)$aic
gam.3.2 = gam(Cscore~s(lcavol,5)+s(lpsa,5)+svi, data = train)$aic
gam.3.3 = gam(Cscore~s(lcavol,5)+s(lcp,5)+svi, data = train)$aic
gam.3.4 = gam(Cscore~s(lcavol,5)+s(lcp,5)+s(lpsa,5), data = train)$aic
aic_3 = getAIC("gam.3.",4)

gam.3 = gam.3.1
gam.2.1 = gam(Cscore~s(lpsa,5)+svi, data = train)$aic
gam.2.2 = gam(Cscore~s(lcp,5)+svi, data = train)$aic
gam.2.3 = gam(Cscore~s(lcp,5)+s(lpsa,5), data = train)$aic
aic_2 = getAIC("gam.2.", 3)

gam.2 = gam.2.3
gam.1.1 = gam(Cscore~s(lpsa,5), data = train)$aic
gam.1.2 = gam(Cscore~s(lcp,5), data = train)$aic
aic_1 = getAIC("gam.1.",2)
aic_1
```

```
## [[1]]
## [1] 466
##
## [[2]]
## [1] 526
```

```
gam.1 = gam.1.1
```

To pick the best GAM using backward selection, we first manually perform backwards selection on the models. The best model with a specified number of variables is selected by using the AIC. A starting model includes every continuous variable with the degrees of freedom set to five. We use AIC instead of R^2 value since this value is readily available in the summary and we are not confident R^2 is a good measure for evaluating a GAM.

```
gam5 = gam(Cscore~s(lcavol,5)+s(lweight,5)+s(lcp,5)+s(lpsa,5)+svi, data = train)
gam4 = gam(Cscore~s(lcavol,5)+s(lcp,5)+s(lpsa,5)+svi, data = train)
```

```
gam3 = gam(Cscore~s(lcp,5)+s(lpsa,5)+svi, data = train)
gam2 = gam(Cscore~s(lcp,5)+s(lpsa,5), data = train)
gam1 = gam(Cscore~s(lpsa,5), data = train)

gam5.mse = mean((Cscore-predict(gam5, prostate))[-sample]^2)
gam4.mse = mean((Cscore-predict(gam4, prostate))[-sample]^2)
gam3.mse = mean((Cscore-predict(gam3, prostate))[-sample]^2)
gam2.mse = mean((Cscore-predict(gam2, prostate))[-sample]^2)
gam1.mse = mean((Cscore-predict(gam1, prostate))[-sample]^2)
testerrs = c(gam5.mse,gam4.mse,gam3.mse,gam2.mse,gam1.mse)
testerrs
```

```
## [1] 515 452 420 342 409
```

To select the best model out of all the models with different numbers of variables, we use cross-validation. The model with two variables has the lowest test MSE, so we used this model for the rest of our analysis, to vary the degrees of freedom of the predictor variables.

```
MSE = 5000
for (i in 1:15){
  for (j in 1:15){
    model = gam(Cscore~s(lcp,i)+s(lpsa,j), data = train)
    mse = mean((Cscore-predict(model, prostate))[-sample]^2)
    if (mse < MSE){
      i.select = i
      j.select = j
      MSE = mse
    }
  }
}
backwards_selected = gam(Cscore~s(lcp,2)+s(lpsa,3), data = train)
summary(backwards_selected)
```

```
##
## Call: gam(formula = Cscore ~ s(lcp, 2) + s(lpsa, 3), data = train)
## Deviance Residuals:
##    Min     1Q Median     3Q    Max
## -81.43 -14.81  -3.01  13.17 108.74
##
## (Dispersion Parameter for gaussian family taken to be 834)
##
##     Null Deviance: 191001 on 47 degrees of freedom
## Residual Deviance: 35048 on 42 degrees of freedom
## AIC: 467
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##            Df Sum Sq Mean Sq F value  Pr(>F)
## s(lcp, 2)   1  51115   51115    61.2 9.8e-10 ***
## s(lpsa, 3)  1  49653   49653    59.5 1.4e-09 ***
## Residuals  42  35048     834
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Anova for Nonparametric Effects
##             Npar Df Npar F   Pr(F)
## (Intercept)
## s(lcp, 2)         1   3.72   0.061 .
## s(lpsa, 3)        2  30.38 6.9e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We select the model with the lowest test MSE as the best model. The degrees of freedom of both predictor variables is varied between one and fifteen. As we can see, the best model has two degrees of freedom for lcp, and three degrees of freedom for lpsa.

We have also troubleshooted a backward selection algorithm provided as part of the gam package, in the form of the step.Gam function. The code is available in stepGamfixed.R and is not reproduced here for brevity. The result differs on whether we attempt backward selection with the full data or with a halved training set.

The full dataset takes 19 steps to converge on the model:

$$Cscore = lcavol + age + s(lcp, 4) + s(lpsa, 5)$$

The training set takes seven steps to converge on the model:

$$Cscore = lcavol + s(lweight, 5) + s(age, 5) + s(lbph, 2) + s(lcp, 5) + s(lpsa, 5)$$

This has a test MSE of 583. The differences from our implementation can probably be explained by the extra sophistication of the step.Gam implementation, but also demonstrate once again that stepwise variable selection can lead to very varying results and should normally be done extremely carefully, if at all.

Comparison to linear models

To compare our final model with the linear models, we also used the test MSE.

The linear models all have a test MSE between 650 and 900, the test mean squared error of our manually backward selected GAM model is the following:

```
backwards_selected.mse = mean((Cscore-predict(backwards_selected, prostate))[-sample]^2)
backwards_selected.mse
```

```
## [1] 245
```

It is safe to conclude that this nonlinear model is a better model than all of the linear models, since it reduces the test errors by at least 63.1% compared on the best performing linear model - lasso.