

Advances Computational Econometrics. Chapter 4: Splines and Generalized Additive Models

Thierry Denoeux

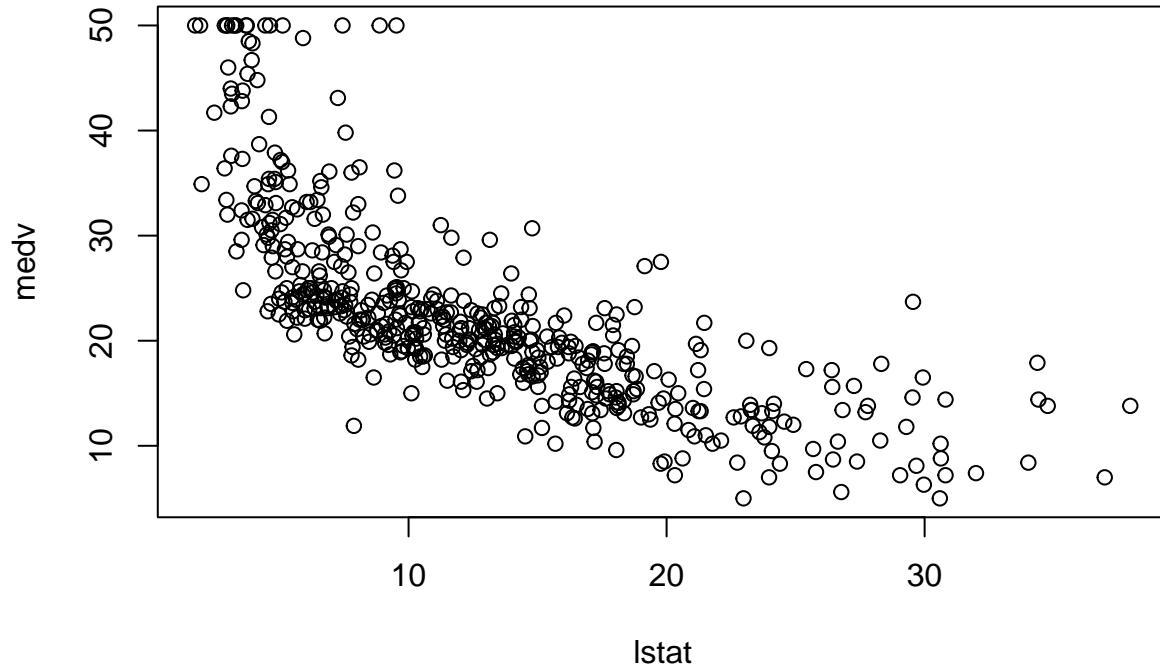
2023-05-05

Exercise 1

Question 1

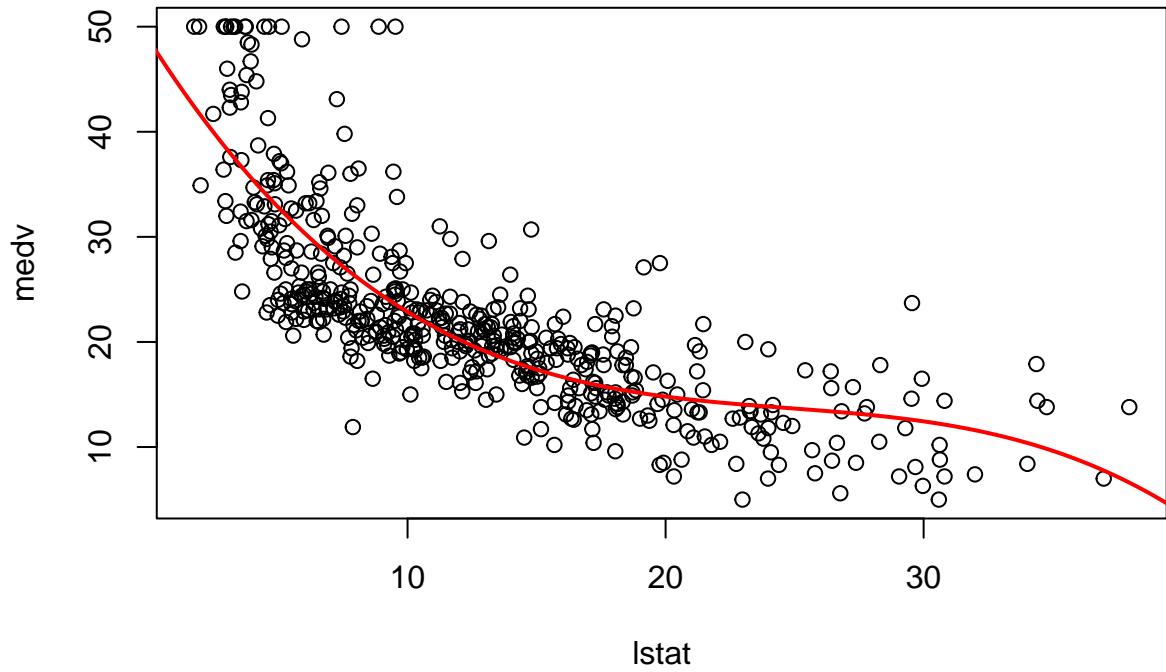
We start by loading the data and plotting medv vs. lstat:

```
library(MASS)
attach(Boston)
plot(lstat,medv)
```



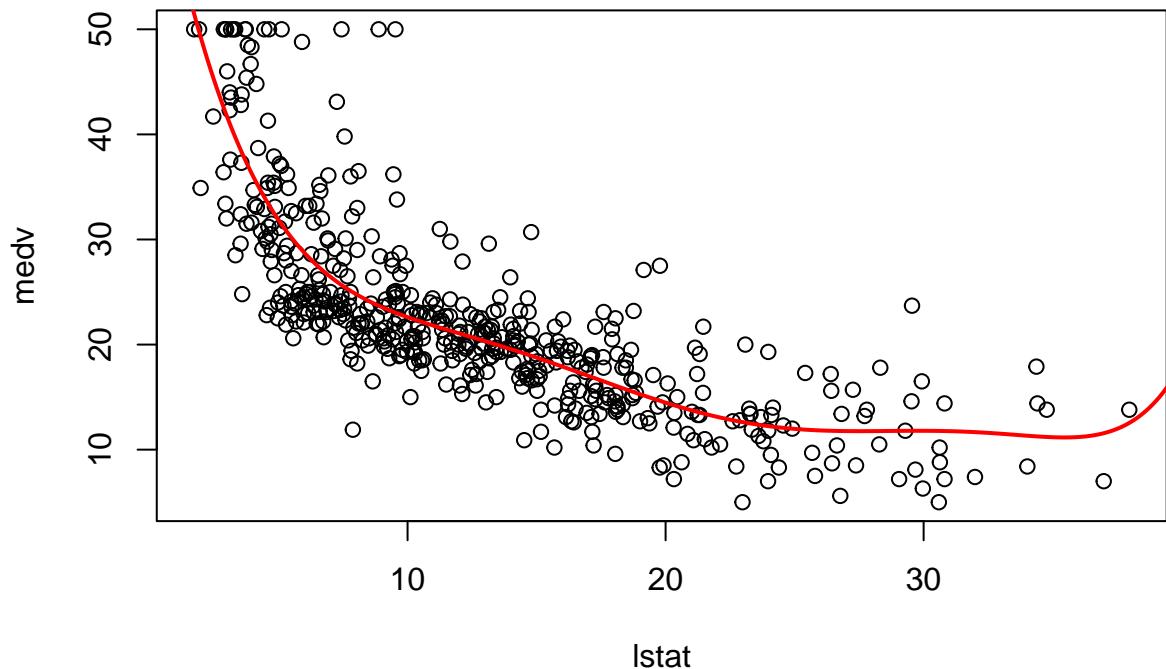
The relation is clearly nonlinear. Let fit a degree 3 polynomial and plot the result:

```
fit=lm(medv~poly(lstat,3),data=Boston)
x<-seq(min(lstat)-10,max(lstat)+10,0.1)
pred<-predict(fit,newdata=data.frame(lstat=x))
plot(lstat,medv)
lines(x,pred,col="red",lwd=2)
```



The result is clearly not optimal. We try to increase the degree to 7:

```
fit=lm(medv~poly(lstat,7),data=Boston)
pred<-predict(fit,newdata=data.frame(lstat=x))
plot(lstat,medv)
lines(x,pred,col="red",lwd=2)
```



This time, the result seems better.

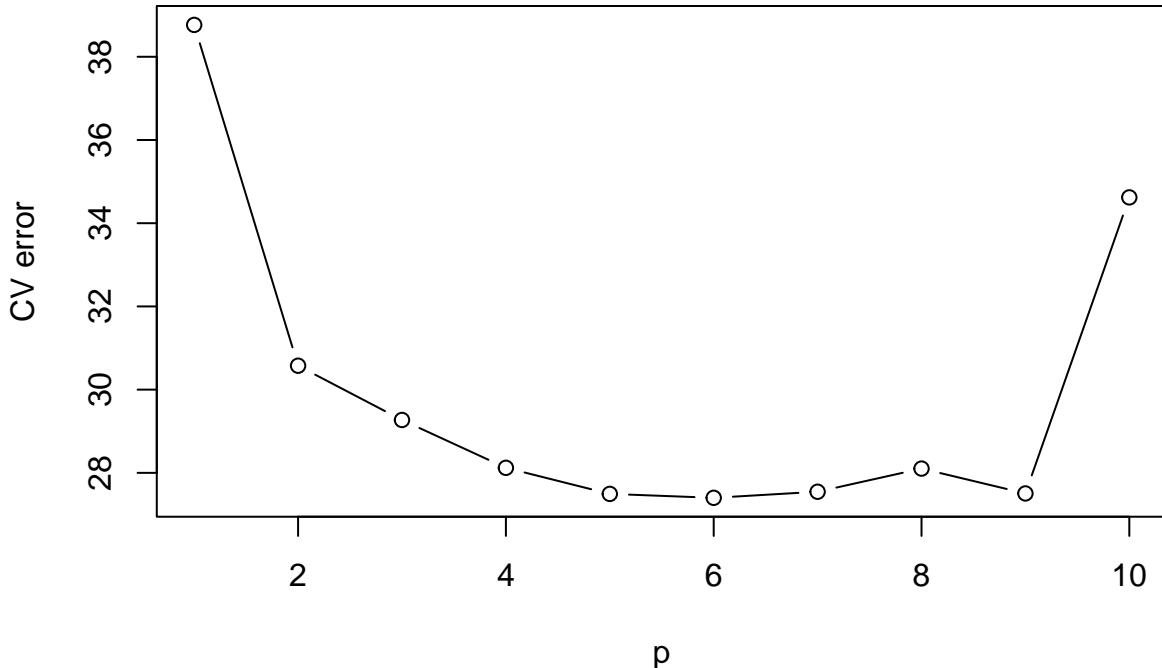
Question 2

We will use 10-fold cross-validation. We start by drawing the subsets randomly:

```
set.seed(15042022)
K<-10
n<-nrow(Boston)
folds=sample(1:K,n,replace=TRUE)
```

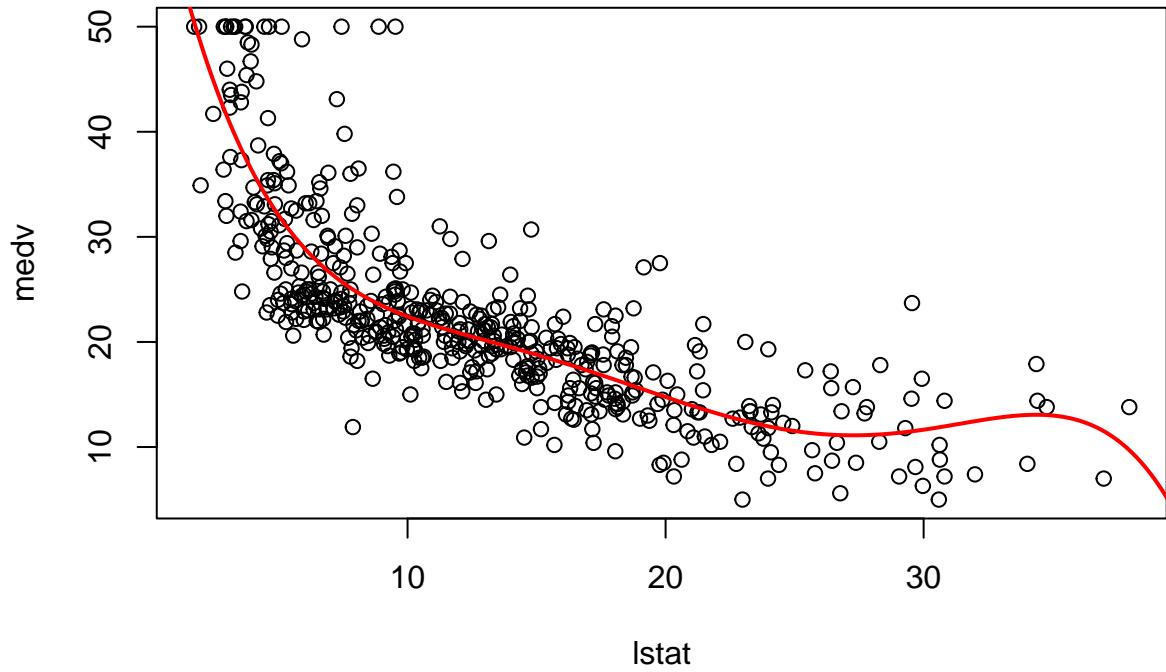
Main cross-validation loop:

```
P<-1:10
N<-length(P)
CV1<-rep(0,N)
for(i in 1:N){
  for(k in 1:K){
    fit=lm(medv~poly(lstat,P[i]),data=Boston[folds!=k,])
    pred<-predict(fit,newdata=Boston[folds==k,])
    CV1[i]<-CV1[i]+ sum((Boston$medv[folds==k]-pred)^2)
  }
  CV1[i]<-CV1[i]/n
}
plot(P,CV1,type='b',xlab='p',ylab='CV error')
```



The degree $p = 5$ is quasi-optimal. We fit this model on the whole dataset:

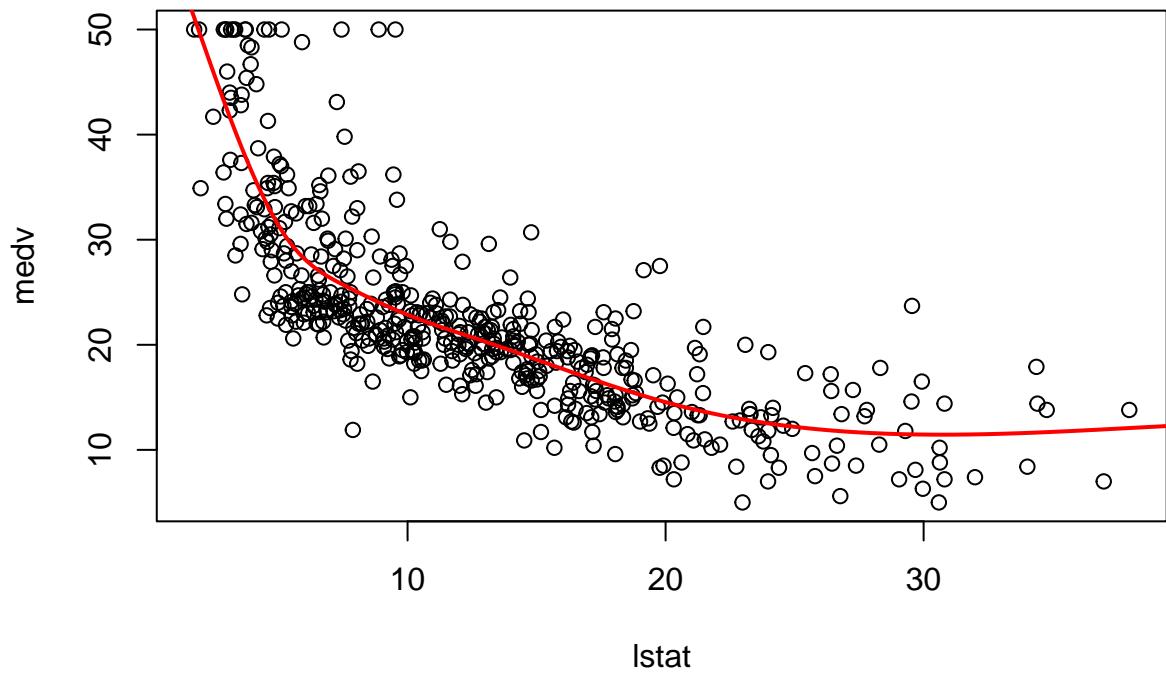
```
fit=lm(medv~poly(lstat,5),data=Boston)
pred1<-predict(fit,newdata=data.frame(lstat=x))
plot(lstat,medv)
lines(x,pred1,col="red",lwd=2)
```



Question 3

We first try a model with $df=7$ degrees of freedom:

```
library(splines)
fit=lm(medv~ns(lstat,df=7),data=Boston)
pred<-predict(fit,newdata=data.frame(lstat=x))
plot(lstat,medv)
lines(x,pred,col="red",lwd=2)
```

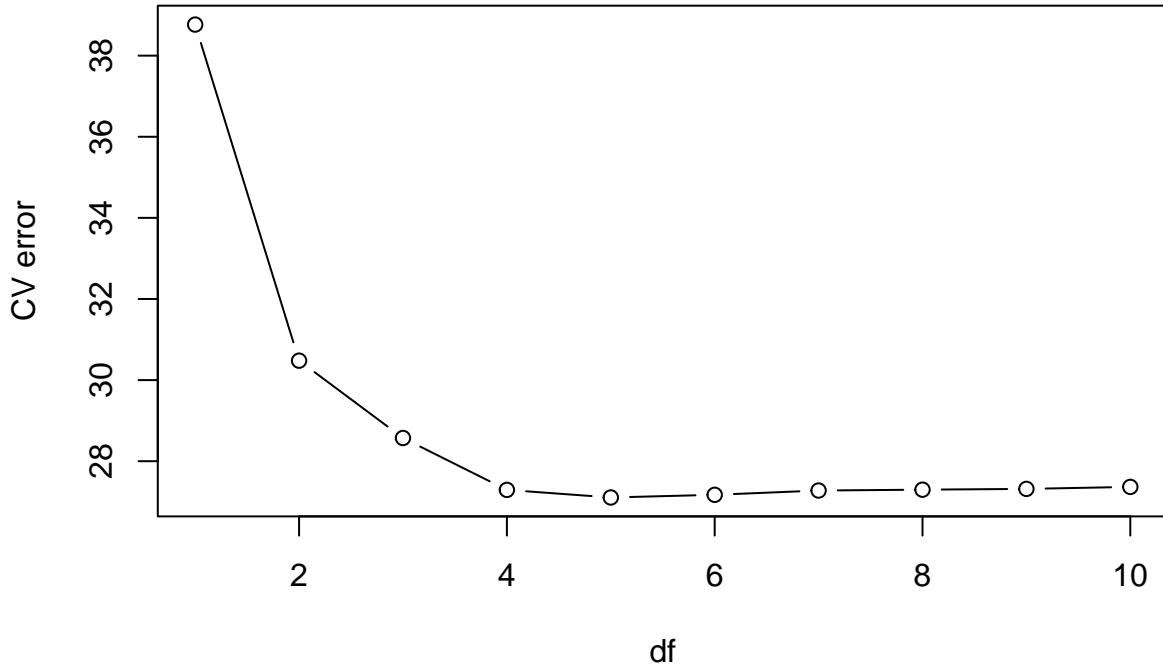


The result is clearly better than that of polynomial regression.

Cross-validation loop:

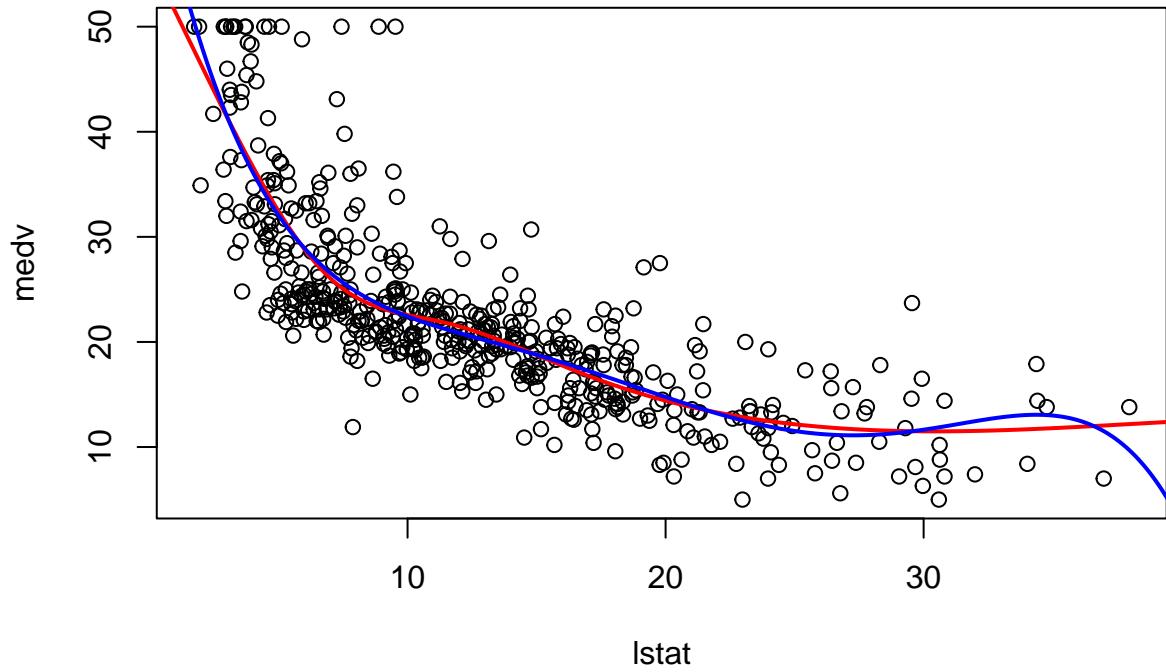
```
DF<-1:10
N<-length(DF)

CV2<-rep(0,N)
for(i in 1:N){
  for(k in 1:K){
    fit=lm(medv~ns(lstat,df=DF[i]),data=Boston[folds!=k,])
    pred<-predict(fit,newdata=Boston[folds==k,])
    CV2[i]<-CV2[i]+ sum((Boston$medv[folds==k]-pred)^2)
  }
  CV2[i]<-CV2[i]/n
}
plot(DF,CV2,type='b',xlab='df',ylab='CV error')
```



The model with $df=4$ is close to optimal. We fit this model on the whole dataset, and plot the prediction function together with that of polynomial regression

```
fit=lm(medv~ns(lstat,df=4),data=Boston)
pred2<-predict(fit,newdata=data.frame(lstat=x))
plot(lstat,medv)
lines(x,pred2,col="red",lwd=2)
lines(x,pred1,col="blue",lwd=2)
```



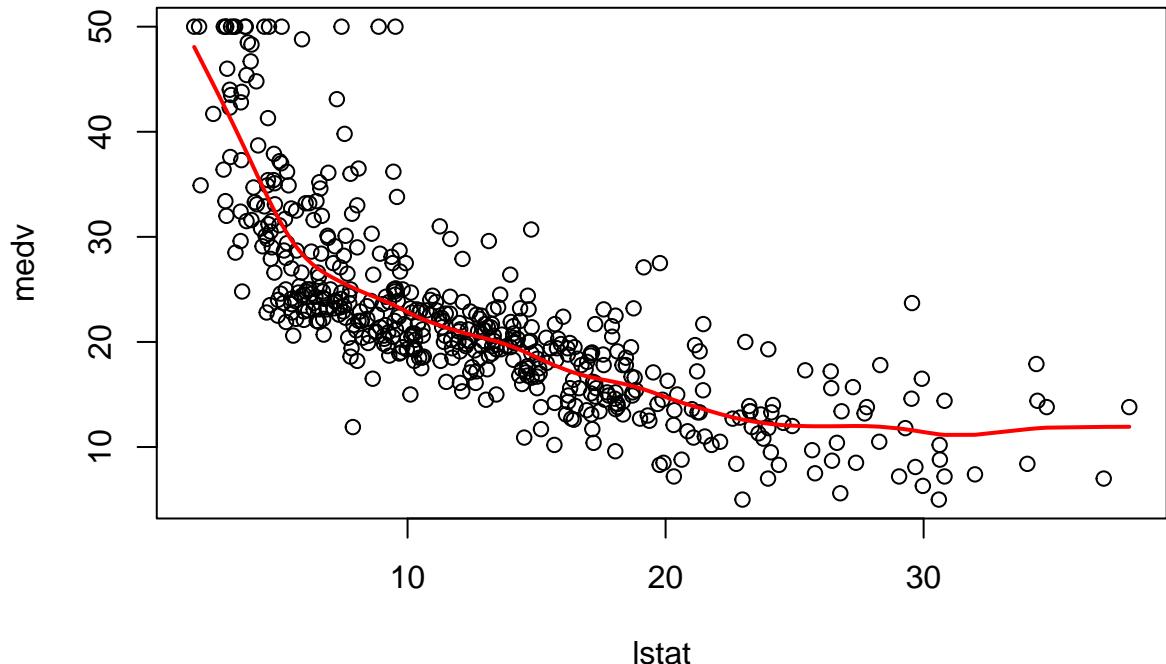
Question 4

Function has a built-in cross-validation procedure:

```
fit<-smooth.spline(Boston$lstat,Boston$medv, cv=TRUE)
dfopt1<-fit$df
print(dfopt1)

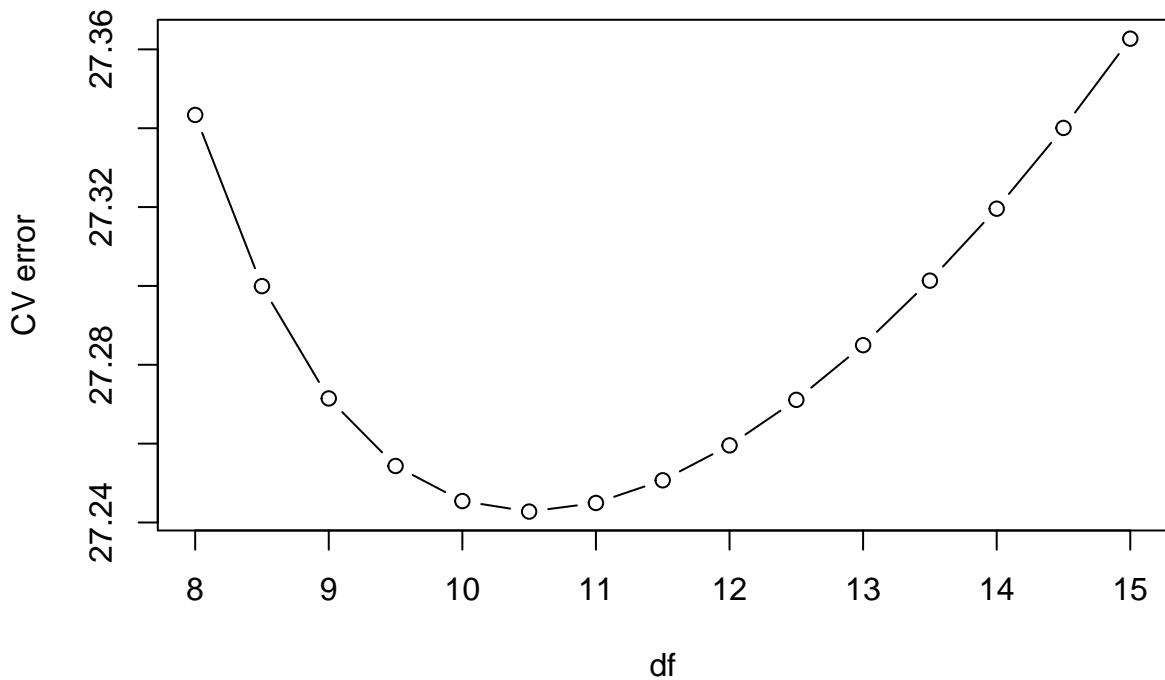
## [1] 11.3742

fit=smooth.spline(Boston$lstat,Boston$medv, df=dfopt1)
plot(lstat,medv)
lines(fit$x,fit$y,col="red",lwd=2)
```



We can also compute the cross-validation error for different degrees of freedom using the same folds as used previously:

```
DF<-seq(8,15,0.5)
N<-length(DF)
CV3<-rep(0,N)
for(i in 1:N){
  for(k in 1:K){
    fit=smooth.spline(Boston$lstat[folds!=k],Boston$medv[folds!=k],df=DF[i])
    pred<-predict(fit,Boston$lstat[folds==k])
    CV3[i]<-CV3[i]+ sum((Boston$medv[folds==k]-pred$y)^2)
  }
  CV3[i]<-CV3[i]/n
}
plot(DF,CV3,type='b',xlab='df',ylab='CV error')
```

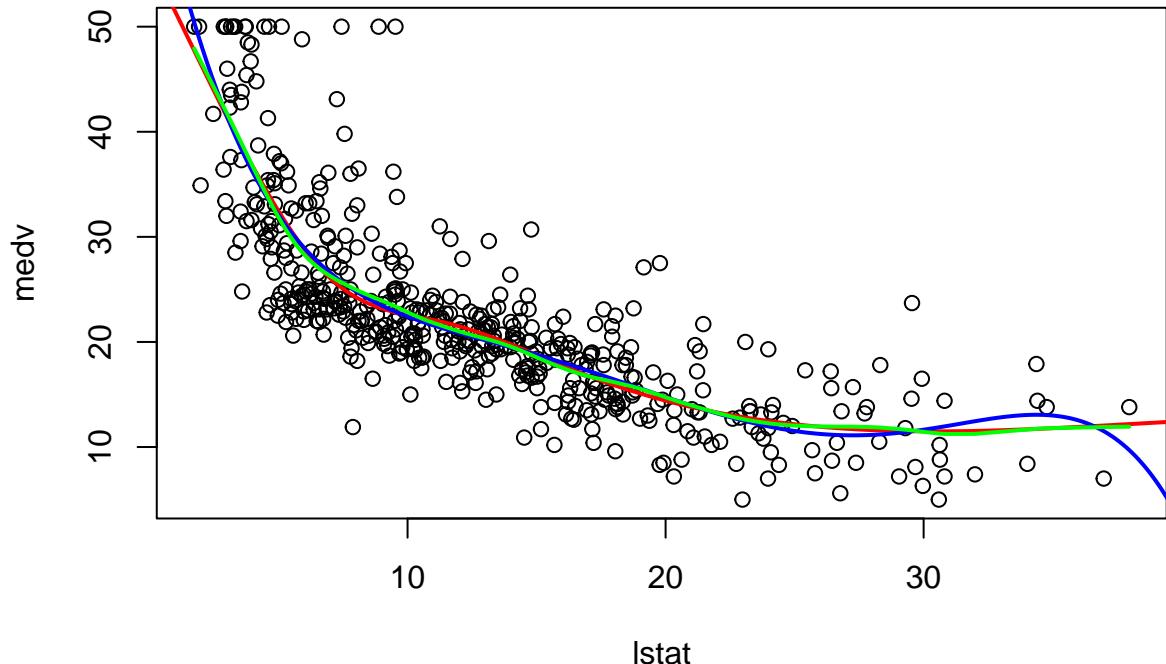


```
dfopt<-DF[which.min(CV3)]
print(dfopt)
```

```
## [1] 10.5
```

We can now plot the three prediction functions on the same graph:

```
fit=smooth.spline(Boston$lstat,Boston$medv,df=dfopt)
pred3<-predict(fit,newdata=data.frame(lstat=x))
plot(lstat,medv)
lines(x,pred2,col="red",lwd=2)
lines(x,pred1,col="blue",lwd=2)
lines(pred3$x,pred3$y,col="green",lwd=2)
```



The natural cubic spline and smoothing splines solutions are very similar.

Exercise 2

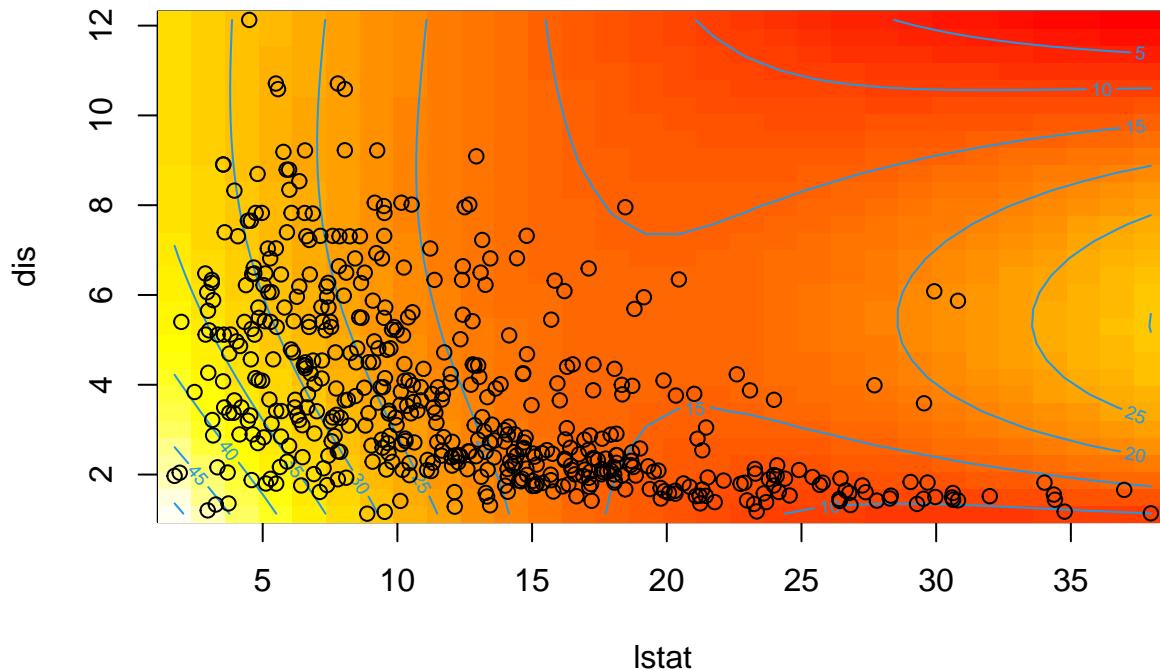
We use function `gam` from package `mgcv`, with three basis functions along each dimension:

```
library(mgcv)
fit<-gam(medv ~ te(lstat,dis,k=3),data=Boston)
```

Generation of the contour plot:

```
vis.gam(fit,plot.type = "contour",color="heat")
points(lstat,dis,pch=1)
```

linear predictor



There seems to be some interaction between variables `lstat` and `dis`: the median house value seems to decrease with distance to main employment centers when `lstat` is small, but the inverse tendency is observed for larger values of `lstat`. This phenomenon remains to be explained.

We detach package `mgcv`, because in the next exercise we will use function `gam` of package `gam`:

```
detach("package:mgcv")
```

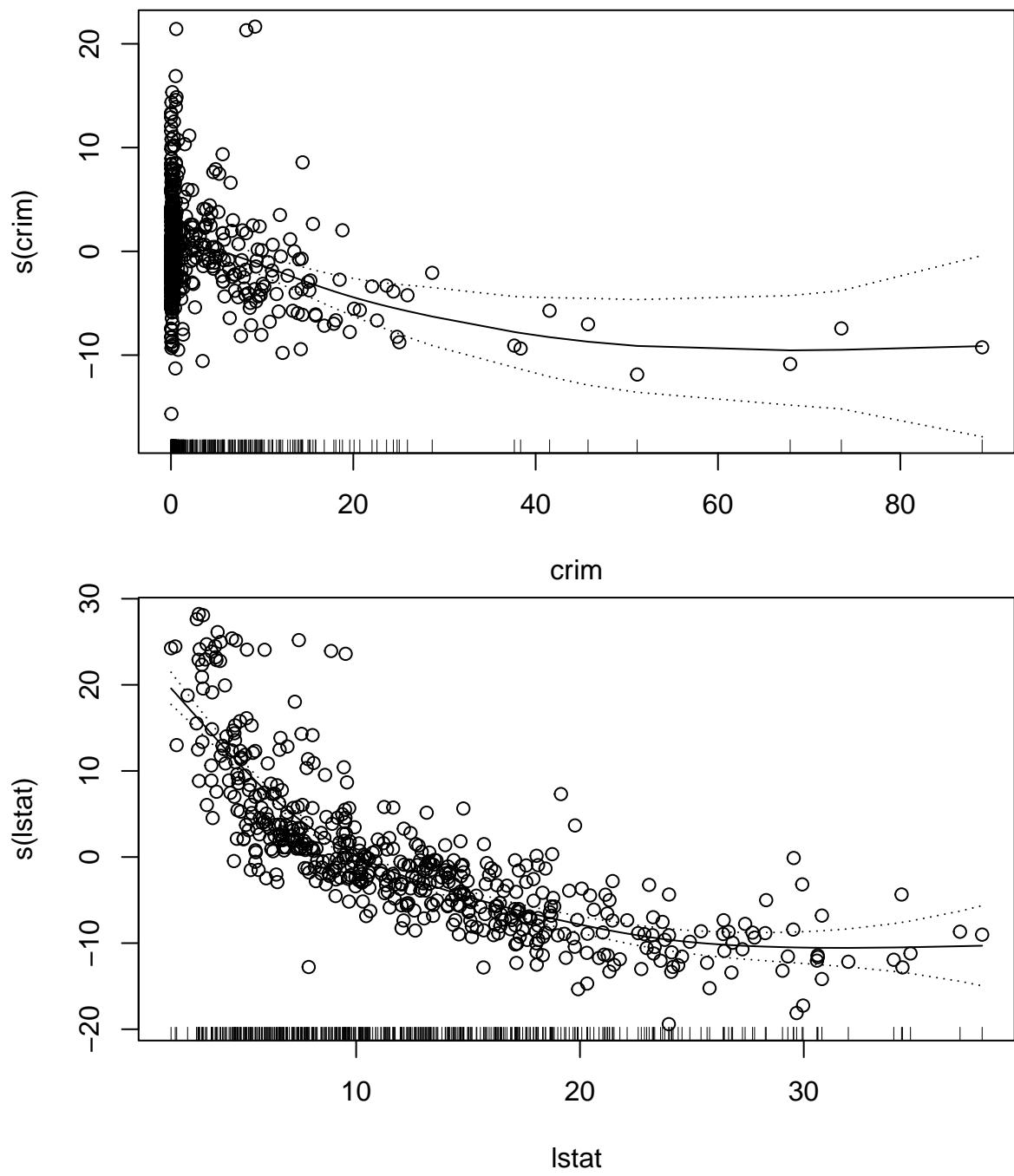
Exercise 3

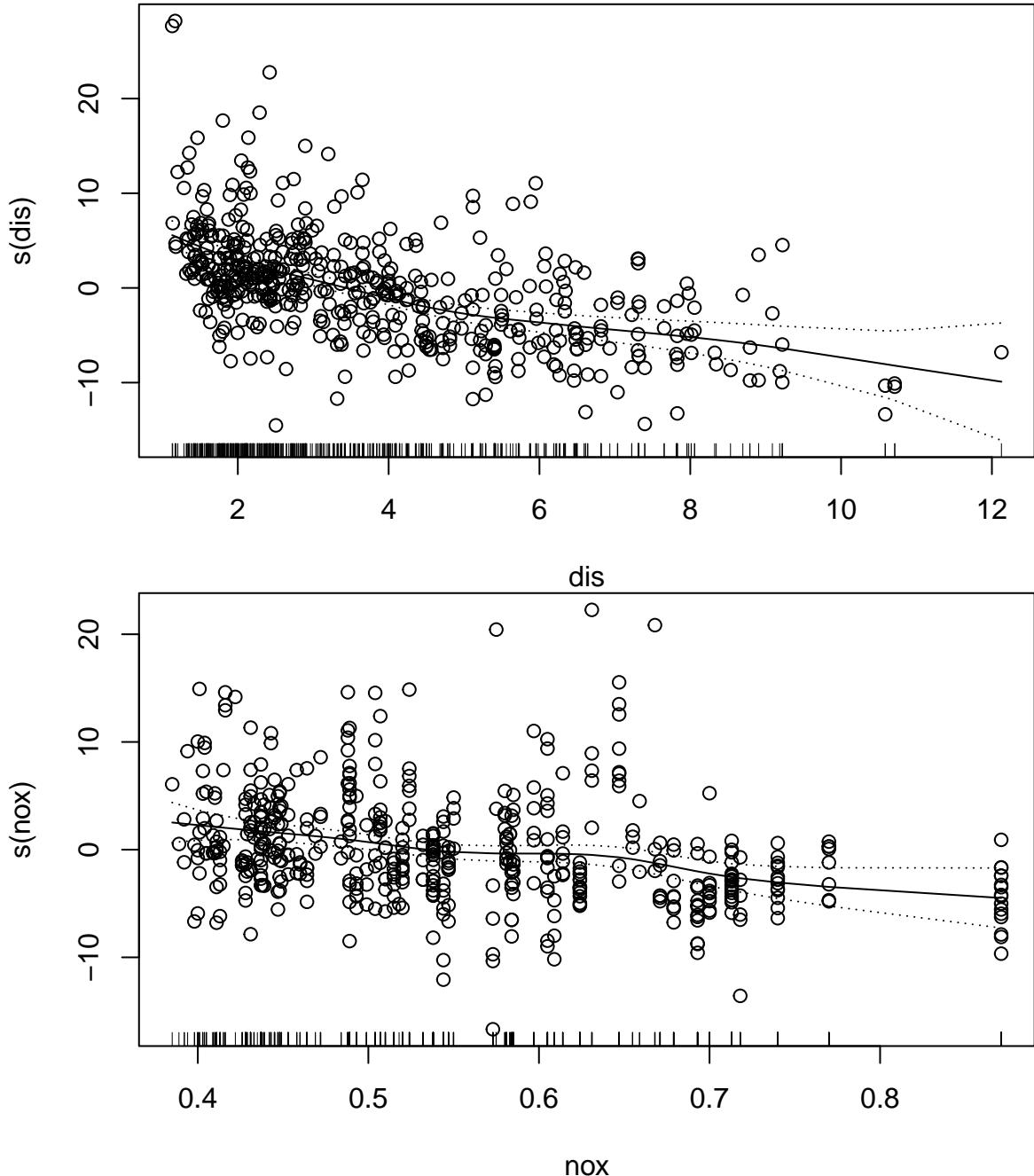
Question 1

We first load package `gam` and fit the model with default parameters:

```
library(gam)
fit<-gam(medv ~ s(crim) + s(lstat) + s(dis) + s(nox), data=Boston, trace=TRUE)

## GAM s.wam loop 1: deviance = 11568.87
## GAM s.wam loop 2: deviance = 11568.87
plot(fit, residuals=TRUE, se=TRUE)
```





We can see that the value of houses decreases with increased crime rate, low status of the population, distance to employment centers, and nox pollution.

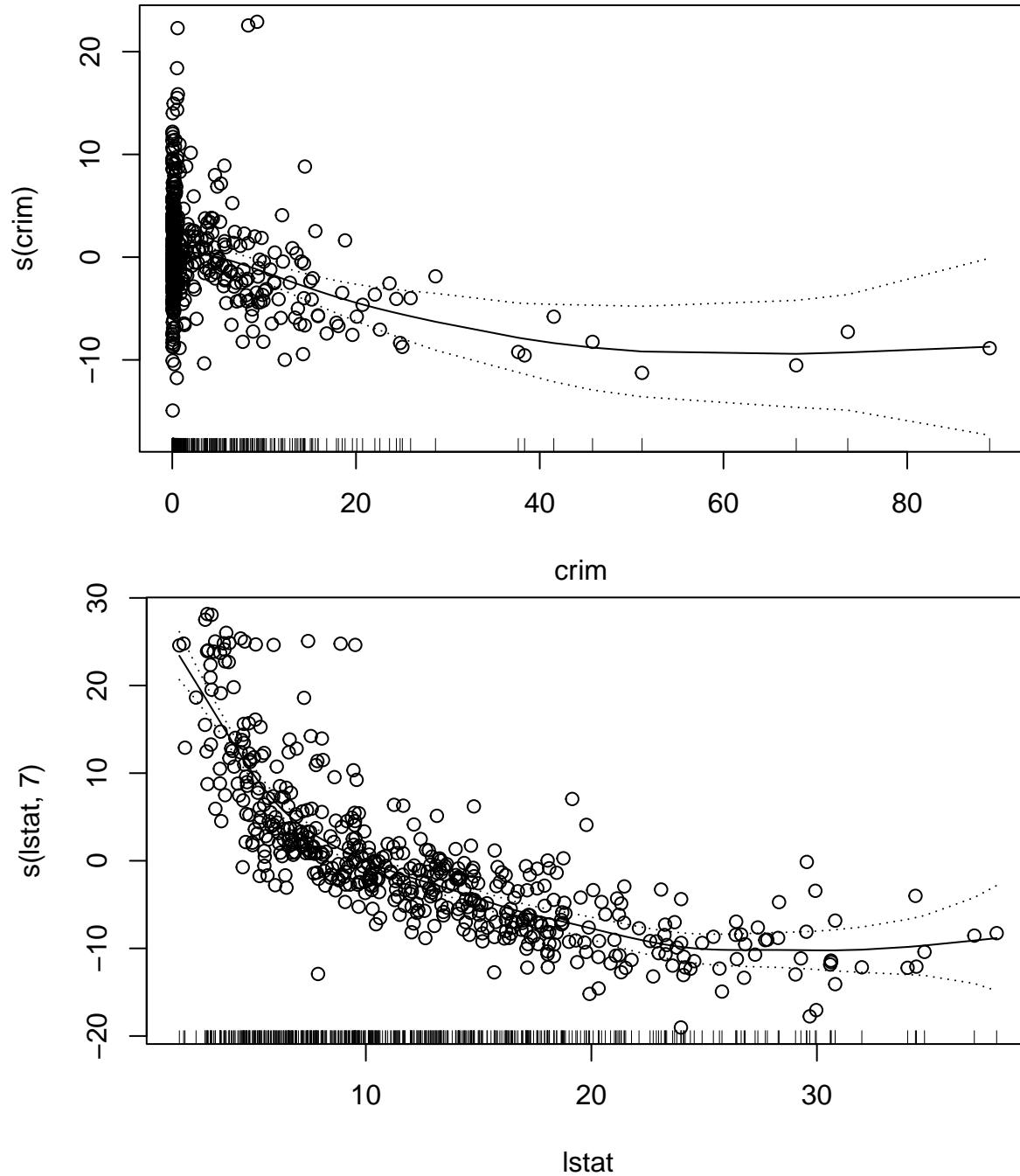
Question 2

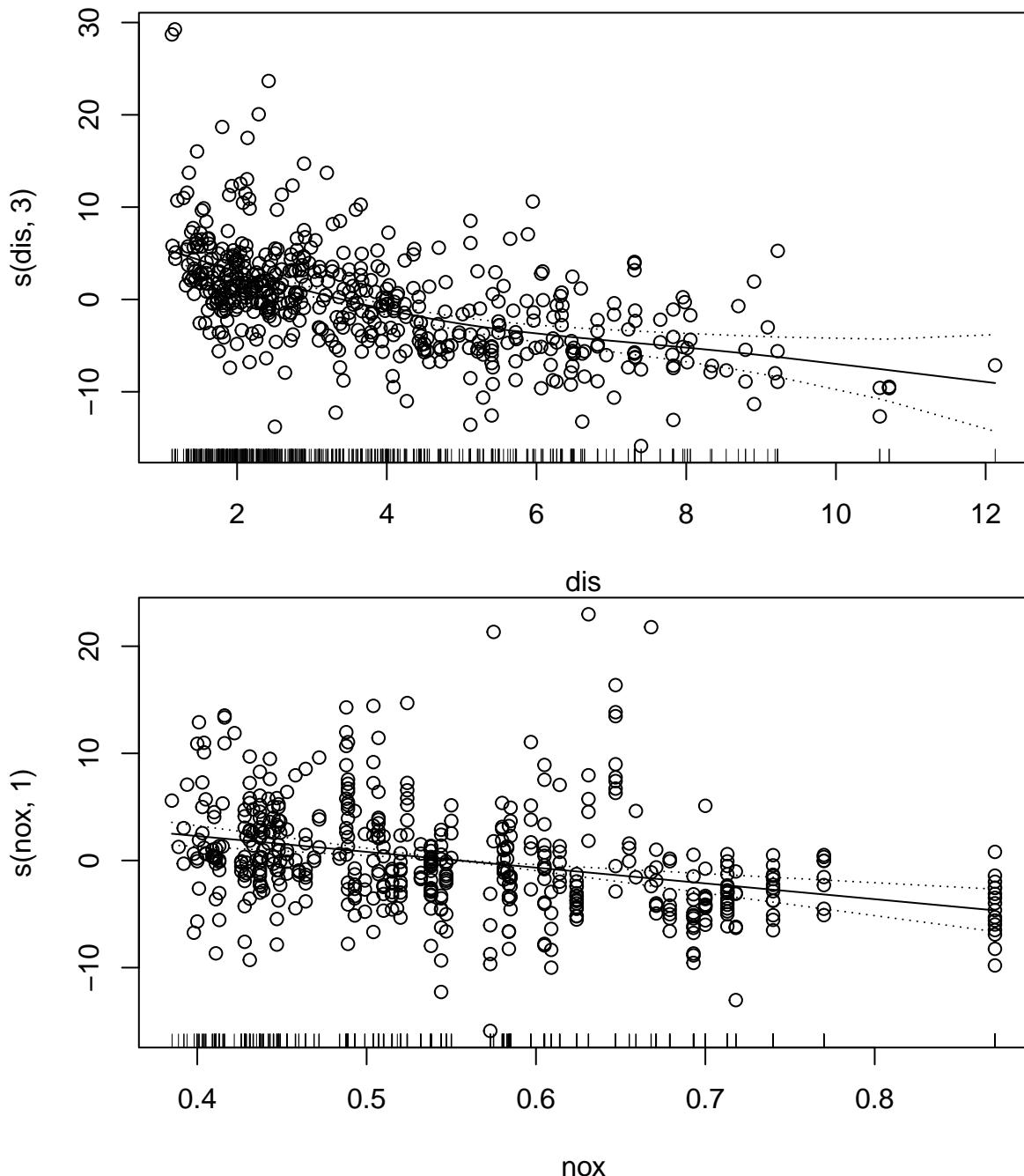
It seems that the degrees of freedom could be increased for the term related to `lstart` and slightly decreased for that related to `dis`, while the influence of `nox` could be captured with a linear term:

```
fit<-gam(medv ~ s(crim) + s(lstat,7) + s(dis,3) + s(nox,1),data=Boston,trace=TRUE)
```

```
## GAM s.wam loop 1: deviance = 11300.09
## GAM s.wam loop 2: deviance = 11300.09
```

```
plot(fit, residuals=TRUE, se=TRUE)  
## Warning in pf(nl.chisq/nldf, nldf, rdf): Production de NaN
```





Question 3

We start by creating the $K = 10$ folds for ten-fold cross-validation:

```
K<-10
folds=sample(1:K,nrow(Boston),replace=TRUE)
```

Calculation of cross-validation MSEs with standard errors:

```
err<-matrix(0,10,4)
for(k in 1:K){
```

```

fit1<-gam(medv ~ s(crim) + s(lstat) + s(dis) + s(nox),data=Boston[folds!=k,])
fit2<-gam(medv ~ s(crim) + s(lstat,7) + s(dis,3) + s(nox,1),data=Boston[folds!=k,])
fit3<-lm(medv ~ ns(crim,4) + ns(lstat,7) + ns(dis,3) + ns(nox),data=Boston[folds!=k,])
fit4<-lm(medv ~ crim + lstat + dis + nox,data=Boston[folds!=k,])
pred1<-predict(fit1,Boston[folds==k,])
pred2<-predict(fit2,Boston[folds==k,])
pred3<-predict(fit3,Boston[folds==k,])
pred4<-predict(fit4,Boston[folds==k,])
err[k,1]<-mean((Boston$medv[folds==k]-pred1)^2)
err[k,2]<-mean((Boston$medv[folds==k]-pred2)^2)
err[k,3]<-mean((Boston$medv[folds==k]-pred3)^2)
err[k,4]<-mean((Boston$medv[folds==k]-pred4)^2)
}
print(colMeans(err))

## [1] 24.03718 23.34793 23.52693 35.54325
print(apply(err,2,sd)/sqrt(10))

## [1] 3.291852 3.459968 3.617686 4.289294

```