ST464 - Assignment 4 - Solutions

Sean O'Riogain (18145426)

30 April 2019

```
knitr::opts_chunk$set(echo = TRUE)
getwd()
```

```
## [1] "C:/Users/oriogain/Dropbox/Maynooth/ST674 - Machine Learning/Assignments"
```

```
suppressPackageStartupMessages(library(tidyverse))  # Needed for dplyr functions etc.
suppressPackageStartupMessages(library(MASS))       # Needed for Boston data etc.
suppressPackageStartupMessages(library(splines))    # Needed for bs() etc.
suppressPackageStartupMessages(library(gam))        # Needed for gam() etc.
suppressPackageStartupMessages(library(tree))       # Needed for tree() etc.
suppressPackageStartupMessages(library(glmnet))     # Needed for glmnet() etc.
```

Question 1

For the Boston data available in package MASS we wish to relate dis (weighted mean of distances to five Boston employment centres) to nox (nitrogen oxides concentration in parts per 10 million).

```
data("Boston")
str(Boston)
```

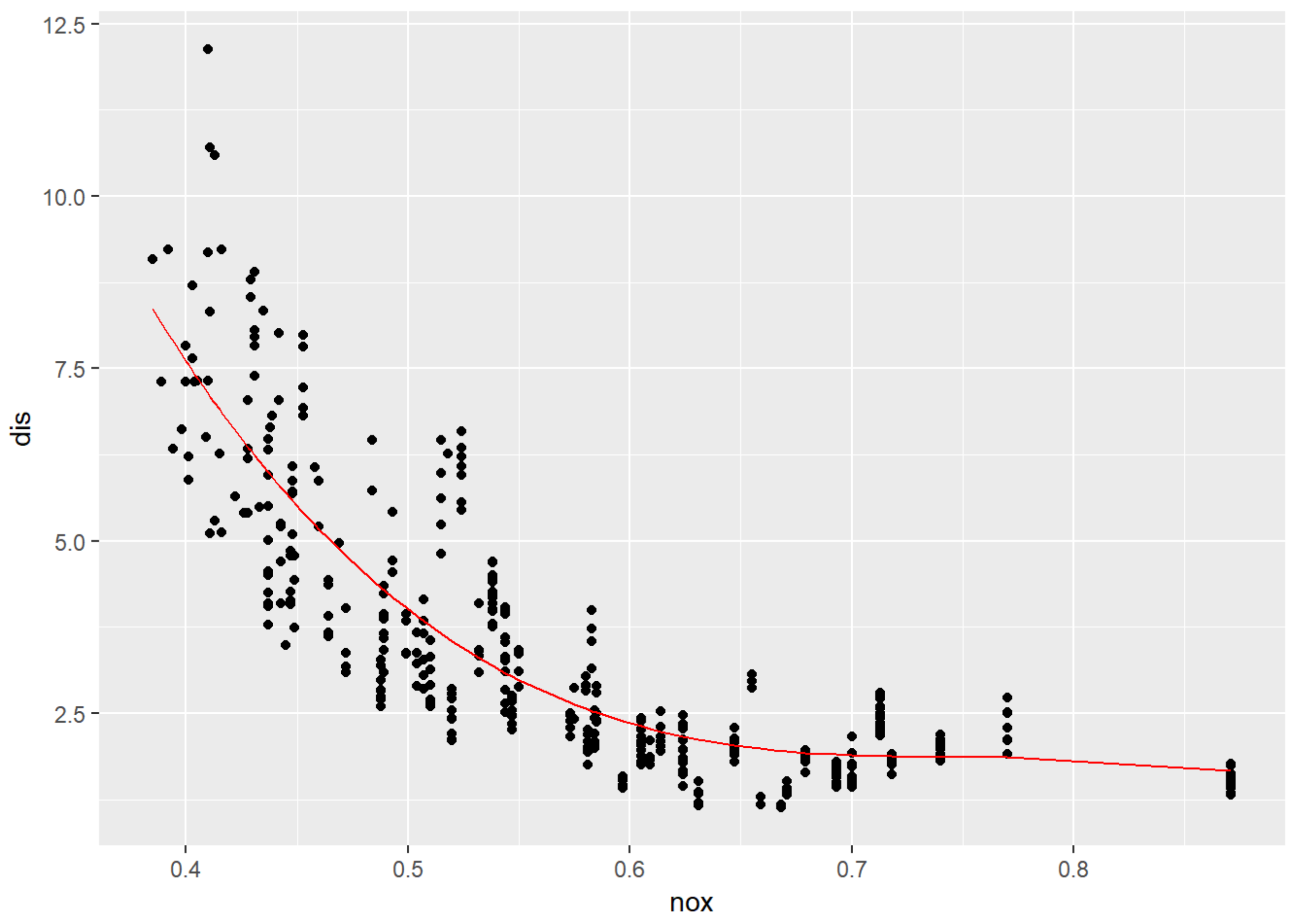
```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

a. Fit a cubic polynomial to the data.

```
f.q1a<-lm(dis ~ nox + I(nox^2) + I(nox^3), data=Boston)
```

Plot the data and the fit.

```
ggplot(data=Boston, aes(x=nox, y=dis)) + geom_point() +
  geom_line(aes(y=fitted(f.q1a)), color="red")
```



Comment on the fit.

The fit appears to be pretty good except for the highest values of dis, where it bypasses those data points.

Calculate the MSE.

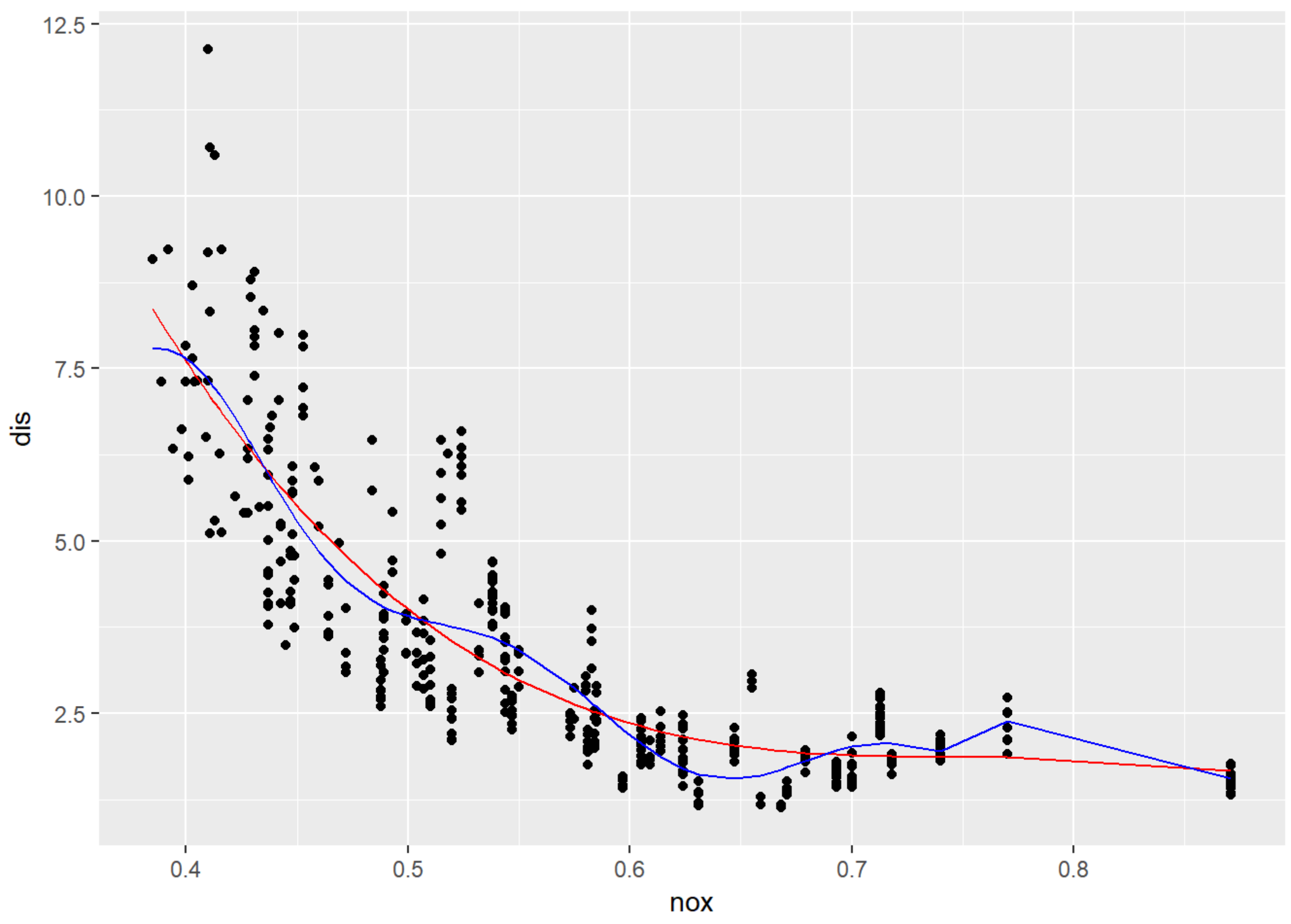
```
mse.q1a<-mean(f.q1a$residuals^2)
mse.q1a
```

```
## [1] 1.094805
```

b. Repeat (a), this time using a 10th degree polynomial.

```
f.q1b<-lm(dis~nox+I(nox^2)+I(nox^3)+I(nox^4)+I(nox^5)+I(nox^6)+I(nox^7)+
            I(nox^8)+I(nox^9)+I(nox^10), data=Boston)

ggplot(data=Boston, aes(x=nox, y=dis)) + geom_point() +
  geom_line(aes(y=fitted(f.q1a)), colour="red") +
  geom_line(aes(y=fitted(f.q1b)), colour="blue")
```



```
mse.q1b<-mean(f.q1b$residuals^2)
mse.q1b
```

```
## [1] 1.028995
```

Compare the fits and the MSE.

The new (blue) fit appears to be better than the previous cubic (red) one except for the highest values of dis, where it curves sharply away from those data points. The lower MSE value for the blue fit seems to confirm this.

Use anova to compare the two fits and comment on your findings.

```
anova(f.q1a,f.q1b)
```

```
## Analysis of Variance Table
##
## Model 1: dis ~ nox + I(nox^2) + I(nox^3)
## Model 2: dis ~ nox + I(nox^2) + I(nox^3) + I(nox^4) + I(nox^5) + I(nox^6) +
##     I(nox^7) + I(nox^8) + I(nox^9) + I(nox^10)
##   Res.Df    RSS Df Sum of Sq     F    Pr(>F)
## 1    502 553.97
## 2    496 520.67  6      33.3 5.287 2.681e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results above indicate that the additional terms (nox^4 to nox^{10}) make a statistically signifcant difference to the model fit and, therefore, should be retained at least (if not added to).

c. Describe how you might use cross-validation to select the optimal degree (say between 1 and 10).

It should be possible to create a function to build and fit the required polynomial expression based on a passed parameter which specifies the required order number (integer) and variables. Then invoke it iteratively, passing order values in the required range, to store the resultant MSE value of each fit in an array. The number of the array element with the lowest (minimum) MSE value would then give the optimal order (degree) number. We could then use Anova to compare the fit at that optimal degree with that achieved for one of our earlier attempts at assessing goodness of fit at a lower degree to confirm the goodness of optimal fit statistically.

d. Carry out the cross-validation procedure.

```
# This function builds the regression formula for the required polynomial expression.
cv.formula<-function(x, y, o=2){
  p<-paste(y," ~ ",x,sep="")

  if(o > 1){
    for(i in c(2:o)){
      p<-paste(p," + I(",x,"^",i,")",sep="")
    }
  }

  return(p)
}

# This function performs the regression for the required polynomial expression
cv.fit<-function(x, y, o=2){
  f<-lm(formula=cv.formula(x,y,o), data=Boston)

  return(f)
}

# This function performs the regression for the required polynomial expression and calculates its MSE
cv.mse<-function(x, y, o=2){
  mse<-mean(residuals(cv.fit(x,y,o))^2)

  return(mse)
}

# This loop iteratively performs regression for the polynomial expressions with the specified range of
 degrees (1 to 100 in this case)
for(i in c(1:100)){
  ifelse(i==1,mse<-cv.mse("nox","dis",i),
           mse<-c(mse,cv.mse("nox","dis",i)))
}

head(mse)
```

```
## [1] 1.806764 1.169503 1.094805 1.093300 1.072423 1.072342
```

Note that when iterating 10 times, the 10th order polynomial had the lowest MSE value. Therefore, I decided to try iterating 100 times to see where the ultimate lowest MSE might be found (see the result below).

What is the optimal degree?

```
min(which(mse==min(mse)))
```

```
## [1] 54
```

Now let’s use Anova to see if the fit for this degree is statistically better than that for a degree that we looked at earlier (the 10th degree was the best fit we found prior to this cross validation exercise).

```
anova(cv.fit("nox","dis",10),cv.fit("nox","dis",54))
```

```
## Analysis of Variance Table
##
## Model 1: dis ~ nox + I(nox^2) + I(nox^3) + I(nox^4) + I(nox^5) + I(nox^6) +
##     I(nox^7) + I(nox^8) + I(nox^9) + I(nox^10)
## Model 2: dis ~ nox + I(nox^2) + I(nox^3) + I(nox^4) + I(nox^5) + I(nox^6) +
##     I(nox^7) + I(nox^8) + I(nox^9) + I(nox^10) + I(nox^11) +
##     I(nox^12) + I(nox^13) + I(nox^14) + I(nox^15) + I(nox^16) +
##     I(nox^17) + I(nox^18) + I(nox^19) + I(nox^20) + I(nox^21) +
##     I(nox^22) + I(nox^23) + I(nox^24) + I(nox^25) + I(nox^26) +
##     I(nox^27) + I(nox^28) + I(nox^29) + I(nox^30) + I(nox^31) +
##     I(nox^32) + I(nox^33) + I(nox^34) + I(nox^35) + I(nox^36) +
##     I(nox^37) + I(nox^38) + I(nox^39) + I(nox^40) + I(nox^41) +
##     I(nox^42) + I(nox^43) + I(nox^44) + I(nox^45) + I(nox^46) +
##     I(nox^47) + I(nox^48) + I(nox^49) + I(nox^50) + I(nox^51) +
##     I(nox^52) + I(nox^53) + I(nox^54)
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    496 520.67
## 2    490 502.07  6    18.602 3.0258 0.00649 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Anova results above appear to confirm that the fit using optimal degree value (54) is statisitically better than the best fit achieved previously (using a 10th degree polynomial expression).

e. Use bs() to fit a regression spline with 4 degrees of freedom.

```
f.q1e<-lm(dis~bs(nox,4),data=Boston)
```

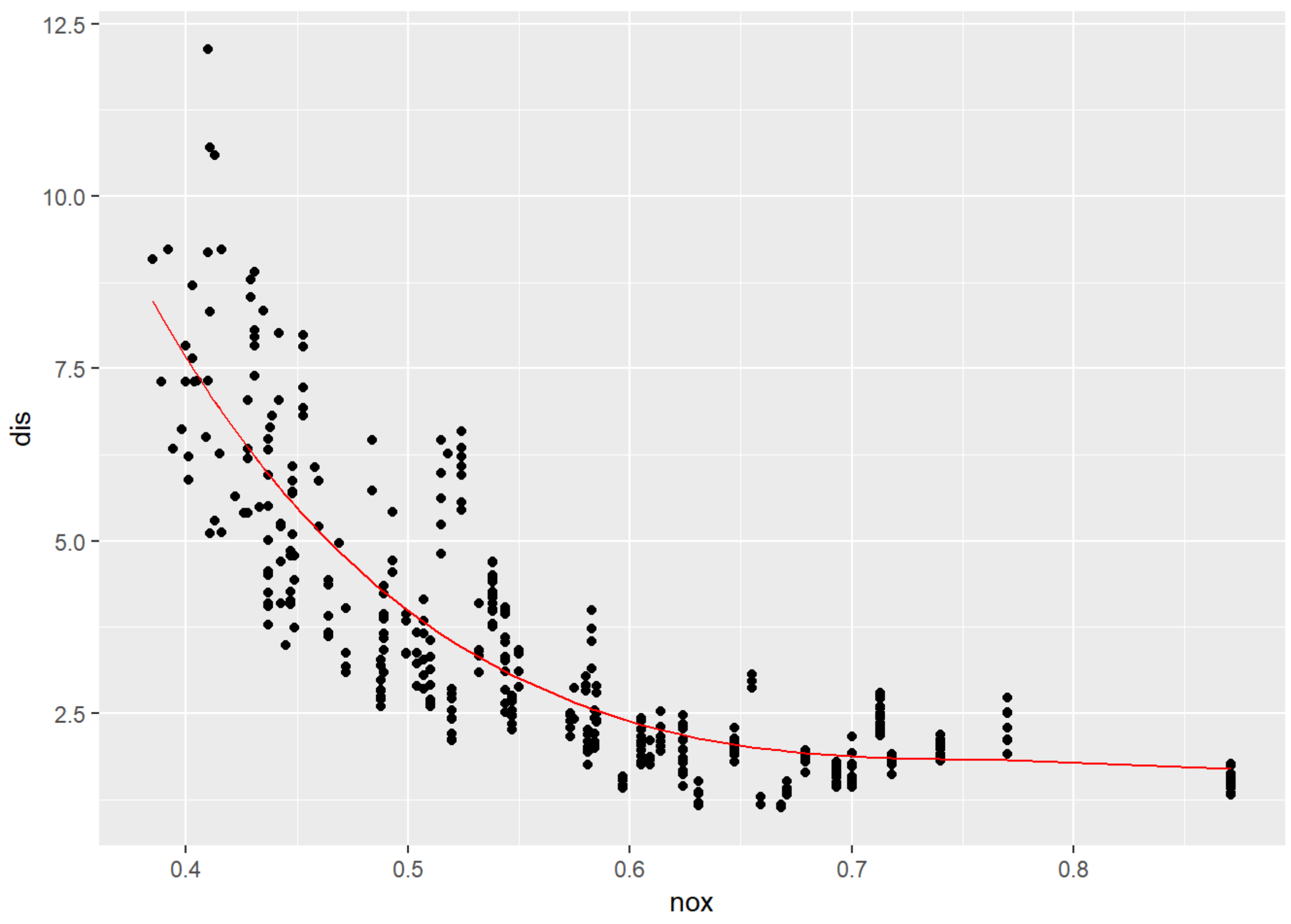
What are the knots used?

```
attr(bs(Boston$nox, df=4), "knots")
```

```
##   50%
## 0.538
```

Plot the data and the fit.

```
ggplot(data=Boston, aes(x=nox, y=dis)) + geom_point() +
  geom_line(aes(y=fitted(f.q1e)), colour="red")
```



Comment on the fit.

The fit is pretty good except for the highest values of dis, where it bypasses those data points.

Calculate the MSE.

```
mean(f.q1e$residuals^2)
```

```
## [1] 1.094149
```

f. Fit a curve using a smoothing spline with the automatically chosen amount of smoothing.

```
f.q1f<-smooth.spline(Boston$nox,Boston$dis, cv=T)
```

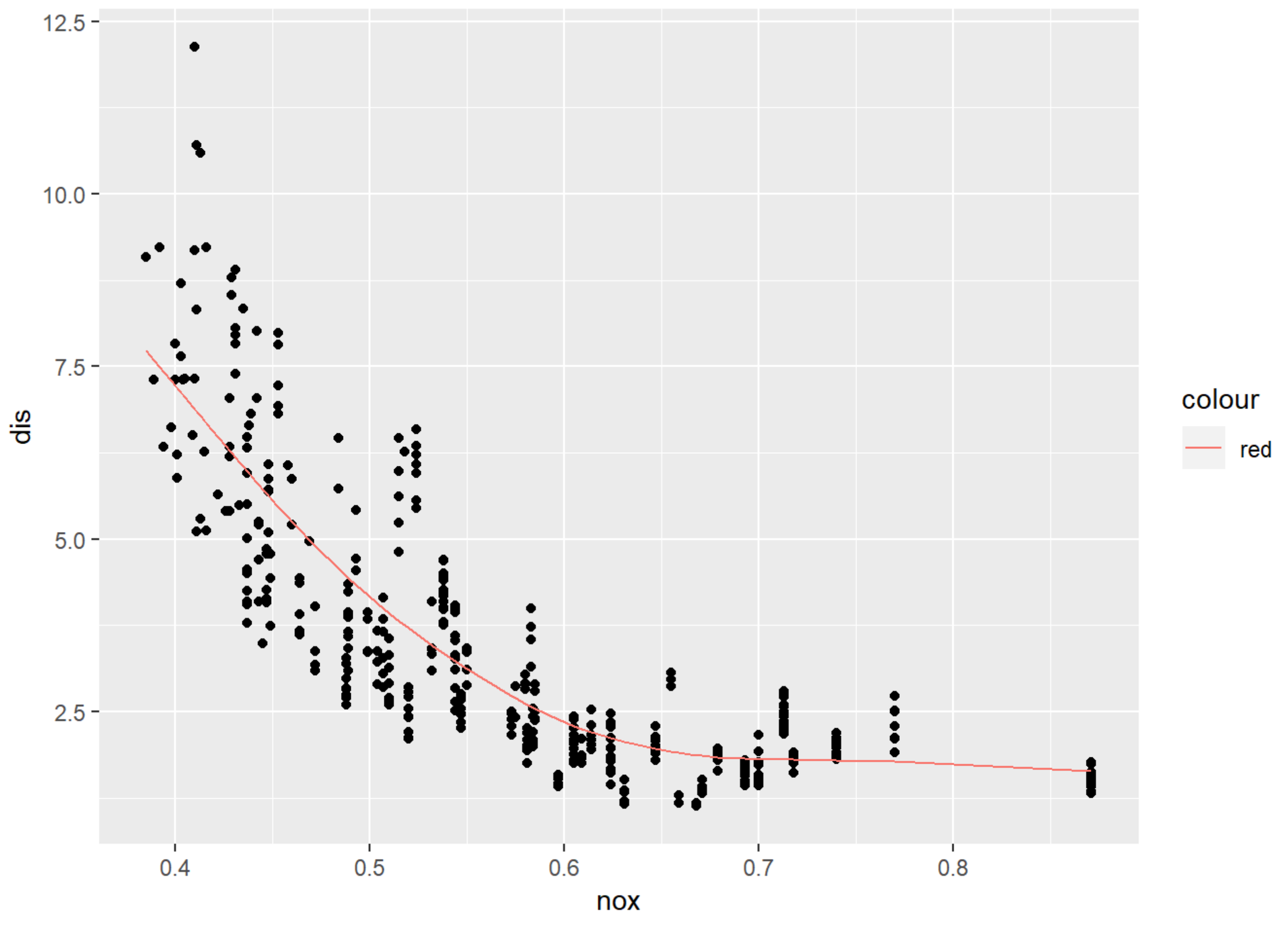
```
## Warning in smooth.spline(Boston$nox, Boston$dis, cv = T): cross-validation
## with non-unique 'x' values seems doubtful
```

```
f.q1f
```

```
## Call:
## smooth.spline(x = Boston$nox, y = Boston$dis, cv = T)
##
## Smoothing Parameter  spar= 1.050661  lambda= 0.07031691 (16 iterations)
## Equivalent Degrees of Freedom (Df): 4.128915
## Penalized Criterion (RSS): 407.277
## PRESS(l.o.o. CV): 1.119355
```

Display the fit.

```
ggplot(data=Boston, aes(x=nox, y=dis)) + geom_point() +
  geom_line(aes(y=fitted(f.q1f), colour="red"))
```



Does the automatic fit give a good result?

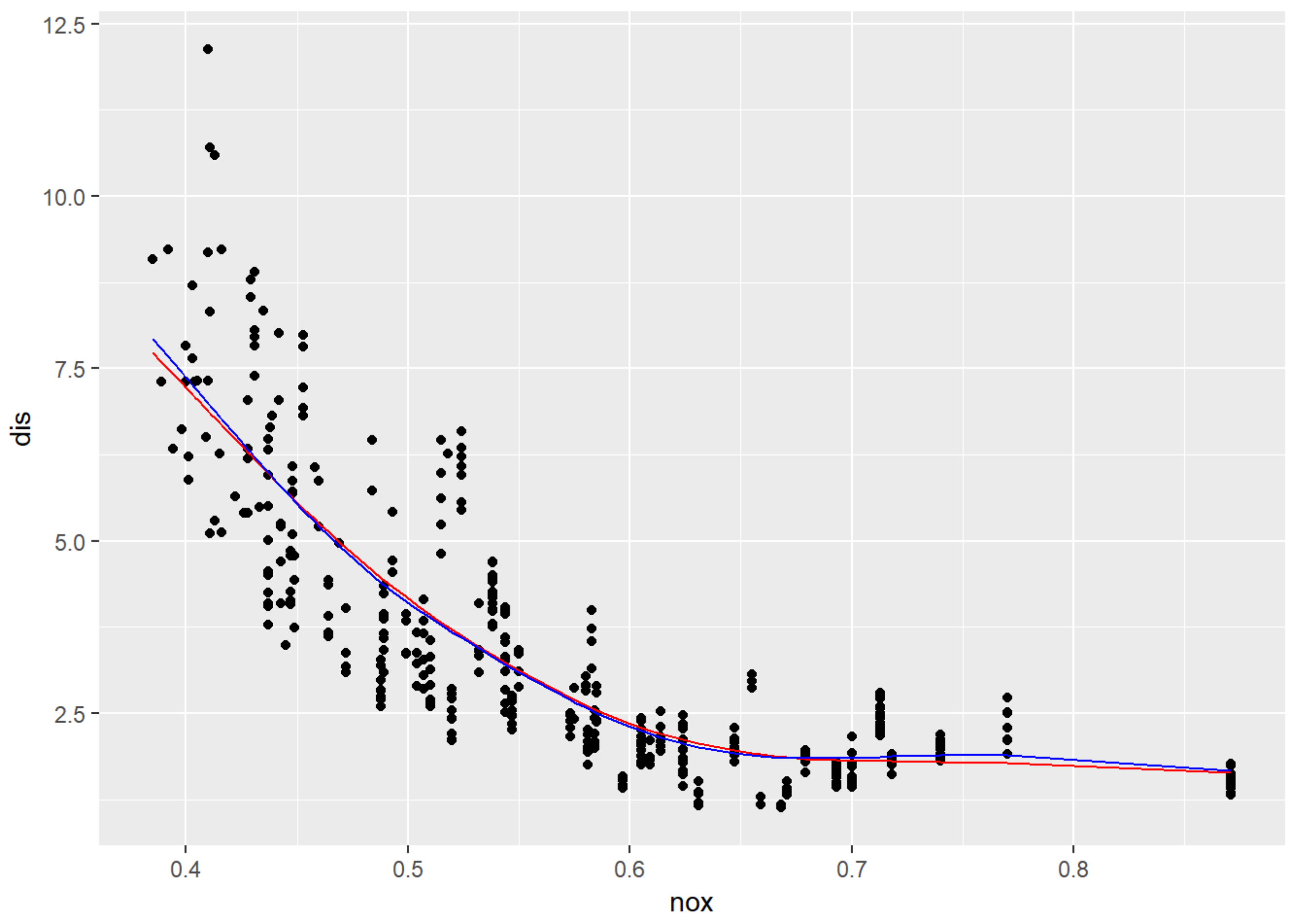
This apears to be a pretty good fit except for the highest values of dis, where it bypasses those data points.

g. Now use smoothing spline with a larger value of spar.

```
f.q1g<-smooth.spline(Boston$nox,Boston$dis,spar=1)
```

Overlay both smoothing spline fits on the plot.

```
ggplot(data=Boston,aes(x=nox,y=dis)) + geom_point() +
  geom_line(aes(y=fitted(f.q1f)), colour="red") +
  geom_line(aes(y=fitted(f.q1g)), colour="blue")
```



Which looks better?

The (blue) fit with the higher spar value appears to be slightly better than the previous (red) one (especially at the highest values of dis).

Question 2

Using the Boston data, with dis as the response and predictors medv, age and nox.

a. Split the data into training 60% and test 40%.

```
set.seed(123)
s<-sample(seq(1, nrow(Boston)),0.6*nrow(Boston))
Boston.train<-Boston[s,]
Boston.test<-Boston[-s,]
nrow(Boston.train)
```

```
## [1] 303
```

```
nrow(Boston.test)
```

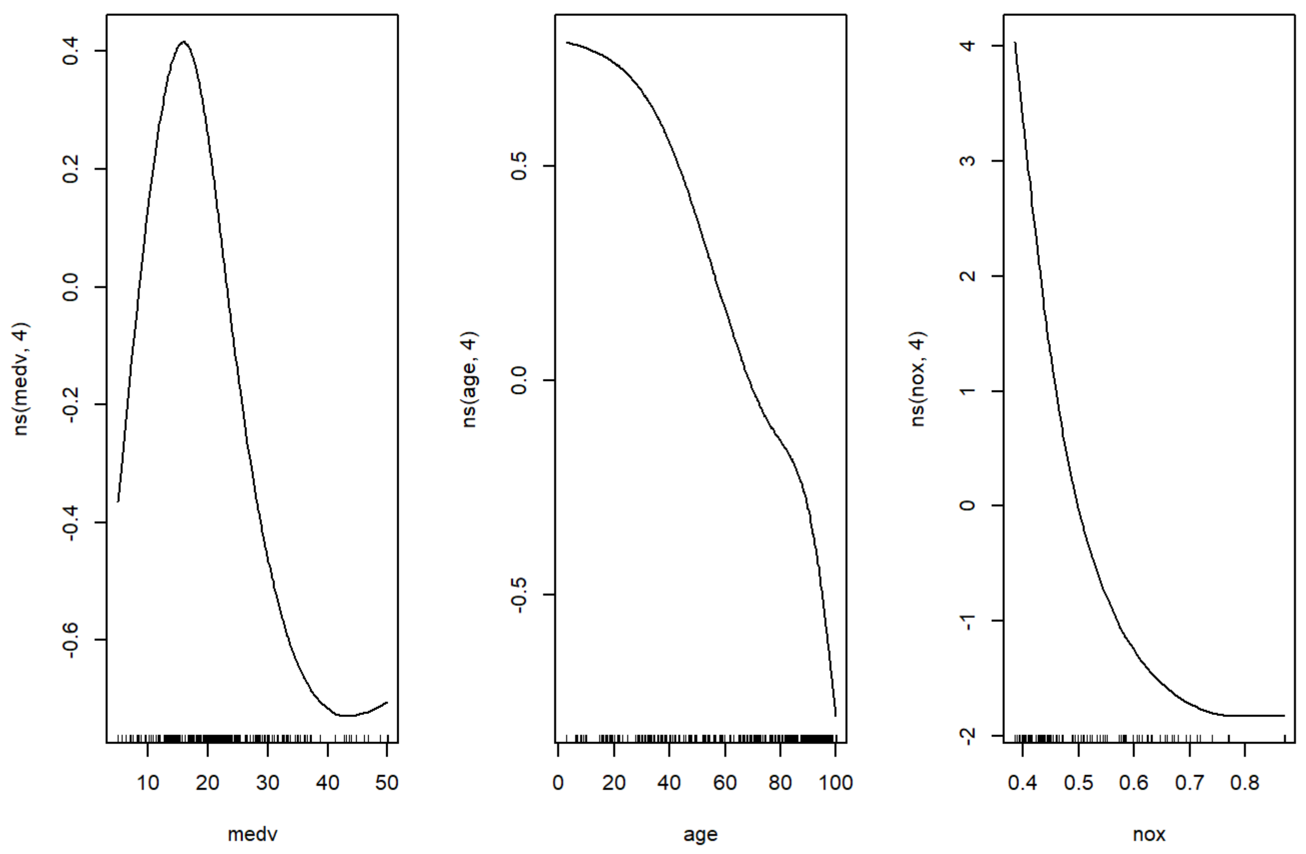
```
## [1] 203
```

Using the training data, fit a generalised additive model (GAM). Use ns with 4 degrees of freedom for each predictor.

```
f.q2a<-lm(dis ~ ns(medv,4) + ns(age,4) + ns(nox,4), data=Boston.train)
```

b. Use plot.gam to display the results.

```
par(mfrow=c(1,3))
plot.Gam(f.q2a)
```



Does it appear if a linear term is appropriate for any of the predictors?

No. None of the above plots indicate linearity for their respective model.

c. Simplify the model fit in part (a). Refit the model.

```
f.q2c<-lm(dis ~ ns(nox,4), data=Boston.train)
```

Use anova to compare the two fits

```
anova(f.q2c, f.q2a)
```

```
## Analysis of Variance Table
##
## Model 1: dis ~ ns(nox, 4)
## Model 2: dis ~ ns(medv, 4) + ns(age, 4) + ns(nox, 4)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    298 362.46
## 2    290 313.64  8    48.819 5.6424 1.149e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

and comment on your results.

The very low p-value in the Anova results above indicates that the unsimplified model (2) is a significantly (statistically speaking) better fit than the simplified model (1) with the medv and age predictors removed.

Question 4

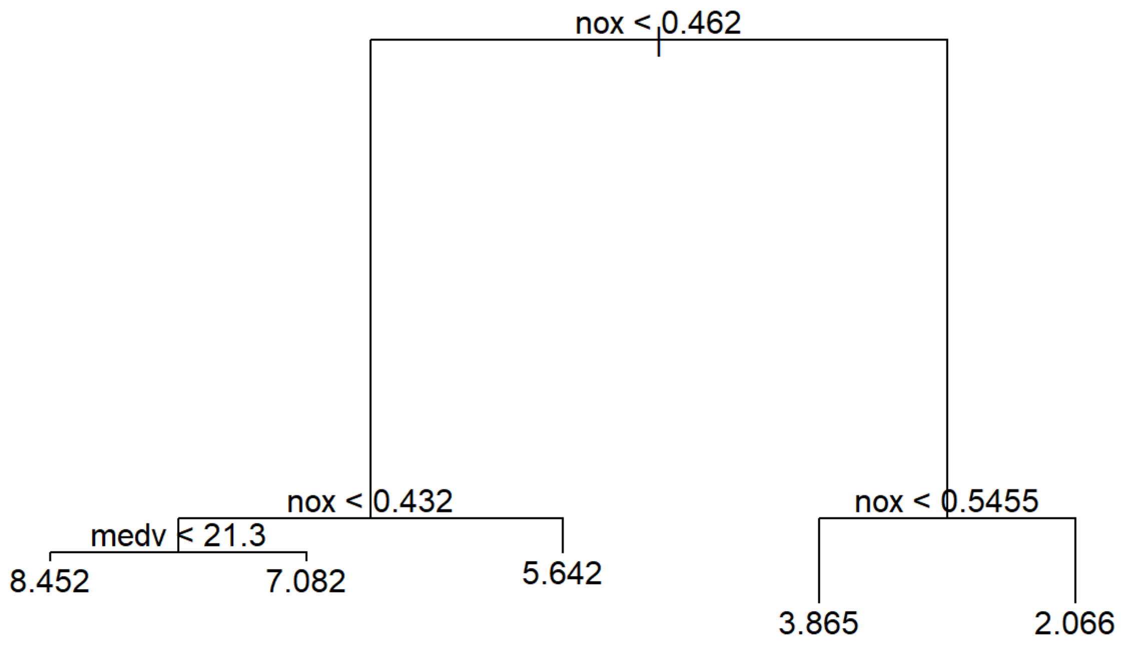
a. For the training data in question 2, fit a tree model. Use dis as response, and predictors medv, age and nox.

```
f.q4a<-tree(dis ~ medv + age + nox, data=Boston.train)
summary(f.q4a)
```

```
##
## Regression tree:
## tree(formula = dis ~ medv + age + nox, data = Boston.train)
## Variables actually used in tree construction:
## [1] "nox"  "medv"
## Number of terminal nodes:  5
## Residual mean deviance:  1.034 = 308.2 / 298
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.14700 -0.58100 -0.07791  0.00000  0.42530  5.04400
```

Draw the tree.

```
plot(f.q4a)
text(f.q4a)
```



Calculate the training and test MSE.

```
mse.train<-mean((predict(f.q4a) - Boston.train$dis)^2)
paste("Training MSE =", round(mse.train,5))
```

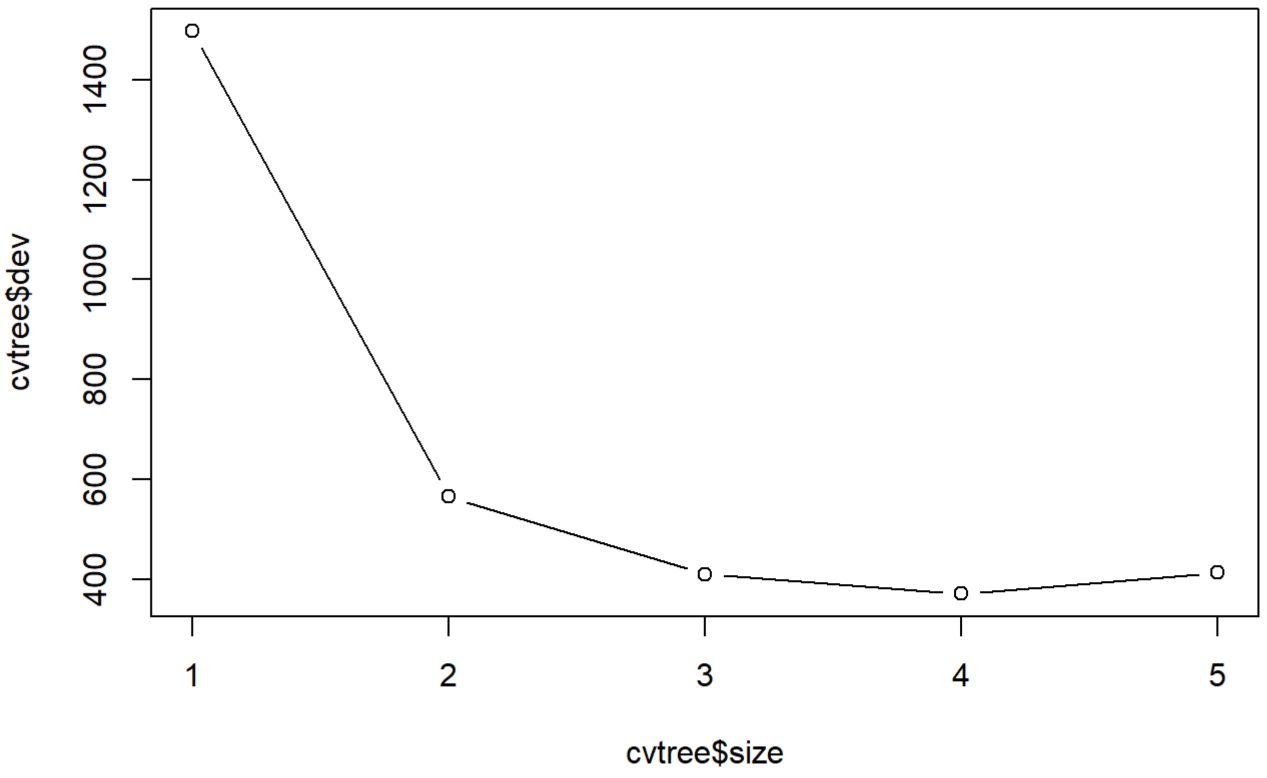
```
## [1] "Training MSE = 1.01727"
```

```
mse.test<-mean((predict(f.q4a, newdata=Boston.test) - Boston.test$dis)^2)
paste("    Test MSE =", round(mse.test, 5))
```

```
## [1] "    Test MSE = 0.84638"
```

b. Use cv.tree to select a pruned tree.

```
set.seed(123)
cvtree <- cv.tree(f.q4a)
plot(cvtree$size,cvtree$dev,type="b")
```



If pruning is required, fit and draw the pruned tree.

```
f.q4b<-prune.tree(f.q4a,best=4)
summary(f.q4b)
```

```
##
## Regression tree:
## snip.tree(tree = f.q4a, nodes = 4L)
## Variables actually used in tree construction:
## [1] "nox"
## Number of terminal nodes:  4
## Residual mean deviance:  1.085 = 324.5 / 299
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.25500 -0.58100 -0.06899  0.00000  0.43130  4.75400
```

Calculate the training and test MSE.

```
mse.train.pruned<-mean((predict(f.q4b) - Boston.train$dis)^2)
paste("Training MSE =", round(mse.train.pruned,5))
```

```
## [1] "Training MSE = 1.07093"
```

```
mse.test.pruned<-mean((predict(f.q4b, newdata=Boston.test) - Boston.test$dis)^2)
paste("    Test MSE =", round(mse.test.pruned, 5))
```

```
## [1] "    Test MSE = 0.84713"
```

Compare the results to those in (a).

The training MSE value of the unpruned tree is less than that of the pruned tree which indicates that it makes a better fit for the training dataset. However, the pruned tree achieved a lower test MSE value than the unpruned tree did. This an indicator that the unpruned model may be overfitted vis a vis the training dataset.

c. Which fit is better, the (optionally pruned) tree or the GAM?

```
mse.gam.train<-mean((predict(f.q2a)-Boston.train$dis)^2)
paste("Training MSE (GAM) =", round(mse.gam.train,5))
```

```
## [1] "Training MSE (GAM) = 1.03511"
```

The GAM delivers a better (lower) MSE (0.7906) on the training dataset than that achieved by the pruned tree model (0.82112).

Compare their performance on the test data.

```
mse.gam.test<-mean((predict(f.q2a, newdata=Boston.test)-Boston.test$dis)^2)
paste("    Test MSE (GAM) =", round(mse.gam.test,5))
```

```
## [1] "    Test MSE (GAM) = 0.89282"
```

The GAM delivers a worse (higher) MSE (1.29454) on the test dataset than that achieved by the pruned tree model (1.26445).

Question 5

For the data generated in question 6, Assignment 3:

```
set.seed(1)
x <- rnorm(100)
y <- 1 + .2*x+3*x^2+.6*x^3 + rnorm(100)
d <- data.frame(x=x,y=y)
```

a. Fit a regression model containing predictors X, X^2, . . .X^{10}.

```
formula<-cv.formula("x","y",10)
formula
```

```
## [1] "y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10)"
```

```
f.q5a<-lm(formula,data=d)
summary(f.q5a)
```

```
##
## Call:
## lm(formula = formula, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9774 -0.5895 -0.1238  0.4923  2.1505
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.17283    0.19971   5.873 7.28e-08 ***
## x            0.71409    0.59009   1.210    0.229
## I(x^2)       1.86854    1.29174   1.447    0.152
## I(x^3)      -0.33114    1.68567  -0.196    0.845
## I(x^4)       1.90383    2.14977   0.886    0.378
## I(x^5)       0.55110    1.35654   0.406    0.686
## I(x^6)      -1.26499    1.31956  -0.959    0.340
## I(x^7)      -0.15569    0.39731  -0.392    0.696
## I(x^8)       0.31987    0.32511   0.984    0.328
## I(x^9)       0.01628    0.03817   0.426    0.671
## I(x^10)     -0.02690    0.02749  -0.979    0.330
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9719 on 89 degrees of freedom
## Multiple R-squared:  0.951,  Adjusted R-squared:  0.9455
## F-statistic: 172.7 on 10 and 89 DF,  p-value: < 2.2e-16
```

Based on the output in summary() which terms are needed in the model?

The above output indicates that none of the terms are significant.

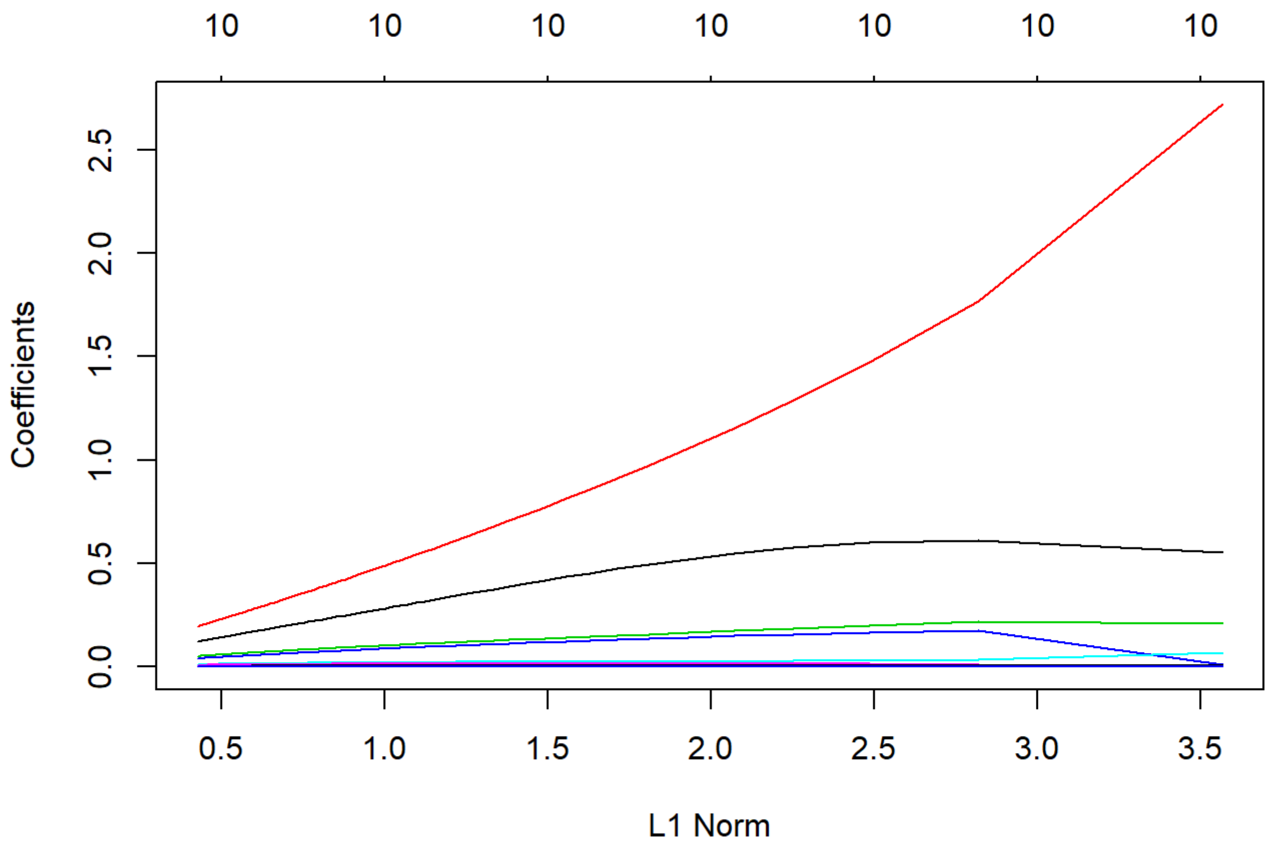
b. Fit a ridge regression model using the glmnet function over a grid of values for λ ranging from 0.001 to 50.

```
grid<-seq(0.001,50,length=100)
x<-model.matrix(as.formula(formula),data=d)
y<-d$y
f.q5b<-glmnet(x,y,alpha=0,lambda=grid)
summary(f.q5b)
```

```
##           Length Class     Mode
## a0         100   -none-    numeric
## beta      1100   dgCMatrix S4
## df         100   -none-    numeric
## dim          2   -none-    numeric
## lambda     100   -none-    numeric
## dev.ratio  100   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         5   -none-    call
## nobs         1   -none-    numeric
```

Plot coefficients vs penalty using the default plot method.

```
plot(f.q5b)
```



Use the inbuilt function cv.glmnet to choose the tuning parameter λ.

```
set.seed(123)
cv<-cv.glmnet(x,y,alpha=0)
lambda.q5b<-cv$lambda.min
lambda.q5b
```

```
## [1] 0.4000507
```

How do the coefficients at the optimal value of λ compare to the linear regression ones in (a)?

```
coefficients<-data.frame(lm=f.q5a$coefficients,ridge=coef(cv)[-2])
coefficients
```

```
##                      lm         ridge
## (Intercept)  1.17282867  1.6062726386
## x            0.71409233  0.6101938744
## I(x^2)       1.86853993  1.6791856551
## I(x^3)      -0.33113515  0.2148939001
## I(x^4)       1.90382807  0.1724604457
## I(x^5)       0.55109577  0.0314359406
## I(x^6)      -1.26499408  0.0127949493
## I(x^7)      -0.15569320  0.0035129278
## I(x^8)       0.31986888  0.0003363051
## I(x^9)       0.01627747  0.0002841421
## I(x^10)     -0.02690171 -0.0001501457
```

As shown in the table above, the ridge regression cofficients (for the most part) follow a decreasing pattern culminating in their only negative value, while the values for the linear regression coefficients are more randomly distributed in terms of both value and sign.

c. Repeat (b) for lasso regression instead of ridge.

Fit a Lasso regression model using the glmnet function over a grid of values for λ ranging from 0.001 to 50.

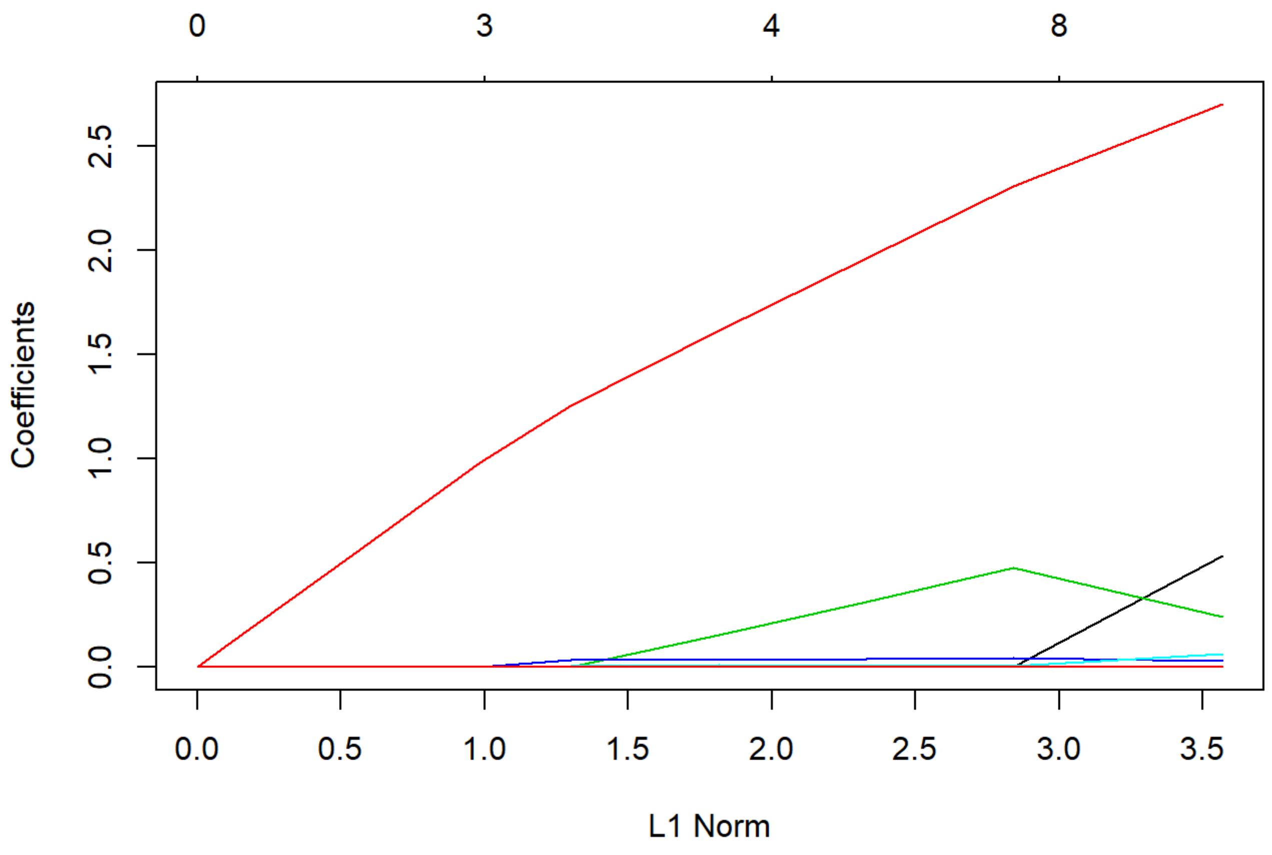
```
grid<-seq(0.001,50,length=100)
x<-model.matrix(as.formula(formula),data=d)
y<-d$y
f.q5c<-glmnet(x,y,alpha=1,lambda=grid)
summary(f.q5c)
```

```
##           Length Class     Mode   
## a0         100   -none-    numeric
## beta      1100   dgCMatrix S4     
## df         100   -none-    numeric
## dim          2   -none-    numeric
## lambda     100   -none-    numeric
## dev.ratio  100   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         5   -none-    call   
## nobs         1   -none-    numeric
```

Plot coefficients vs penalty using the default plot method.

```
plot(f.q5c)
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
## unique 'x' values
```



Use the inbuilt function cv.glmnet to choose the tuning parameter λ.

```
set.seed(123)
cv<-cv.glmnet(x,y,alpha=1)
lambda.q5c<-cv$lambda.min
lambda.q5c
```

```
## [1] 0.07323881
```

How do the coefficients at the optimal value of λ compare to the linear regression ones in (a)?

```
coefficients<-data.frame(lm=f.q5a$coefficients,lasso=coef(cv)[-2])
coefficients
```

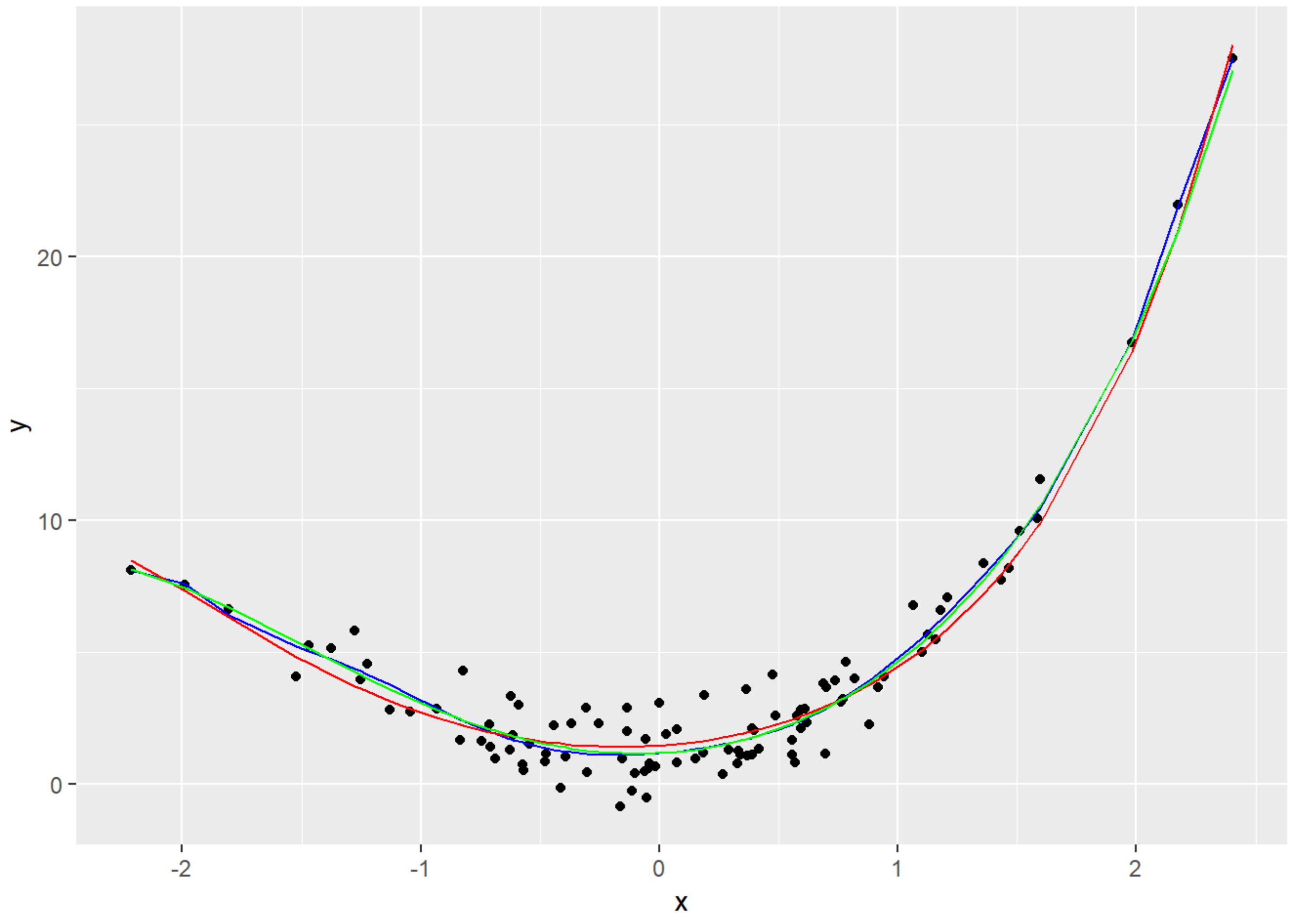
```
##                       lm        lasso
## (Intercept)  1.17282867 1.2382695641
## x            0.71409233 0.2539977431
## I(x^2)       1.86853993 2.5627274115
## I(x^3)      -0.33113515 0.4081060668
## I(x^4)       1.90382807 0.0445976551
## I(x^5)       0.55109577 0.0276827581
## I(x^6)      -1.26499408 0.0000000000
## I(x^7)      -0.15569320 0.0007376607
## I(x^8)       0.31986888 0.0000000000
## I(x^9)       0.01627747 0.0000000000
## I(x^10)     -0.02690171 0.0000000000
```

As shown in the table above, apart from 4 zero values, the lasso regression coefficient values are all positive, unlike their linear regression equivalents.

d. Plot the data y vs x and superimpose the fitted models from linear regression, ridge and lasso with optimal values of lambda as chosen by cross-validation.

```
ridge.pred<-predict(f.q5b, s=lambda.q5b,
                      newx=model.matrix(as.formula(formula), data=d))
lasso.pred<-predict(f.q5c, s=lambda.q5c,
                      newx=model.matrix(as.formula(formula), data=d))

ggplot(aes(x=x,y=y), data=d) +
  geom_point() +
  geom_line(aes(y=fitted(f.q5a)),colour="blue") +
  geom_line(aes(y=ridge.pred),colour="red") +
  geom_line(aes(y=lasso.pred),colour="green")
```



It is remarkable, for this dataset at least, how similar the fits produced by linear, ridge and lasso regression have achieved are once the optimal value of lambda was used for the latter two methods.