Craig Van Vliet & Ryan Thomas

ECO 7100 – Econometrics 1

March 25th, 2019

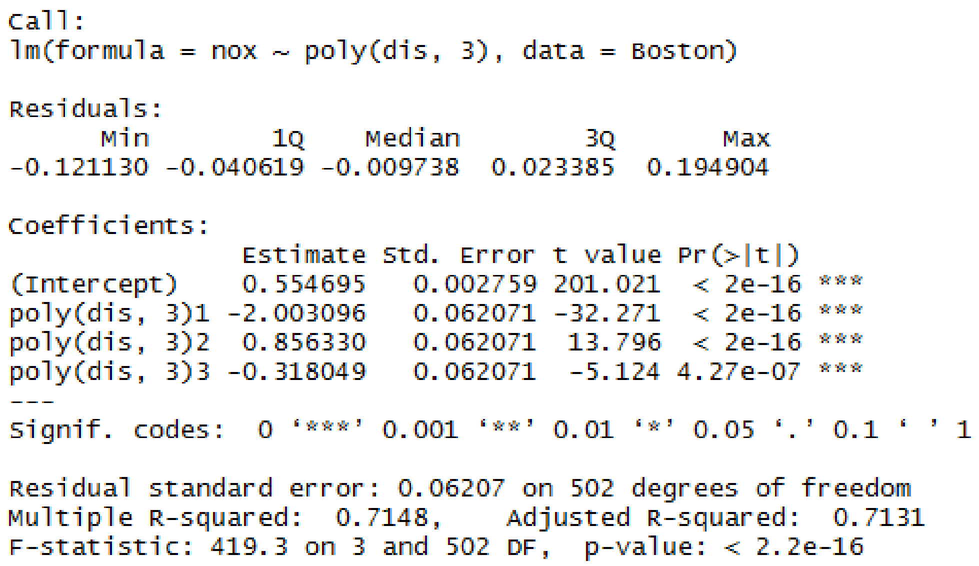
# Lab: Regression Spline and Cross-validation

Assignment Submission

1) This question uses the variables dis (the weighted mean of distances to five Boston employment centers) and nox (nitrogen oxides concentration in parts per 10 million) from the Boston data. We will treat dis as the predictor and nox as the response.

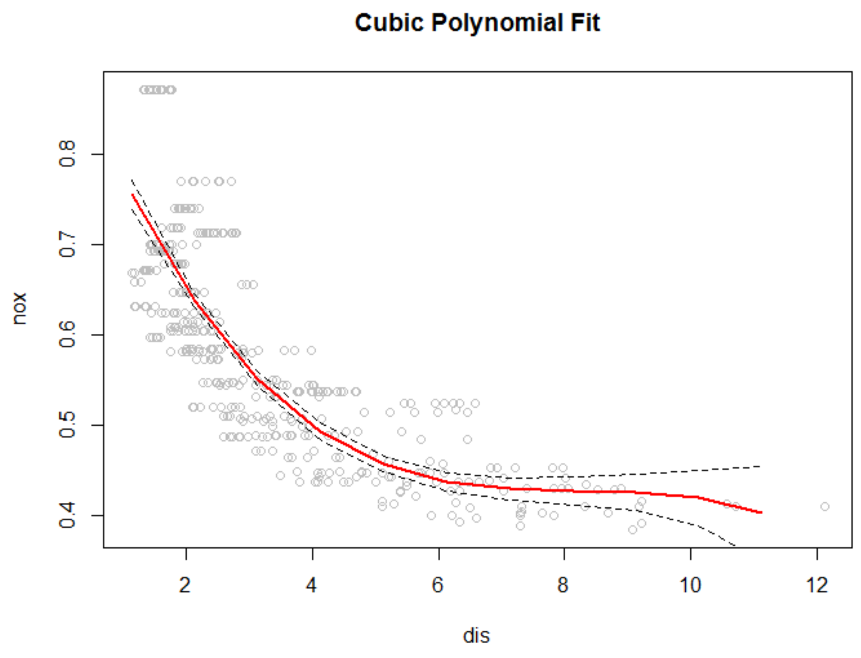
a) Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

```
set.seed(1)
library(MASS)
attach(Boston)

m3 <- lm(nox~poly(dis,3),data=Boston)
summary(m3)
```



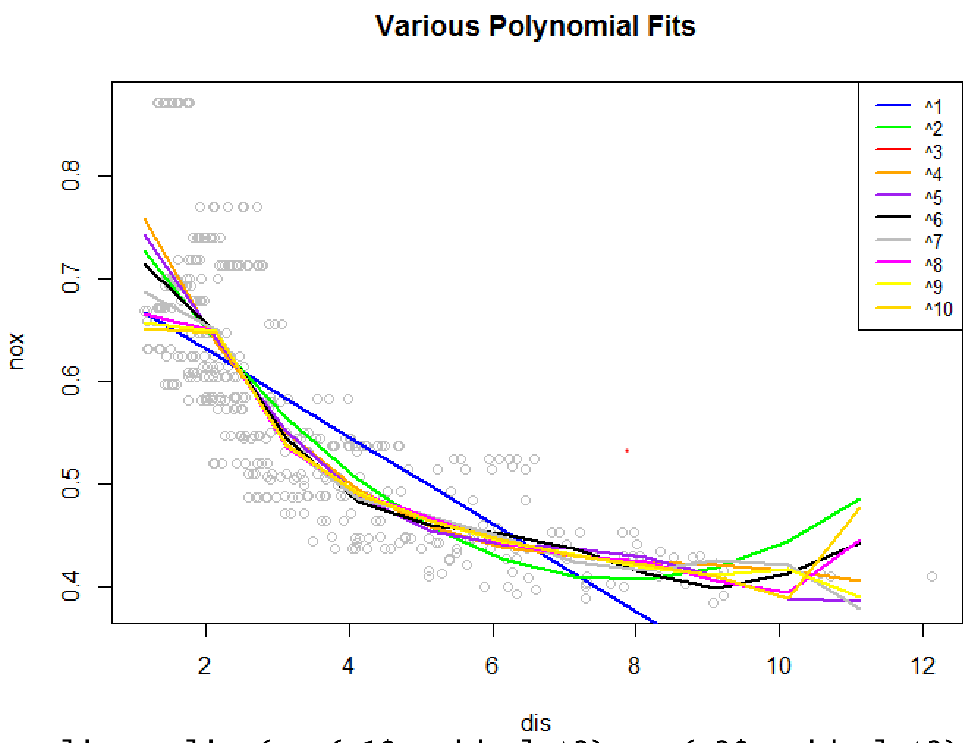
```
Call:
lm(formula = nox ~ poly(dis, 3), data = Boston)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.121130 -0.040619 -0.009738  0.023385  0.194904 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.06207 on 502 degrees of freedom
Multiple R-squared:  0.7148,	Adjusted R-squared:  0.7131 
F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
dislims <- range(dis)
dis.grid <- seq(from=dislims[1],to=dislims[2])

pred <- predict(m3,newdata=list(dis=dis.grid),se=T)

plot(dis,nox,col="gray")
title("Cubic Polynomial Fit")
lines(dis.grid,pred$fit,lwd=2,col="red")
lines(dis.grid,pred$fit+2*pred$se,lty="dashed")
lines(dis.grid,pred$fit-2*pred$se,lty="dashed")
```



(b) Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```
plot(dis,nox,col="gray")
title("Various Polynomial Fits")
m1 <- lm(nox~dis,data=Boston)
pred <- predict(m1,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="blue")
m2 <- lm(nox~poly(dis,2),data=Boston)
pred <- predict(m2,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="green")
m4 <- lm(nox~poly(dis,4),data=Boston)
pred <- predict(m4,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="orange")
m5 <- lm(nox~poly(dis,5),data=Boston)
pred <- predict(m5,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="purple")
m6 <- lm(nox~poly(dis,6),data=Boston)
pred <- predict(m6,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="black")
m7 <- lm(nox~poly(dis,7),data=Boston)
pred <- predict(m7,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="gray")
m8 <- lm(nox~poly(dis,8),data=Boston)
pred <- predict(m8,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="magenta")
m9 <- lm(nox~poly(dis,9),data=Boston)
pred <- predict(m9,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="yellow")
m10 <- lm(nox~poly(dis,10),data=Boston)
pred <- predict(m10,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="gold")
legend("topright",legend=c("^1","^2","^3","^4","^5","^6","^7","^8","^9","^10"),
col=c("blue","green","red","orange","purple","black","gray","magenta","yellow","g
old"),lty=1,lwd=2,cex=.8)
```

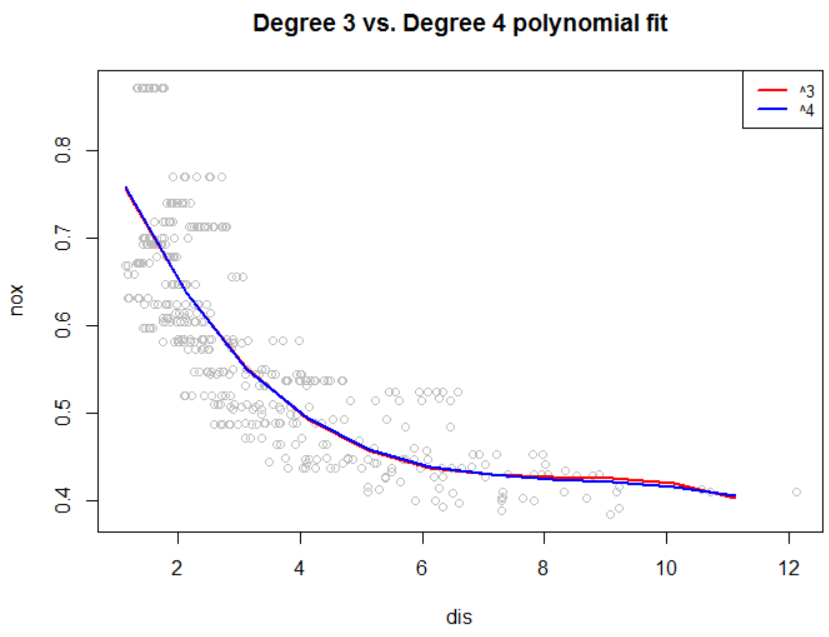


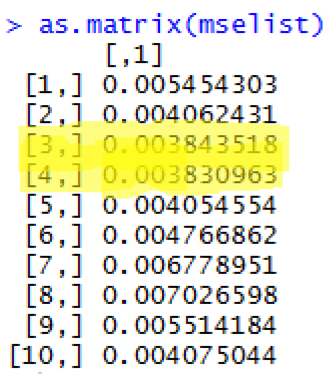
```
rsslist <-list(sum(m1$residuals^2),sum(m2$residuals^2),sum(m3$residuals^2),
sum(m4$residuals^2),sum(m5$residuals^2),sum(m6$residuals^2),sum(m7$residuals^2),
sum(m8$residuals^2),sum(m9$residuals^2),sum(m10$residuals^2))
as.numeric(rsslist)
> as.numeric(rsslist)
 [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484 1.835630 1.833331 1.832171
```

(c) Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

```
library(caret)
library(elasticnet)
set.seed(1)
train.control <- trainControl(method = "repeatedcv", number = 10,repeats=3)
mcv1 <- train(nox~dis,data=Boston,method ="lm",trControl=train.control)
mse1 <- (mcv1$results$RMSE^2)
mcv2 <- train(nox~poly(dis,2),data=Boston,method ="lm",trControl=train.control)
mse2 <- (mcv2$results$RMSE^2)
mcv3 <- train(nox~poly(dis,3),data=Boston,method ="lm",trControl=train.control)
mse3 <- (mcv3$results$RMSE^2)
mcv4 <- train(nox~poly(dis,4),data=Boston,method ="lm",trControl=train.control)
mse4 <- (mcv4$results$RMSE^2)
mcv5 <- train(nox~poly(dis,5),data=Boston,method ="lm",trControl=train.control)
mse5 <- (mcv5$results$RMSE^2)
mcv6 <- train(nox~poly(dis,6),data=Boston,method ="lm",trControl=train.control)
mse6 <- (mcv6$results$RMSE^2)
mcv7 <- train(nox~poly(dis,7),data=Boston,method ="lm",trControl=train.control)
mse7 <- (mcv7$results$RMSE^2)
mcv8 <- train(nox~poly(dis,8),data=Boston,method ="lm",trControl=train.control)
mse8 <- (mcv8$results$RMSE^2)
mcv9 <- train(nox~poly(dis,9),data=Boston,method ="lm",trControl=train.control)
mse9 <- (mcv9$results$RMSE^2)
mcv10 <- train(nox~poly(dis,10),data=Boston,method ="lm",trControl=train.control)
mse10 <- (mcv10$results$RMSE^2)
mselist <- list(mse1,mse2,mse3,mse4,mse5,mse6,mse7,mse8,mse9,mse10)
```

```
as.matrix(mselist)

plot(dis,nox,col="gray")
title("Degree 3 vs. Degree 4 polynomial fit")
m3 <- lm(nox~poly(dis,3),data=Boston)
pred <- predict(m3,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="red")
m4 <- lm(nox~poly(dis,4),data=Boston)
pred <- predict(m4,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="blue")
legend("topright",legend=c("^3","^4"),col=c("red","blue"),lty=1,lwd=2,cex=.8)
```



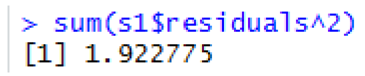


```
> as.matrix(mselist)
         [,1]
 [1,] 0.005454303
 [2,] 0.004062431
 [3,] 0.003843518
 [4,] 0.003830963
 [5,] 0.004054554
 [6,] 0.004766862
 [7,] 0.006778951
 [8,] 0.007026598
 [9,] 0.005514184
[10,] 0.004075044
```

Third-degree and fourth-degree polynomials have the lowest average MSE values over 3 repeated 10-fold cross-validations. Since the third-degree polynomial is simpler and cubic polynomials are standard practice, the third degree is optimal.

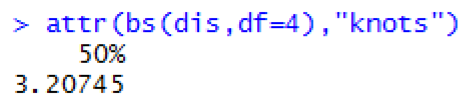
(d) Use the bs() function to fit a regression spline to predict nox using dis. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

```
library(splines)
s1 <- lm(nox~bs(dis,df=4),data=Boston)
sum(s1$residuals^2)
```

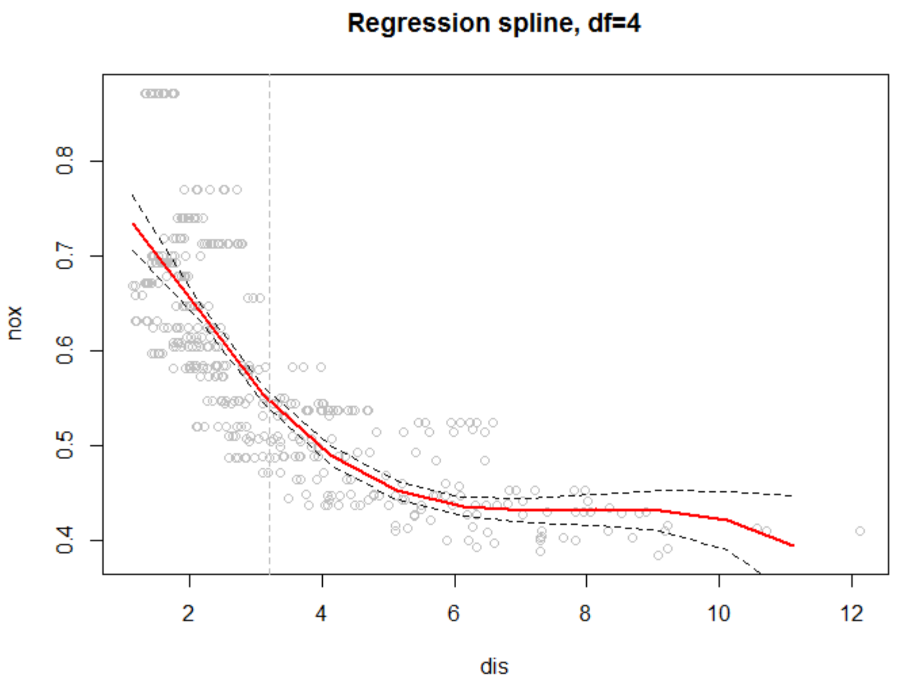


```
> sum(s1$residuals^2)
[1] 1.922775
```

```
pred <- predict(s1,newdata=list(dis=dis.grid),se=T)
plot(dis,nox,col="gray")
lines(dis.grid,pred$fit,lwd=2,col="red")
abline(v=c(3.207),lty=2,col="gray")
title("Regression spline, df=4")
lines(dis.grid,pred$fit+2*pred$se,lty="dashed")
lines(dis.grid,pred$fit-2*pred$se,lty="dashed")
attr(bs(dis,df=4),"knots")
```



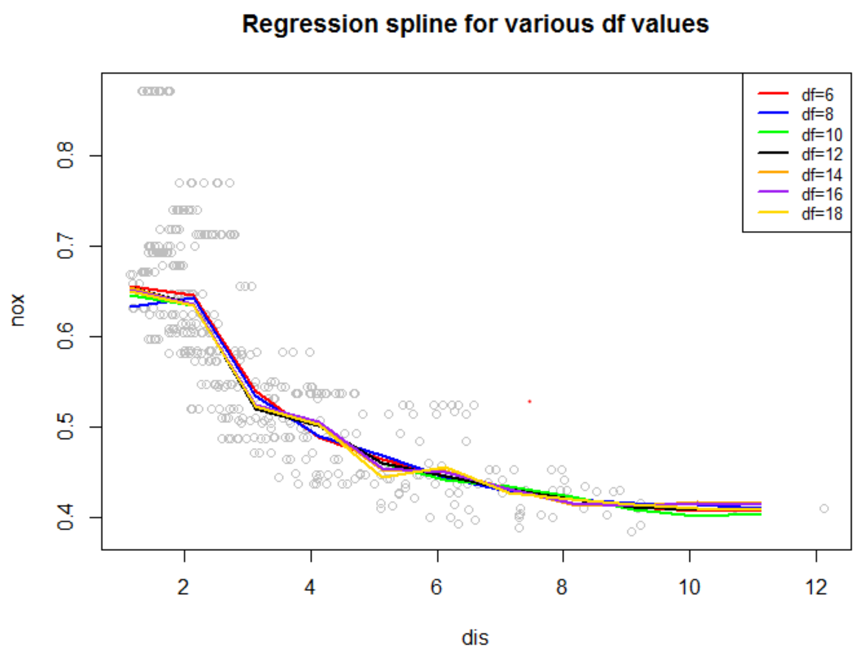
```
> attr(bs(dis,df=4),"knots")
     50%
 3.20745
```



Using the bs() function does not allow us to specify both the degrees of freedom and the knots. The function sets a single knot at the median dis value (3.207) when four degrees of freedom are specified. According to the bs() function documentation, the number of knots is determined by df minus degree, which is 4 – 3 = 1 in our case. The appropriate quantile is then chosen based on the knot number.

(e) Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

```
plot(dis,nox,col="gray")
title("Regression spline for various df values")
s2 <- lm(nox~bs(dis,df=6),data=Boston)
pred <- predict(s2,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="red")
s3 <- lm(nox~bs(dis,df=8),data=Boston)
pred <- predict(s3,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="blue")
s4 <- lm(nox~bs(dis,df=10),data=Boston)
pred <- predict(s4,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="green")
s5 <- lm(nox~bs(dis,df=12),data=Boston)
pred <- predict(s5,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="black")
s6 <- lm(nox~bs(dis,df=14),data=Boston)
pred <- predict(s6,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="orange")
s7 <- lm(nox~bs(dis,df=16),data=Boston)
pred <- predict(s7,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="purple")
s8 <- lm(nox~bs(dis,df=18),data=Boston)
pred <- predict(s8,newdata=list(dis=dis.grid),se=T)
lines(dis.grid,pred$fit,lwd=2,col="gold")
legend("topright",legend=c("df=6","df=8","df=10","df=12","df=14","df=16","df=18")
,col=c("red","blue","green","black","orange","purple","gold"),lty=1,lwd=2,cex=.8)

rsslist2 <- list(sum(s2$residuals^2),sum(s3$residuals^2),sum(s4$residuals^2),
sum(s5$residuals^2),sum(s6$residuals^2),sum(s7$residuals^2),sum(s8$residuals^2))
as.matrix(rsslist2)
```



```
> as.matrix(rsslist2)
         [,1]
[1,] 1.833966
[2,] 1.816995
[3,] 1.792535
[4,] 1.788999
[5,] 1.781838
[6,] 1.783546
[7,] 1.775838
```

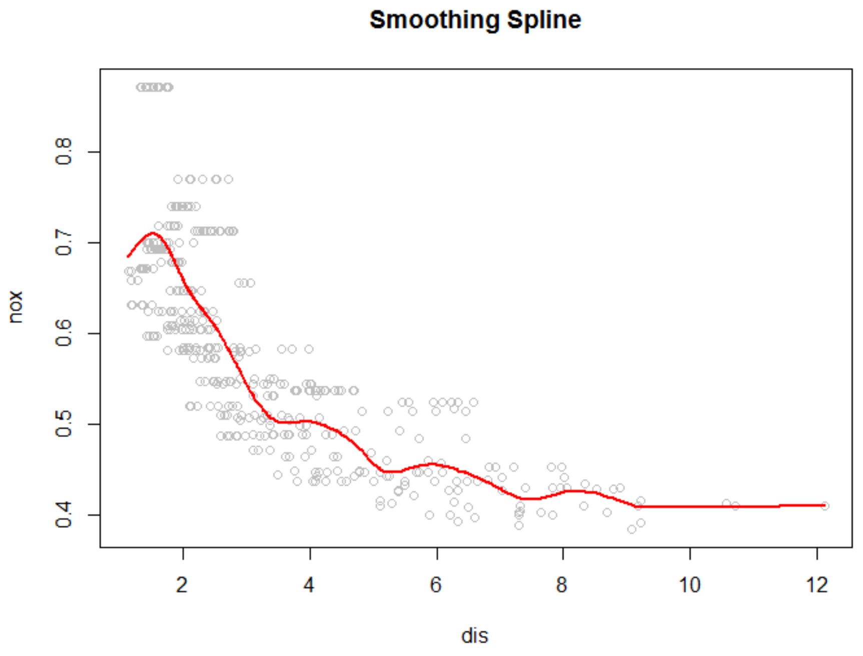
Increasing degrees of freedom decreases RSS, but at a decreasing rate.  RSS is lowest at df = 14.

(f) Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe your results.

```
plot(dis,nox,col="gray")
title("Smoothing Spline")
ss <- smooth.spline(dis,nox,cv=TRUE)
ss$df
```

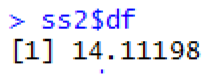
```
> ss$df
[1] 15.42984
```

```
pred <- predict(ss,newdata=list(dis=dis.grid),se=T)
lines(ss,lwd=2,col="red")
```



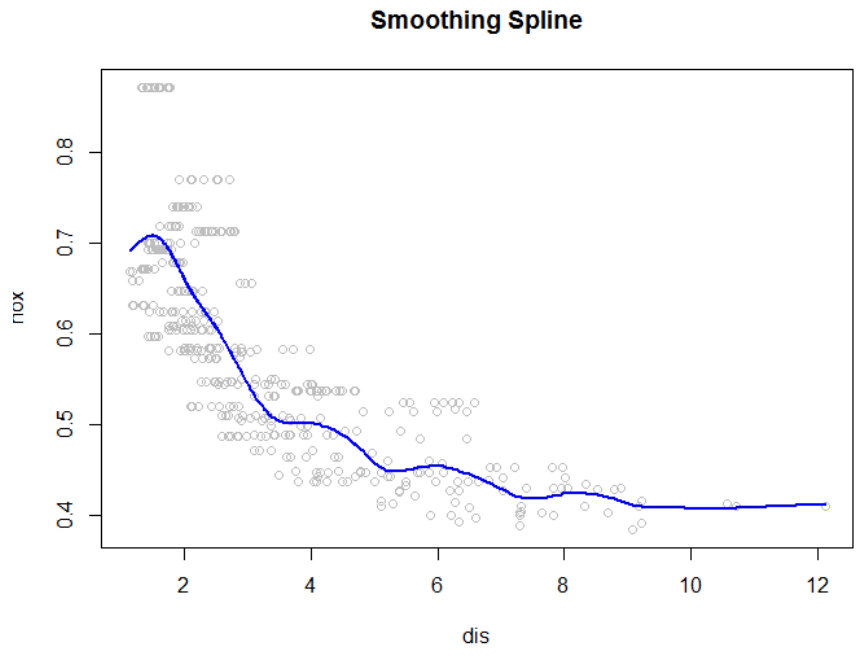
The smooth.spline function uses leave-one-out cross-validation to determine optimal degrees of freedom and smoothing parameter lambda.  The function selects 15.42984 degrees of freedom.

```
ss2 <- smooth.spline(dis,nox,cv=FALSE)
ss2$df
```



```
> ss2$df
[1] 14.11198
```

```
pred <- predict(ss2,newdata=list(dis=dis.grid),se=T)
lines(ss2,lwd=2,col="blue")
```



Changing the cv argument to “false” tells the smooth.spline function to use a more “generalized” cross-validation method according to the function documentation, which may be more appropriate if there are duplicate  values. This results in an optimal df of 14.11198, which is more consistent with the optimal df we found when comparing models in part (e).