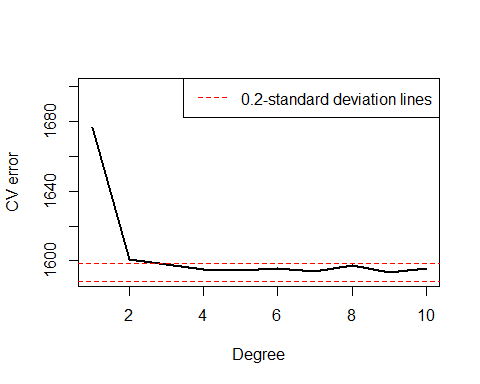
R Notebook

***6. In this exercise, you will further analyze the Wage data set considered throughout this chapter.***

1. Perform polynomial regression to predict wage using age. Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.

Load Wage dataset. Keep an array of all cross-validation errors. We are performing K-fold cross validation with K=10.

set.seed(1)  
library(ISLR)  
library(boot)  
all.deltas = rep(NA, 10)  
for (i in 1:10) {  
 glm.fit = glm(wage~poly(age, i), data=Wage)  
 all.deltas[i] = cv.glm(Wage, glm.fit, K=10)$delta[2]  
}  
plot(1:10, all.deltas, xlab="Degree", ylab="CV error", type="l", pch=20, lwd=2, ylim=c(1590, 1700))  
min.point = min(all.deltas)  
sd.points = sd(all.deltas)  
abline(h=min.point + 0.2 \* sd.points, col="red", lty="dashed")  
abline(h=min.point - 0.2 \* sd.points, col="red", lty="dashed")  
legend("topright", "0.2-standard deviation lines", lty="dashed", col="red")

 he cv-plot with standard deviation lines show that d=3 is the smallest degree giving reasonably small cross-validation error.

We now find best degree using Anova.

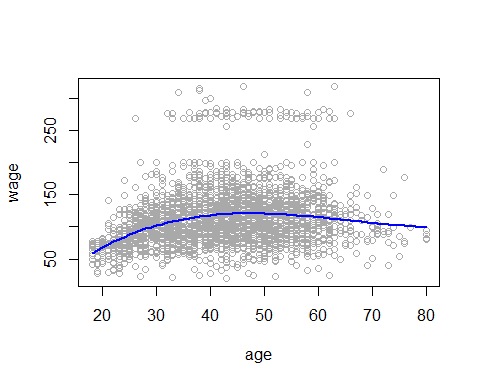
fit.1 = lm(wage~poly(age, 1), data=Wage)  
fit.2 = lm(wage~poly(age, 2), data=Wage)  
fit.3 = lm(wage~poly(age, 3), data=Wage)  
fit.4 = lm(wage~poly(age, 4), data=Wage)  
fit.5 = lm(wage~poly(age, 5), data=Wage)  
fit.6 = lm(wage~poly(age, 6), data=Wage)  
fit.7 = lm(wage~poly(age, 7), data=Wage)  
fit.8 = lm(wage~poly(age, 8), data=Wage)  
fit.9 = lm(wage~poly(age, 9), data=Wage)  
fit.10 = lm(wage~poly(age, 10), data=Wage)  
anova(fit.1, fit.2, fit.3, fit.4, fit.5, fit.6, fit.7, fit.8, fit.9, fit.10)

## Analysis of Variance Table  
##   
## Model 1: wage ~ poly(age, 1)  
## Model 2: wage ~ poly(age, 2)  
## Model 3: wage ~ poly(age, 3)  
## Model 4: wage ~ poly(age, 4)  
## Model 5: wage ~ poly(age, 5)  
## Model 6: wage ~ poly(age, 6)  
## Model 7: wage ~ poly(age, 7)  
## Model 8: wage ~ poly(age, 8)  
## Model 9: wage ~ poly(age, 9)  
## Model 10: wage ~ poly(age, 10)  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 2998 5022216   
## 2 2997 4793430 1 228786 143.7638 < 2.2e-16 \*\*\*  
## 3 2996 4777674 1 15756 9.9005 0.001669 \*\*   
## 4 2995 4771604 1 6070 3.8143 0.050909 .   
## 5 2994 4770322 1 1283 0.8059 0.369398   
## 6 2993 4766389 1 3932 2.4709 0.116074   
## 7 2992 4763834 1 2555 1.6057 0.205199   
## 8 2991 4763707 1 127 0.0796 0.777865   
## 9 2990 4756703 1 7004 4.4014 0.035994 \*   
## 10 2989 4756701 1 3 0.0017 0.967529   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Anova shows that all polynomials above degree 3 are insignificant at 1 significance level.

We now plot the polynomial prediction on the data

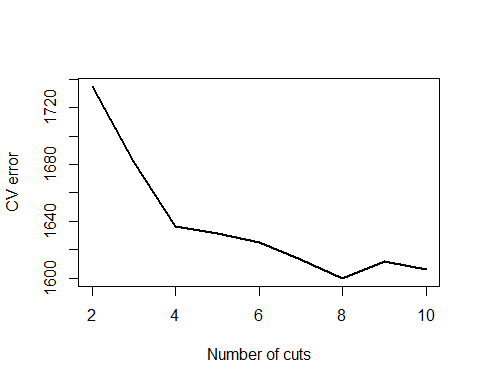
plot(wage~age, data=Wage, col="darkgrey")  
agelims = range(Wage$age)  
age.grid = seq(from=agelims[1], to=agelims[2])  
lm.fit = lm(wage~poly(age, 3), data=Wage)  
lm.pred = predict(lm.fit, data.frame(age=age.grid))  
lines(age.grid, lm.pred, col="blue", lwd=2)



1. Fit a step function to predict wage using age, and perform crossvalidation to choose the optimal number of cuts. Make a plot of the fit obtained.

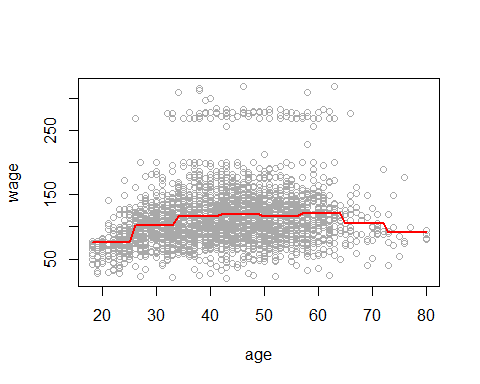
We use cut points of up to 10.

all.cvs = rep(NA, 10)  
for (i in 2:10) {  
 Wage$age.cut = cut(Wage$age, i)  
 lm.fit = glm(wage~age.cut, data=Wage)  
 all.cvs[i] = cv.glm(Wage, lm.fit, K=10)$delta[2]  
}  
plot(2:10, all.cvs[-1], xlab="Number of cuts", ylab="CV error", type="l", pch=20, lwd=2)

 The cross validation shows that test error is minimum for k=8 cuts.

We now train the entire data with step function using 8 cuts and plot it.

lm.fit = glm(wage~cut(age, 8), data=Wage)  
agelims = range(Wage$age)  
age.grid = seq(from=agelims[1], to=agelims[2])  
lm.pred = predict(lm.fit, data.frame(age=age.grid))  
plot(wage~age, data=Wage, col="darkgrey")  
lines(age.grid, lm.pred, col="red", lwd=2)



***9. 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.***

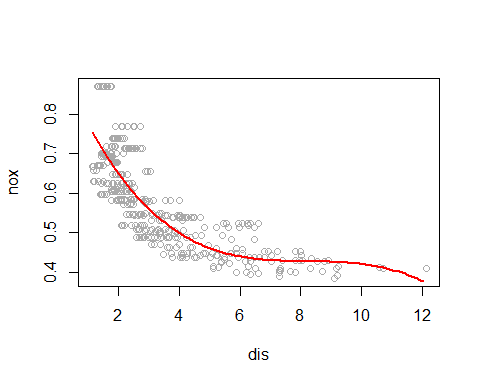
set.seed(1)  
library(MASS)  
attach(Boston)

1. 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.

lm.fit = lm(nox ~ poly(dis, 3), data = Boston)  
summary(lm.fit)

##   
## 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

dislim = range(dis)  
dis.grid = seq(from = dislim[1], to = dislim[2], by = 0.1)  
lm.pred = predict(lm.fit, list(dis = dis.grid))  
plot(nox ~ dis, data = Boston, col = "darkgrey")  
lines(dis.grid, lm.pred, col = "red", lwd = 2)



Summary demonstrates that when forecasting nox with dis, all polynomial terms are meaningful. The plot displays a smooth curve that reasonably fits the data.

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

We plot polynomials of degrees 1 to 10 and save train RSS.

all.rss = rep(NA, 10)  
for (i in 1:10) {  
 lm.fit = lm(nox ~ poly(dis, i), data = Boston)  
 all.rss[i] = sum(lm.fit$residuals^2)  
}  
all.rss

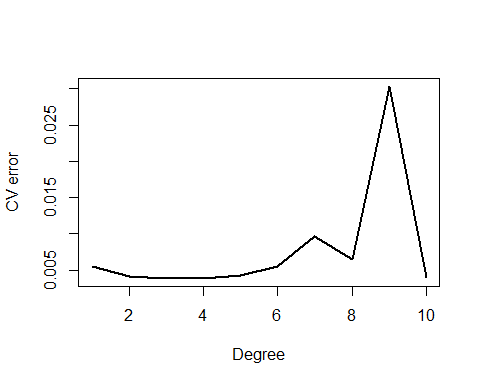
## [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484 1.835630  
## [9] 1.833331 1.832171

As expected, train RSS monotonically decreases with degree of polynomial.

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

We use a 10-fold cross validation to pick the best polynomial degree

library(boot)  
all.deltas = rep(NA, 10)  
for (i in 1:10) {  
 glm.fit = glm(nox ~ poly(dis, i), data = Boston)  
 all.deltas[i] = cv.glm(Boston, glm.fit, K = 10)$delta[2]  
}  
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20,   
 lwd = 2)

 A 10-fold CV reveals that the CV error decreases from degree 1 to degree 3, remains about constant until degree 5, and then increases for higher degrees. The best polynomial degree, in our opinion, is 4.

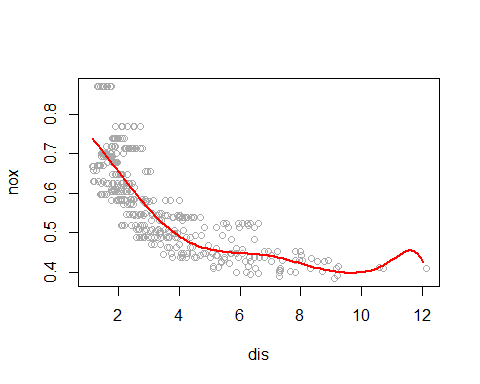
1. Use the bs() function to fit a regression spline to predict nox using dis. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

We can see that dis has roughly 1 and 13 limitations, respectively. We divide this range into four relatively equal portions, and we create knots at [4,7,11]. Recall that the R bs function requires either a df or knots argument. Knots are disregarded if both are specified.

library(splines)  
sp.fit = lm(nox ~ bs(dis, df = 4, knots = c(4, 7, 11)), data = Boston)  
summary(sp.fit)

##   
## Call:  
## lm(formula = nox ~ bs(dis, df = 4, knots = c(4, 7, 11)), data = Boston)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.124567 -0.040355 -0.008702 0.024740 0.192920   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.73926 0.01331 55.537 < 2e-16 \*\*\*  
## bs(dis, df = 4, knots = c(4, 7, 11))1 -0.08861 0.02504 -3.539 0.00044 \*\*\*  
## bs(dis, df = 4, knots = c(4, 7, 11))2 -0.31341 0.01680 -18.658 < 2e-16 \*\*\*  
## bs(dis, df = 4, knots = c(4, 7, 11))3 -0.26618 0.03147 -8.459 3.00e-16 \*\*\*  
## bs(dis, df = 4, knots = c(4, 7, 11))4 -0.39802 0.04647 -8.565 < 2e-16 \*\*\*  
## bs(dis, df = 4, knots = c(4, 7, 11))5 -0.25681 0.09001 -2.853 0.00451 \*\*   
## bs(dis, df = 4, knots = c(4, 7, 11))6 -0.32926 0.06327 -5.204 2.85e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.06185 on 499 degrees of freedom  
## Multiple R-squared: 0.7185, Adjusted R-squared: 0.7151   
## F-statistic: 212.3 on 6 and 499 DF, p-value: < 2.2e-16

sp.pred = predict(sp.fit, list(dis = dis.grid))  
plot(nox ~ dis, data = Boston, col = "darkgrey")  
lines(dis.grid, sp.pred, col = "red", lwd = 2)



1. 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.

We fit regression splines with dfs between 3 and 16.

all.cv = rep(NA, 16)  
for (i in 3:16) {  
 lm.fit = lm(nox ~ bs(dis, df = i), data = Boston)  
 all.cv[i] = sum(lm.fit$residuals^2)  
}  
all.cv[-c(1, 2)]

## [1] 1.934107 1.922775 1.840173 1.833966 1.829884 1.816995 1.825653 1.792535  
## [9] 1.796992 1.788999 1.782350 1.781838 1.782798 1.783546

Train RSS monotonically decreases till df=14 and then slightly increases for df=15 and df=16.

1. 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.

Finally, we use a 10-fold cross validation to find best df. We try all integer values of df between 3 and 16.

all.cv = rep(NA, 16)  
for (i in 3:16) {  
 lm.fit = glm(nox ~ bs(dis, df = i), data = Boston)  
 all.cv[i] = cv.glm(Boston, lm.fit, K = 10)$delta[2]  
}

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.1296, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.1296, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.137, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.137, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.0992), Boundary.knots =  
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.0992), Boundary.knots =  
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.2157), Boundary.knots =  
## c(1.1296, : some 'x' values beyond boundary knots may cause ill-conditioned  
## bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.2157), Boundary.knots =  
## c(1.1296, : some 'x' values beyond boundary knots may cause ill-conditioned  
## bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.354, `66.66667%` =  
## 4.2474: some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.354, `66.66667%` =  
## 4.2474: some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.4212, `66.66667%`  
## = 4.38856666666667: some 'x' values beyond boundary knots may cause ill-  
## conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.4212, `66.66667%`  
## = 4.38856666666667: some 'x' values beyond boundary knots may cause ill-  
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1105, `50%` = 3.2721, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1105, `50%` = 3.2721, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1, `50%` = 3.1323, `75%` =  
## 5.118: some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1, `50%` = 3.1323, `75%` =  
## 5.118: some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.92938, `40%` = 2.55946, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.92938, `40%` = 2.55946, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.93736, `40%` = 2.59666, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.93736, `40%` = 2.59666, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.86156666666667, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.86156666666667, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.79777142857143, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.79777142857143, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.7936, `28.57143%`  
## = 2.16771428571429, : some 'x' values beyond boundary knots may cause ill-  
## conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.7936, `28.57143%`  
## = 2.16771428571429, : some 'x' values beyond boundary knots may cause ill-  
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.734325, `25%` = 2.0941, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.734325, `25%` = 2.0941, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.751575, `25%` = 2.1084, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.751575, `25%` = 2.1084, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`11.11111%` = 1.71552222222222, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`11.11111%` = 1.71552222222222, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`11.11111%` = 1.66286666666667, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`11.11111%` = 1.66286666666667, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.62008, `20%` = 1.92938, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.62008, `20%` = 1.92938, :  
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.6283, `20%` = 1.9512, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.6283, `20%` = 1.9512, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61225454545455, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61225454545455, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61066363636364, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61066363636364, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.60476666666667, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.60476666666667, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.5881, `16.66667%`  
## = 1.82231666666667, : some 'x' values beyond boundary knots may cause ill-  
## conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.5881, `16.66667%`  
## = 1.82231666666667, : some 'x' values beyond boundary knots may cause ill-  
## conditioned bases

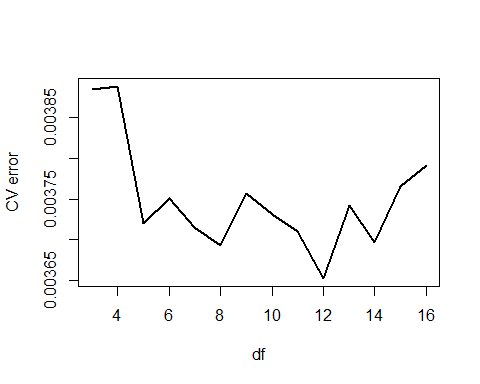
## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.58949230769231, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.58949230769231, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.5741, `15.38462%` =  
## 1.8209, : some 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.5741, `15.38462%` =  
## 1.8209, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.54201428571429, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.54201428571429, : some  
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.5768, `14.28571%`  
## = 1.81652857142857, : some 'x' values beyond boundary knots may cause ill-  
## conditioned bases  
  
## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.5768, `14.28571%`  
## = 1.81652857142857, : some 'x' values beyond boundary knots may cause ill-  
## conditioned bases

plot(3:16, all.cv[-c(1, 2)], lwd = 2, type = "l", xlab = "df", ylab = "CV error")



CV error is jumpier in this case, but attains minimum at df=10. We pick 10 as the optimal degrees of freedom.