Lecture 1 0912

JongCheolLee

2024-09-02

Data1

```r
## Open the dataset linked to the book website
url.ad <- "https://www.statlearning.com/s/Advertising.csv"
Advertising <- read.csv(url.ad, h=T)
attach(Advertising)
```

- attach() R ,

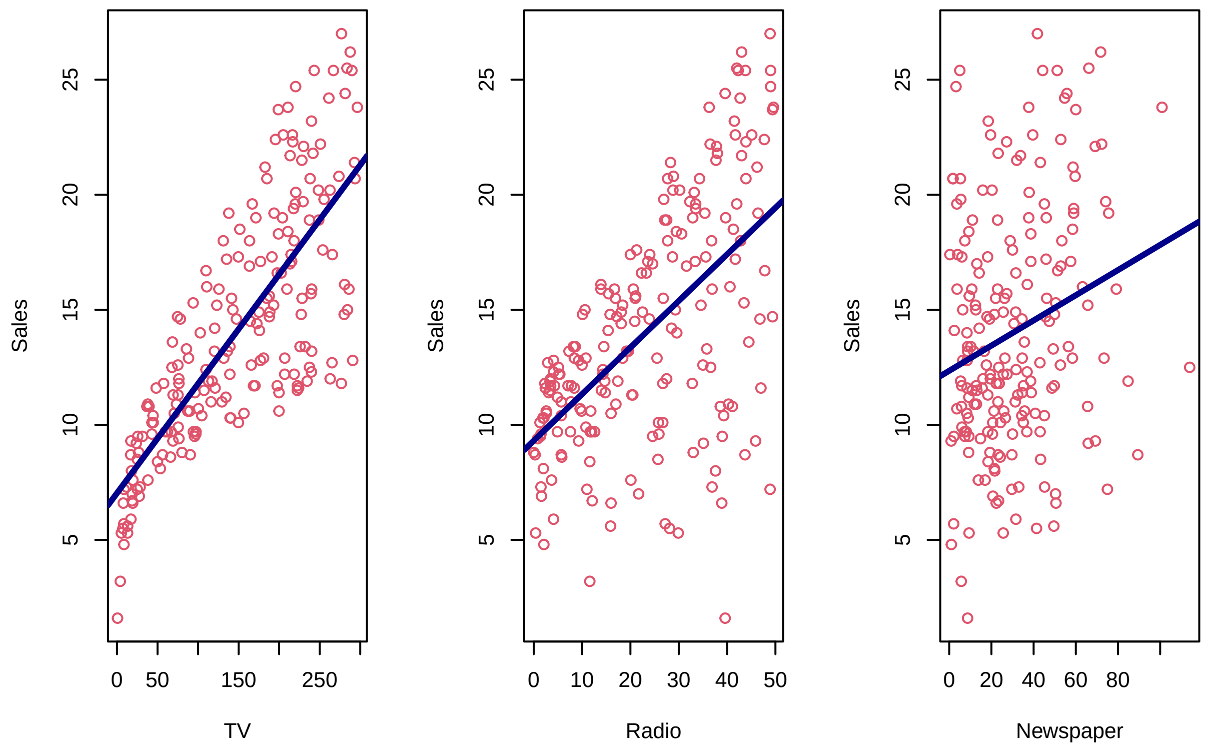
```r
head(Advertising)
```

```
##   X    TV radio newspaper sales
## 1 1 230.1  37.8      69.2  22.1
## 2 2  44.5  39.3      45.1  10.4
## 3 3  17.2  45.9      69.3   9.3
## 4 4 151.5  41.3      58.5  18.5
## 5 5 180.8  10.8      58.4  12.9
## 6 6   8.7  48.9      75.0   7.2
```

```r
summary(Advertising)
```

```
##        X                TV             radio          newspaper     
##  Min.   :  1.00   Min.   :  0.70   Min.   : 0.000   Min.   :  0.30  
##  1st Qu.: 50.75   1st Qu.: 74.38   1st Qu.: 9.975   1st Qu.: 12.75  
##  Median :100.50   Median :149.75   Median :22.900   Median : 25.75  
##  Mean   :100.50   Mean   :147.04   Mean   :23.264   Mean   : 30.55  
##  3rd Qu.:150.25   3rd Qu.:218.82   3rd Qu.:36.525   3rd Qu.: 45.10  
##  Max.   :200.00   Max.   :296.40   Max.   :49.600   Max.   :114.00  
##      sales      
##  Min.   : 1.60  
##  1st Qu.:10.38  
##  Median :12.90  
##  Mean   :14.02  
##  3rd Qu.:17.40  
##  Max.   :27.00  
```

```r
## Least square fit for simple linear regression
par(mfrow = c(1,3))
plot(sales~TV, col=2, xlab="TV", ylab="Sales")
abline(lm(sales~TV)$coef, lwd=3, col="darkblue")
plot(sales~radio, col=2, xlab="Radio", ylab="Sales")
abline(lm(sales~radio)$coef, lwd=3, col="darkblue")
plot(sales~newspaper, col=2, xlab="Newspaper", ylab="Sales")
abline(lm(sales~newspaper)$coef, lwd=3, col="darkblue")
```



Multiple LR

```r
AD <- Advertising[ ,-1] #
```

```r
## Multiple linear regression
lm.fit <- lm(sales ~., AD)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = sales ~ ., data = AD)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -8.8277 -0.8908  0.2418  1.1893  2.8292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.938889   0.311908   9.422   <2e-16 ***
## TV           0.045765   0.001395  32.809   <2e-16 ***
## radio        0.188530   0.008611  21.893   <2e-16 ***
## newspaper   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.686 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

```r
names(lm.fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```r
coef(lm.fit)
```

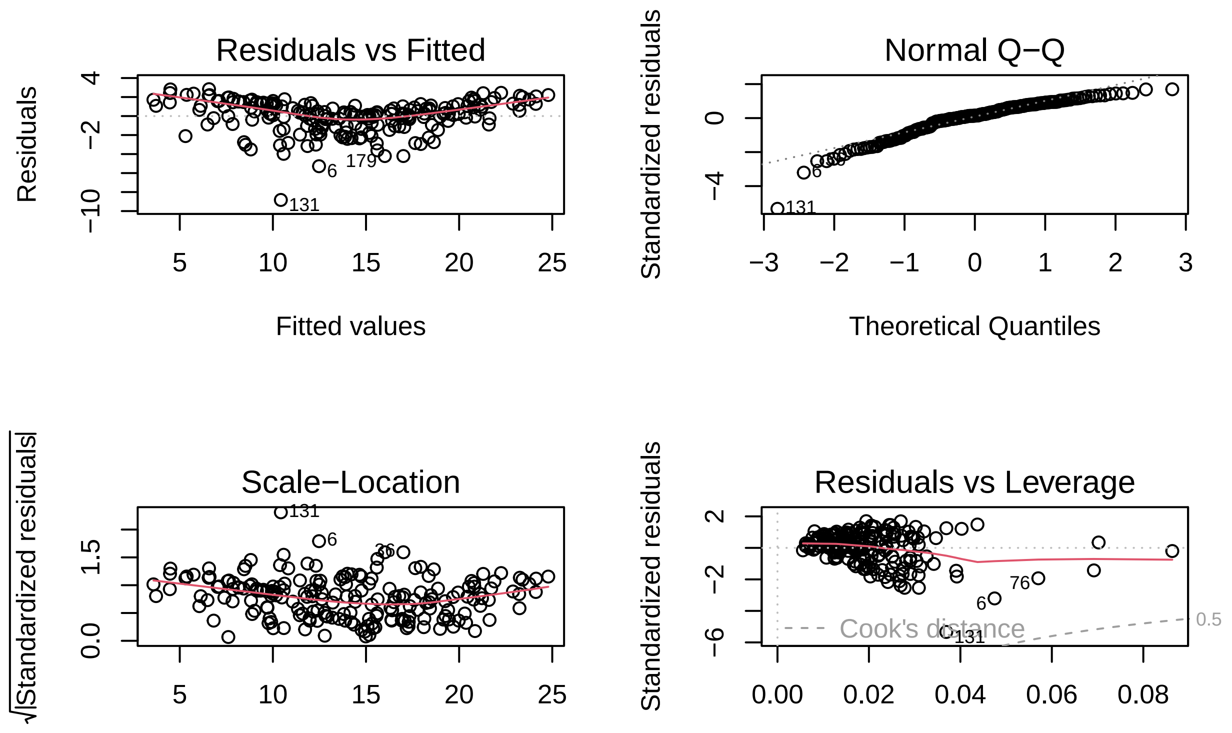
```
##  (Intercept)           TV        radio    newspaper
##  2.938889369  0.045764645  0.188530017 -0.001037493
```

```r
confint(lm.fit)
```

```
##                   2.5 %     97.5 %
## (Intercept)  2.32376228 3.55401646
## TV           0.04301371 0.04851558
## radio        0.17154745 0.20551259
## newspaper   -0.01261595 0.01054097
```

- Newspaper p Newspaper

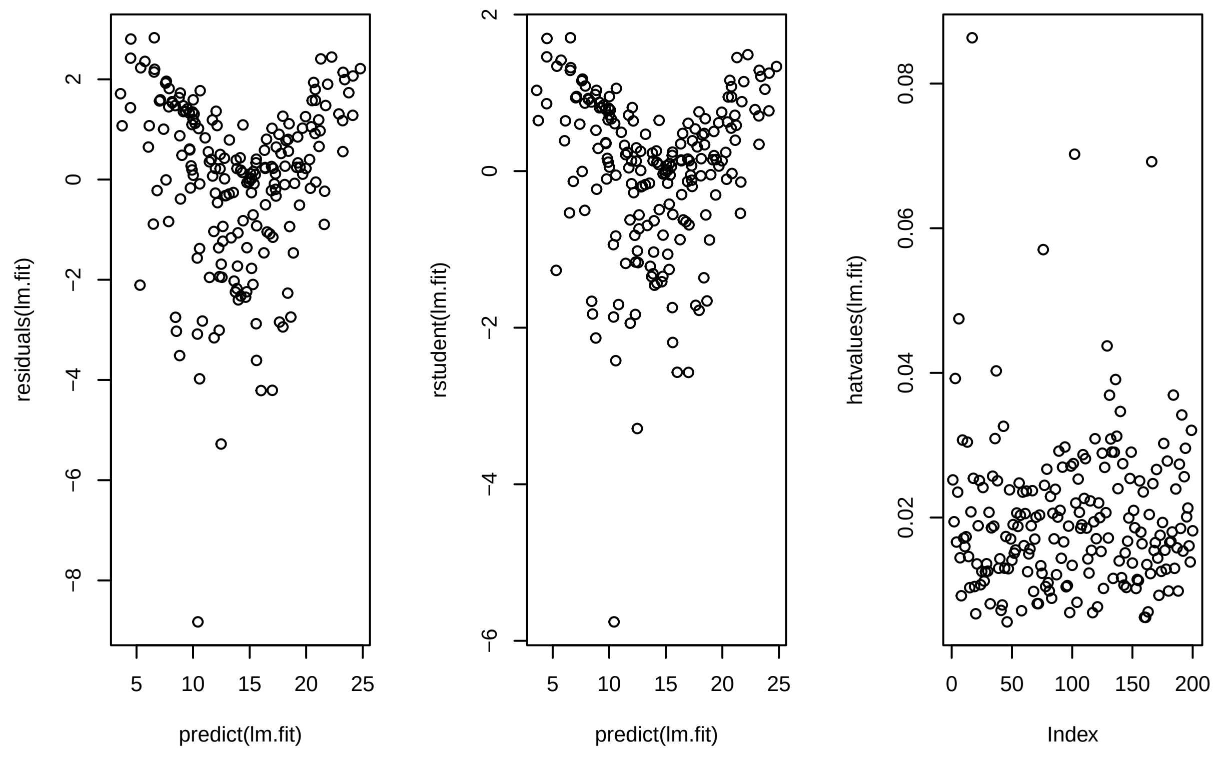
```r
par(mfrow=c(2,2))
plot(lm.fit)
```



```
#dev.off()
```

- Residual Plot: . .
- Q-Q plot: .
- Standardized Residuals:
- Studentized Residuals: (_i)/sqrt(1-h_{ii}). Leverage . .
- Leverage(= Hat value): . Hat matrix . Hat value . Y_hat = HY , i hat value i . ('Hat matrix ')
- Leverage Leverage .
- .

```
par(mfrow=c(1,3))
plot(predict(lm.fit), residuals(lm.fit))
plot(predict(lm.fit), rstudent(lm.fit))
plot(hatvalues(lm.fit))
```



```r
which.max(hatvalues(lm.fit))
```

```
## 17
## 17
```

- hat . hat , .

Data2

```r
url.in <- "https://www.statlearning.com/s/Income1.csv"
Income <- read.csv(url.in, h=T)
attach(Income)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     Income
```

```
## The following object is masked from Advertising:
##
##     X
```

```r
head(Income)
```

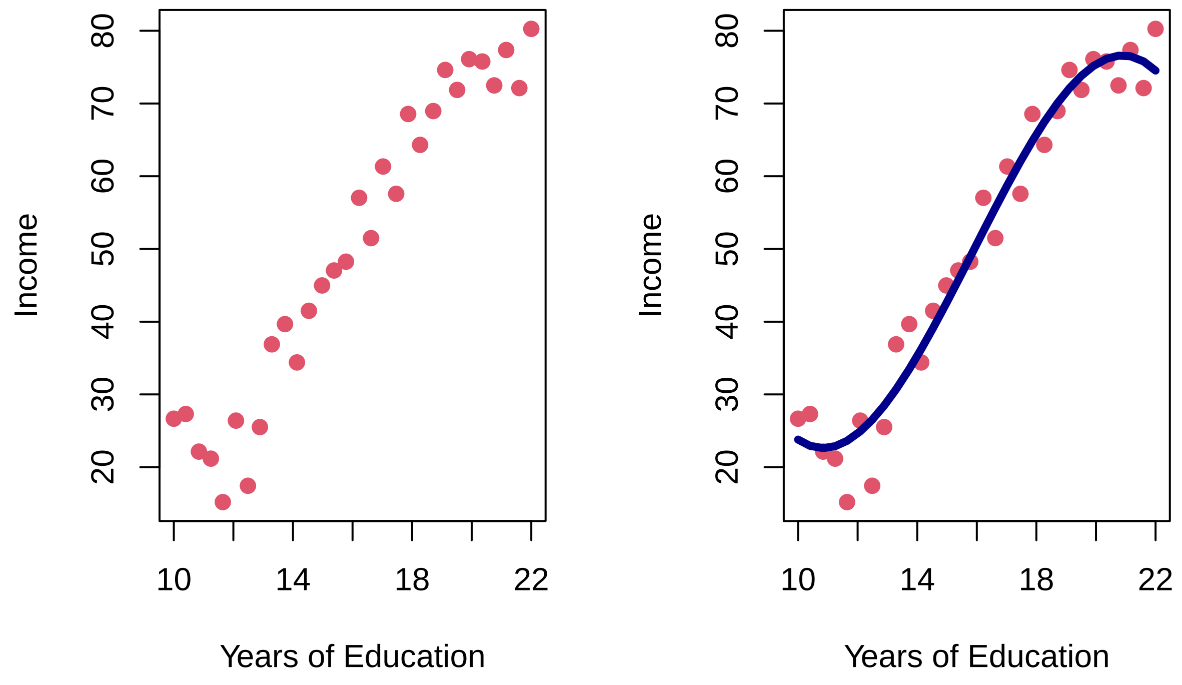
```
##   X Education   Income
## 1 1  10.00000 26.65884
## 2 2  10.40134 27.30644
## 3 3  10.84281 22.13241
## 4 4  11.24415 21.16984
## 5 5  11.64548 15.19263
## 6 6  12.08696 26.39895
```

```r
summary(Income)
```

```
##        X           Education         Income     
##  Min.   : 1.00   Min.   :10.00   Min.   :15.19  
##  1st Qu.: 8.25   1st Qu.:12.99   1st Qu.:29.08  
##  Median :15.50   Median :16.00   Median :49.87  
##  Mean   :15.50   Mean   :16.00   Mean   :50.15  
##  3rd Qu.:22.75   3rd Qu.:19.01   3rd Qu.:71.14  
##  Max.   :30.00   Max.   :22.00   Max.   :80.26
```

Polynomial regression

```r
## Polynomial regression fit
par(mfrow = c(1,2))
plot(Income~Education, col=2, pch=19, xlab="Years of Education",
ylab="Income", data=Income)
g <- lm(Income ~ poly(Education, 3), data=Income)

plot(Income~Education, col=2, pch=19, xlab="Years of Education",
ylab="Income", data=Income)
lines(Income$Education, g$fit, col="darkblue", lwd=4,
ylab="Income", xlab="Years of Education")
```



```
y <- Income$Income
mean((predict(g) - y)^2) # mean(residuals(g)^2)    .
```

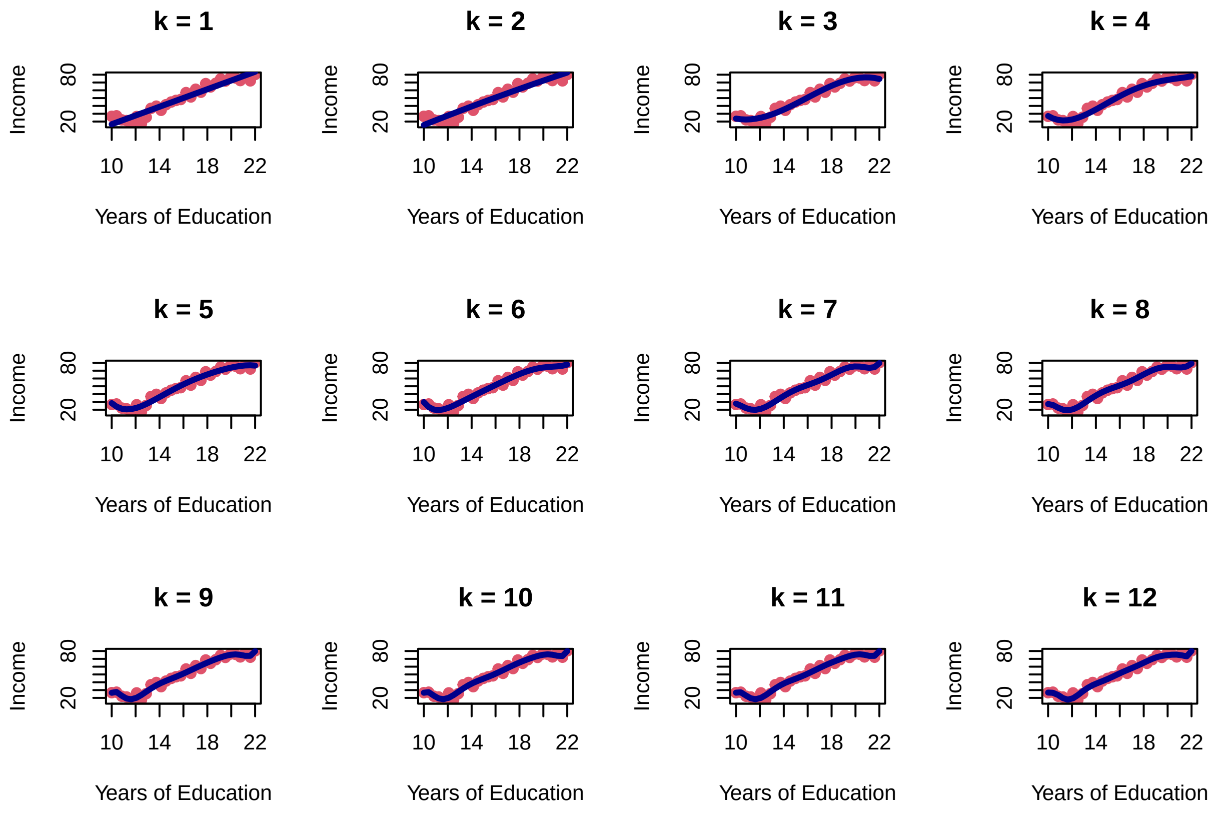
```
## [1] 15.10126
```

```
sum(predict(g) - y)
```

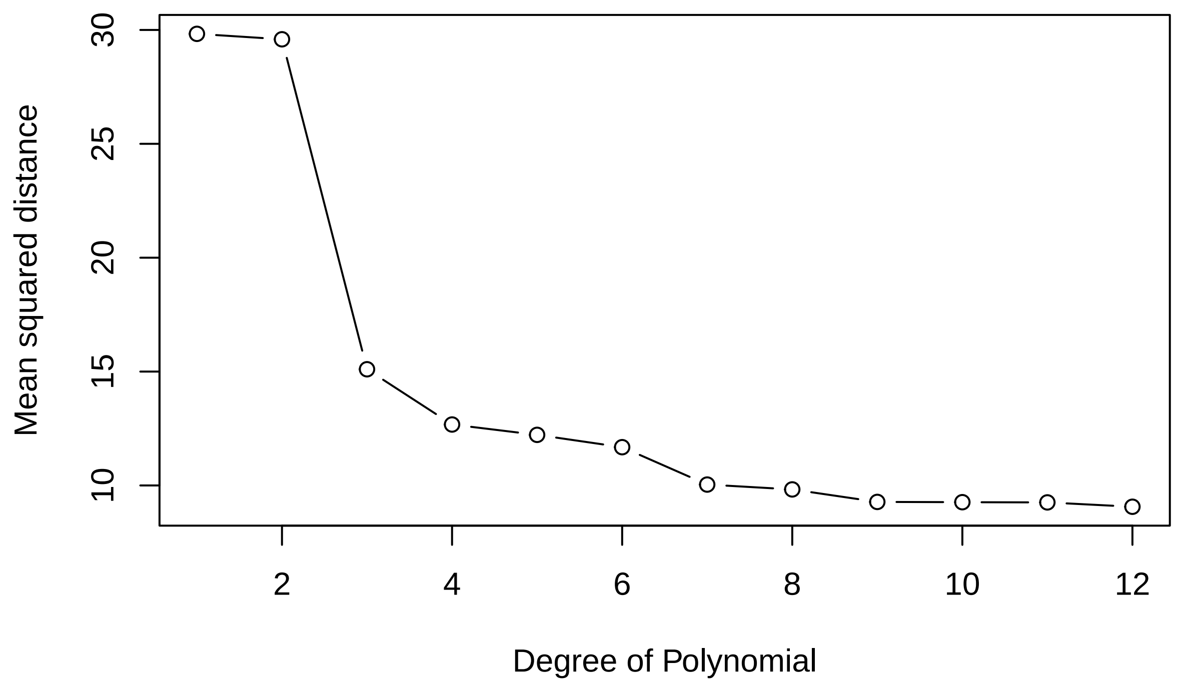
```
## [1] -1.918465e-13
```

Polynomial regression from deg 1 to 12

```
dist <- NULL
par(mfrow=c(3,4))
for (k in 1:12) {
  g <- lm(Income ~ poly(Education, k), data=Income)
  dist[k] <- mean(residuals(g)^2)
  plot(Income~Education, col=2, pch=19,
  xlab="Years of Education", ylab="Income",
  data=Income, main=paste("k =", k))
  lines(Income$Education,g$fit,col="darkblue",lwd=3,
  ylab="Income", xlab="Years of Education")
}
```



```
#x11()
plot(dist, type="b", xlab="Degree of Polynomial",
ylab="Mean squared distance")
```



```
dist # MSE
```

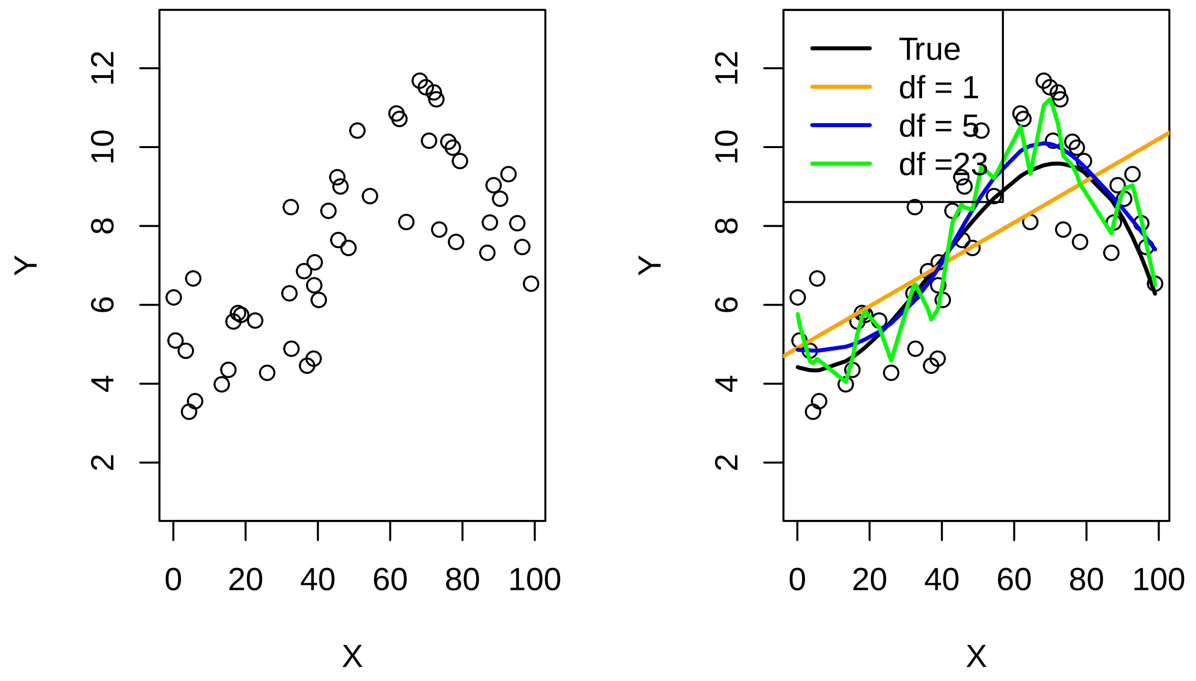
```
##  [1] 29.828816 29.590053 15.101265 12.675769 12.221552 11.680070 10.039115
##  [8]  9.825701  9.276918  9.265054  9.254865  9.064854
```

- 12 MSE . ? => Overfitting .

Training measurement vs Test measurement

```
set.seed(12345)
## Simulate x and y based on a known function
fun1 <- function(x) -(x-100)*(x-30)*(x+15)/13^4+6 # True underlying model
x <- runif(50,0,100)
y <- fun1(x) + rnorm(50) #

## Plot linear regression and splines (Prediction models)
par(mfrow=c(1,2))
plot(x, y, xlab="X", ylab="Y", ylim=c(1,13))
plot(x, y, xlab="X", ylab="Y", ylim=c(1,13))
lines(sort(x), fun1(sort(x)), col=1, lwd=2)
abline(lm(y~x)$coef, col="orange", lwd=2)

lines(smooth.spline(x,y, df=5), col="blue", lwd=2) # smoothing spline
```

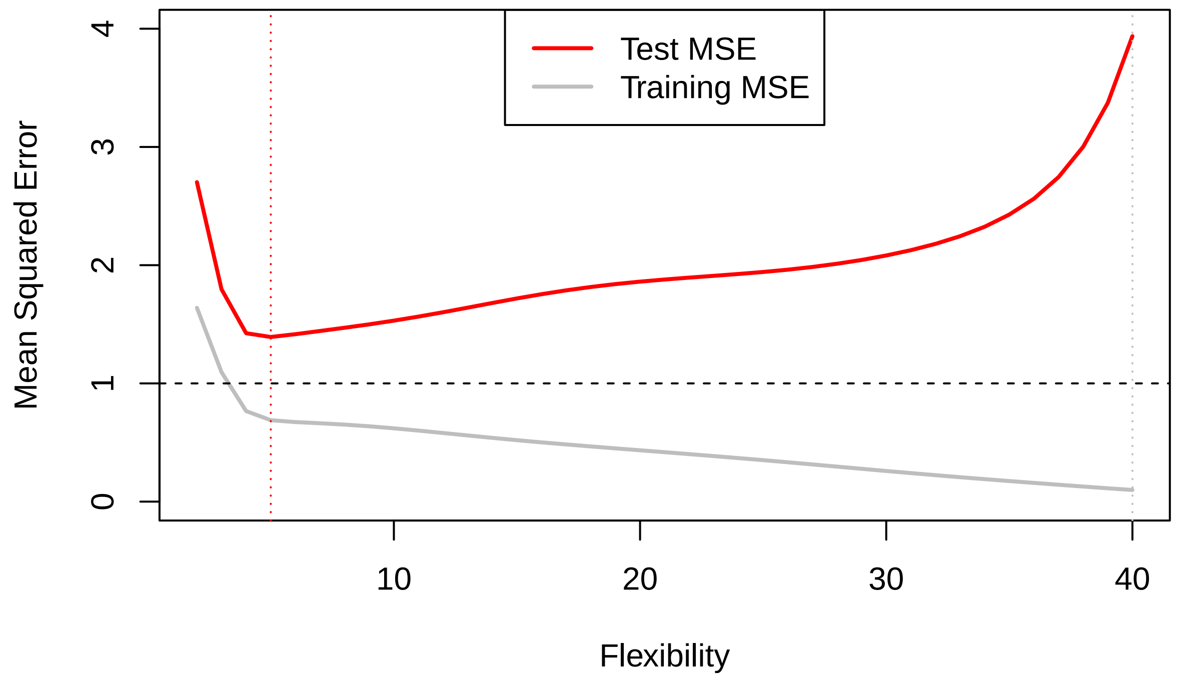
```r
lines(smooth.spline(x,y, df=23), col="green", lwd=2)
legend("topleft", lty=1, col=c(1, "orange", "blue", "green"),
legend=c("True", "df = 1", "df = 5", "df =23"),lwd=2)
```



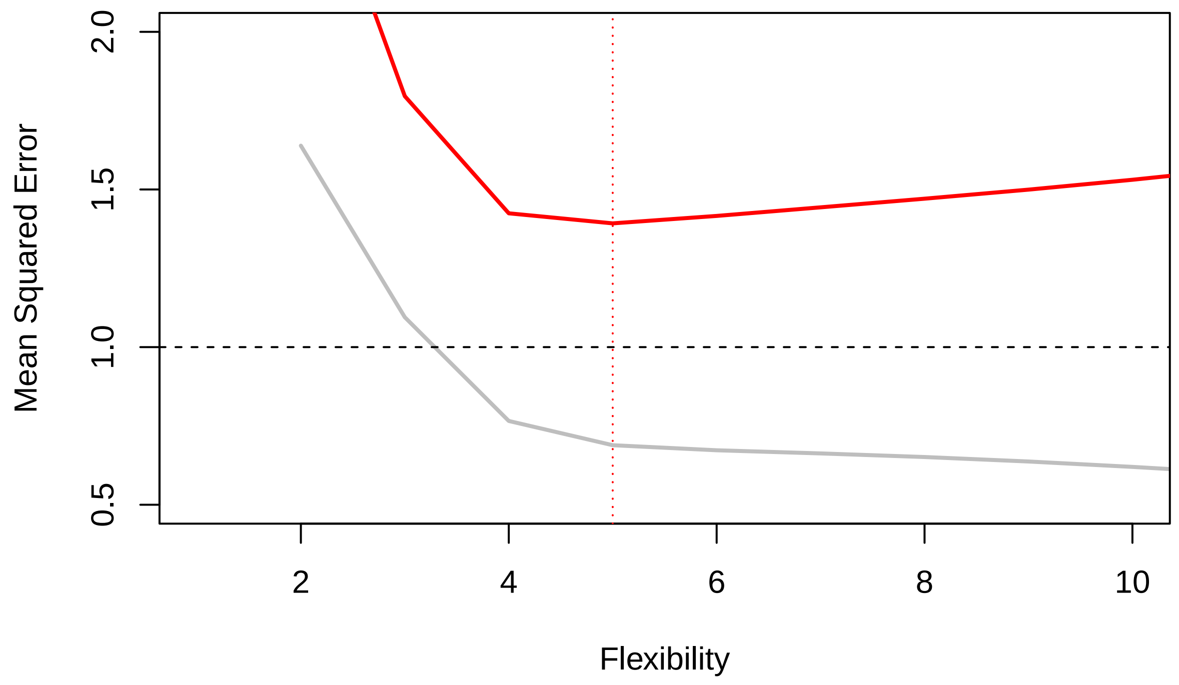
```r
set.seed(45678)
## Simulate training and test data (x, y)
tran.x <- runif(50,0,100)
test.x <- runif(50,0,100)
tran.y <- fun1(tran.x) + rnorm(50)
test.y <- fun1(test.x) + rnorm(50)

## Compute MSE along with different df
df <- 2:40
MSE <- matrix(0, length(df), 2)

for (i in 1:length(df)) {
  tran.fit <- smooth.spline(tran.x, tran.y, df=df[i])
  MSE[i,1] <- mean((tran.y - predict(tran.fit, tran.x)$y)^2) # training set
  MSE[i,2] <- mean((test.y - predict(tran.fit, test.x)$y)^2) # test set
}
```

```r
## Plot both test and training errors
matplot(df, MSE, type="l", col=c("gray", "red"),
xlab="Flexibility", ylab="Mean Squared Error",
lwd=2, lty=1, ylim=c(0,4))
```

```
abline(h=1, lty=2)
legend("top", lty=1, col=c("red", "gray"),lwd=2,
legend=c("Test MSE", "Training MSE"))
abline(v=df[which.min(MSE[,1])], lty=3, col="gray")
abline(v=df[which.min(MSE[,2])], lty=3, col="red")
```

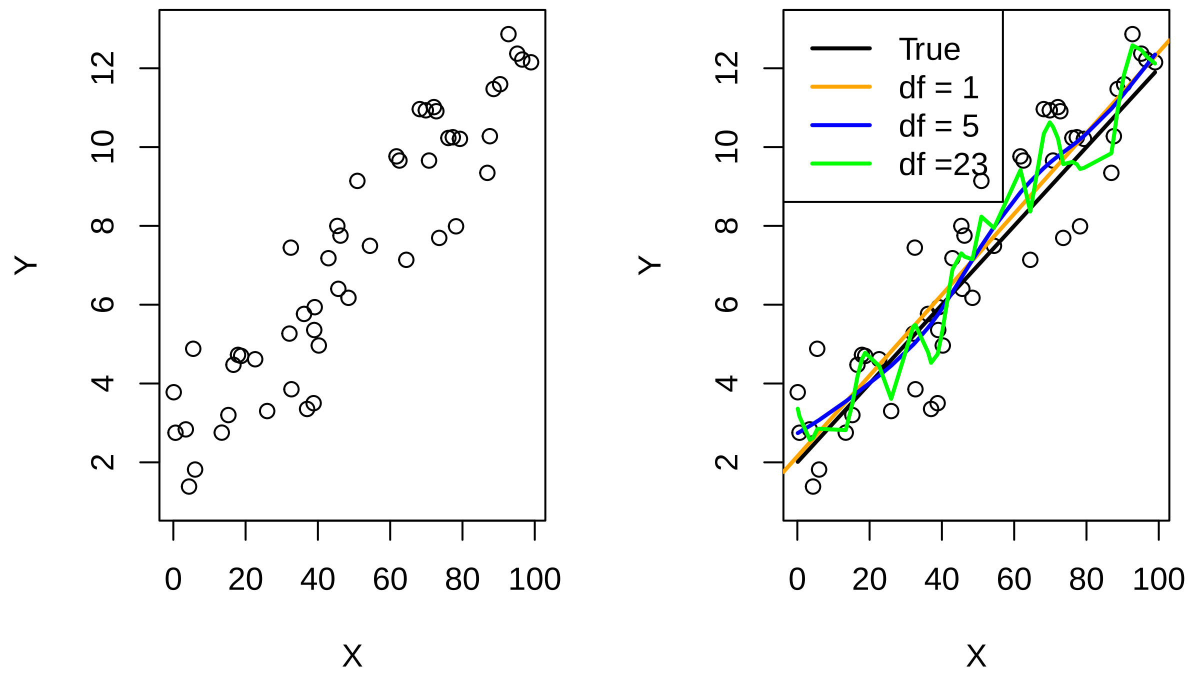


```
matplot(df, MSE, type="l", col=c("gray", "red"),
xlab="Flexibility", ylab="Mean Squared Error",
lwd=2, lty=1, ylim=c(0.5,2), xlim=c(1,10))
abline(h=1, lty=2)
abline(v=df[which.min(MSE[,2])], lty=3, col="red")
```



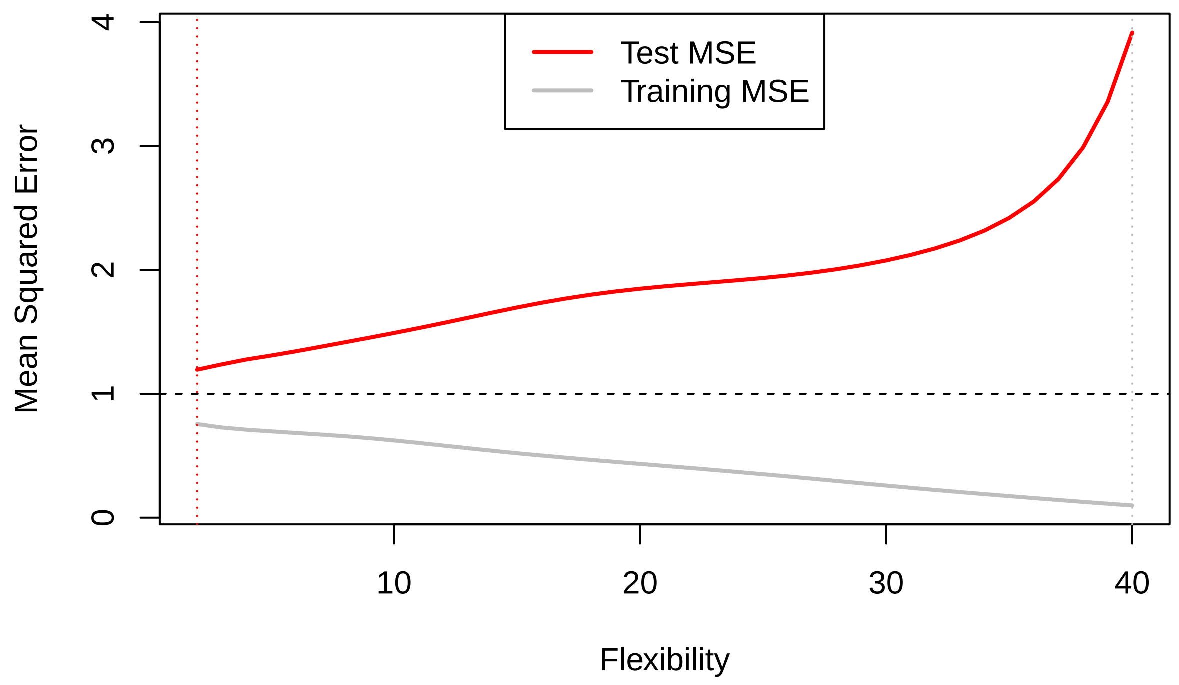
- Training MSE .
- (Unseen) Test MSE , training set .

```r
set.seed(12345)
## Simulate x and y based on a known function
fun2 <- function(x) x/10 +2 # true underlying model: Linear
x <- runif(50,0,100)
y <- fun2(x) + rnorm(50)

## Plot linear regression and splines
par(mfrow=c(1,2))
plot(x, y, xlab="X", ylab="Y", ylim=c(1,13))
plot(x, y, xlab="X", ylab="Y", ylim=c(1,13))
lines(sort(x), fun2(sort(x)), col=1, lwd=2)
abline(lm(y~x)$coef, col="orange", lwd=2)
lines(smooth.spline(x,y, df=5), col="blue", lwd=2)
lines(smooth.spline(x,y, df=23), col="green", lwd=2)
legend("topleft", lty=1, col=c(1, "orange", "blue", "green"),
legend=c("True", "df = 1", "df = 5", "df =23"),lwd=2)
```

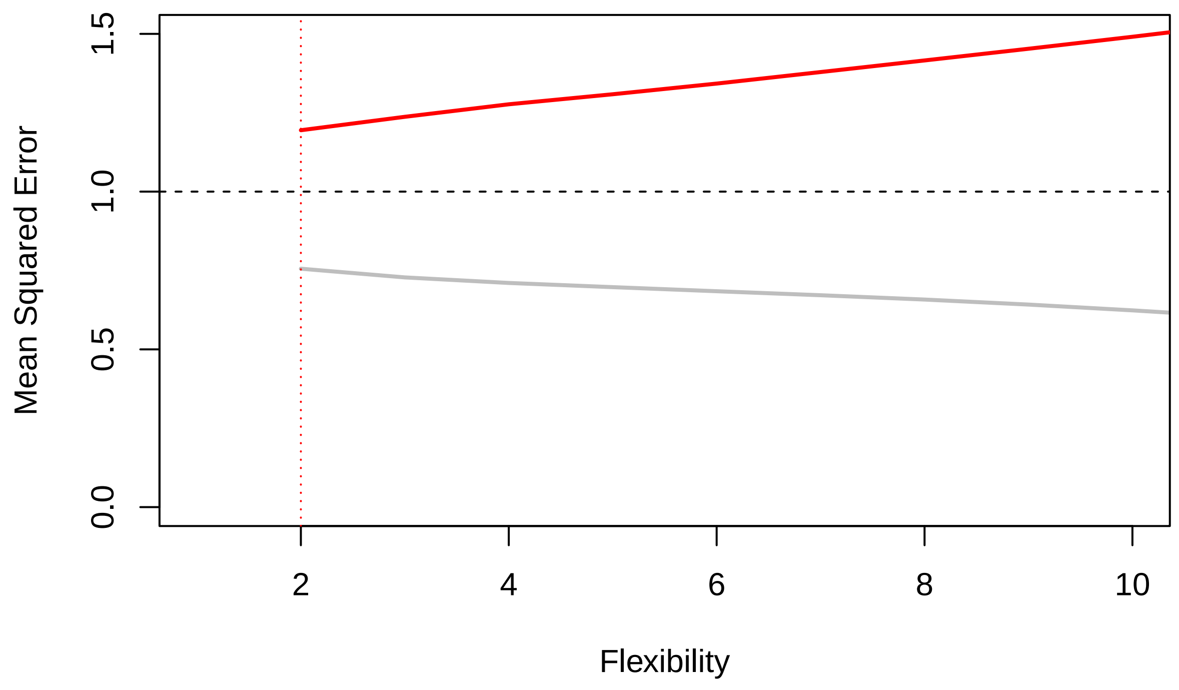


```
set.seed(45678)

## Simulate training and test data (x, y)
tran.x <- runif(50,0,100)
test.x <- runif(50,0,100)
tran.y <- fun2(tran.x) + rnorm(50)
test.y <- fun2(test.x) + rnorm(50)

## Compute MSE along with different df
df <- 2:40
MSE <- matrix(0, length(df), 2)
for (i in 1:length(df)) {
  tran.fit <- smooth.spline(tran.x, tran.y, df=df[i]) # training data  x,y
  MSE[i,1] <- mean((tran.y - predict(tran.fit, tran.x)$y)^2)
  MSE[i,2] <- mean((test.y - predict(tran.fit, test.x)$y)^2)
}
```

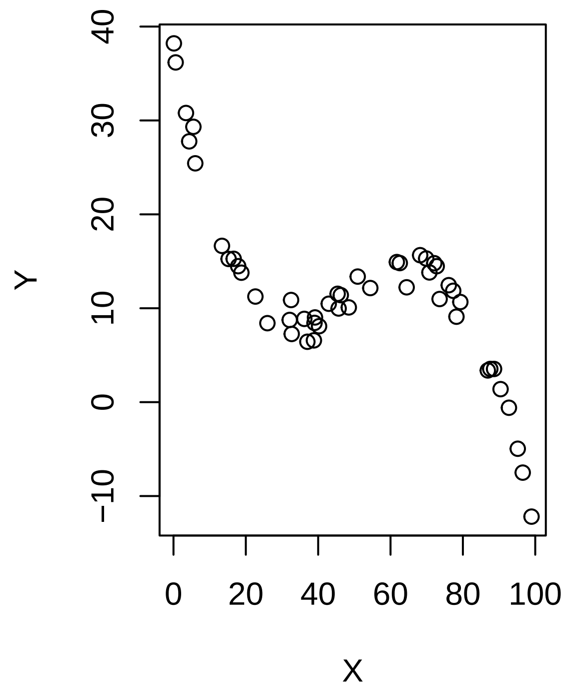
```
## Plot both test and training errors
matplot(df, MSE, type="l", col=c("gray", "red"),
xlab="Flexibility", ylab="Mean Squared Error",
lwd=2, lty=1)
abline(h=1, lty=2)
legend("top", lty=1, col=c("red", "gray"),lwd=2,
legend=c("Test MSE", "Training MSE"))
abline(v=df[which.min(MSE[,1])], lty=3, col="gray")
abline(v=df[which.min(MSE[,2])], lty=3, col="red")
```

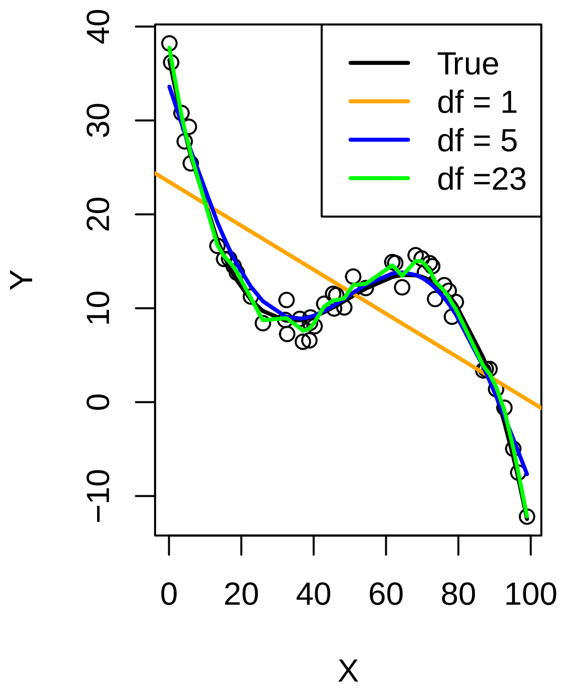


```r
matplot(df, MSE, type="l", col=c("gray", "red"),
xlab="Flexibility", ylab="Mean Squared Error",
lwd=2, lty=1, ylim=c(0,1.5), xlim=c(1,10))
abline(h=1, lty=2)
abline(v=df[which.min(MSE[,2])], lty=3, col="red")
```



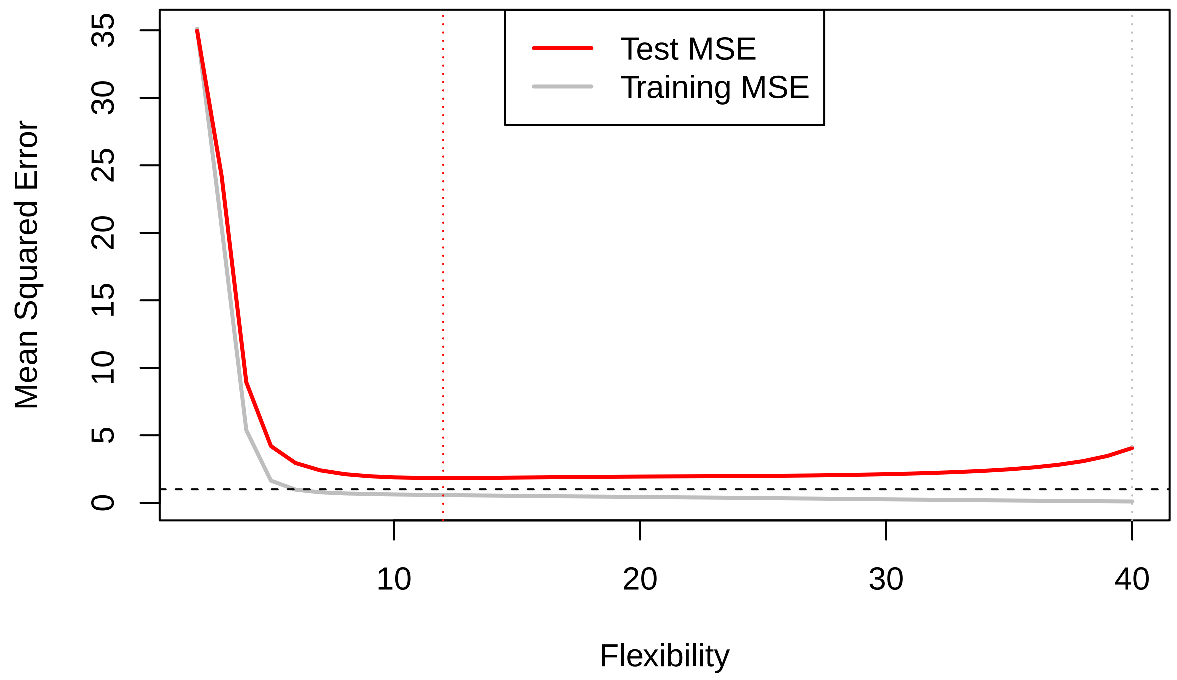
```
set.seed(12345)
## Simulate x and y based on a known function
fun3 <- function(x) -(x-80)*(x-45)*(x-25)/15^3+10
x <- runif(50,0,100)
y <- fun3(x) + rnorm(50)

## Plot linear regression and splines
par(mfrow=c(1,2))
plot(x, y, xlab="X", ylab="Y")
plot(x, y, xlab="X", ylab="Y")
lines(sort(x), fun3(sort(x)), col=1, lwd=2)
abline(lm(y~x)$coef, col="orange", lwd=2)
lines(smooth.spline(x,y, df=5), col="blue", lwd=2)
lines(smooth.spline(x,y, df=23), col="green", lwd=2)
legend("topright", lty=1, col=c(1, "orange", "blue", "green"),
legend=c("True", "df = 1", "df = 5", "df =23"),lwd=2)
```



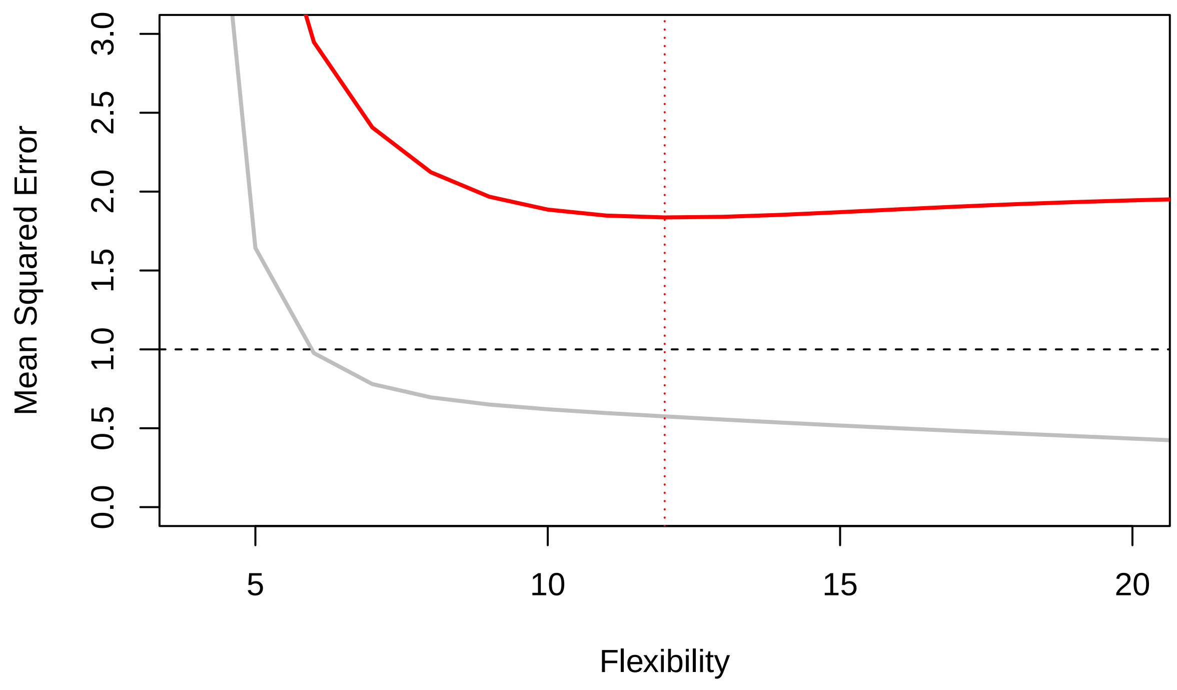


```r
set.seed(45678)
## Simulate training and test data (x, y)
tran.x <- runif(50,0,100)
test.x <- runif(50,0,100)
tran.y <- fun3(tran.x) + rnorm(50)
test.y <- fun3(test.x) + rnorm(50)

## Compute MSE along with different df
df <- 2:40
MSE <- matrix(0, length(df), 2)
for (i in 1:length(df)) {
  tran.fit <- smooth.spline(tran.x, tran.y, df=df[i])
  MSE[i,1] <- mean((tran.y - predict(tran.fit, tran.x)$y)^2)
  MSE[i,2] <- mean((test.y - predict(tran.fit, test.x)$y)^2)
}
```

```r
## Plot both test and training errors
matplot(df, MSE, type="l", col=c("gray", "red"),
xlab="Flexibility", ylab="Mean Squared Error",
lwd=2, lty=1)
abline(h=1, lty=2)
legend("top", lty=1, col=c("red", "gray"),lwd=2,
legend=c("Test MSE", "Training MSE"))
abline(v=df[which.min(MSE[,1])], lty=3, col="gray")
abline(v=df[which.min(MSE[,2])], lty=3, col="red")
```



```
matplot(df, MSE, type="l", col=c("gray", "red"),
xlab="Flexibility", ylab="Mean Squared Error",
lwd=2, lty=1, ylim=c(0,3), xlim=c(4,20))
abline(h=1, lty=2)
abline(v=df[which.min(MSE[,2])], lty=3, col="red")
```



Validation Set Approach

```r
library(ISLR)
```

```
## Warning:    'ISLR'  R   4.2.3
```

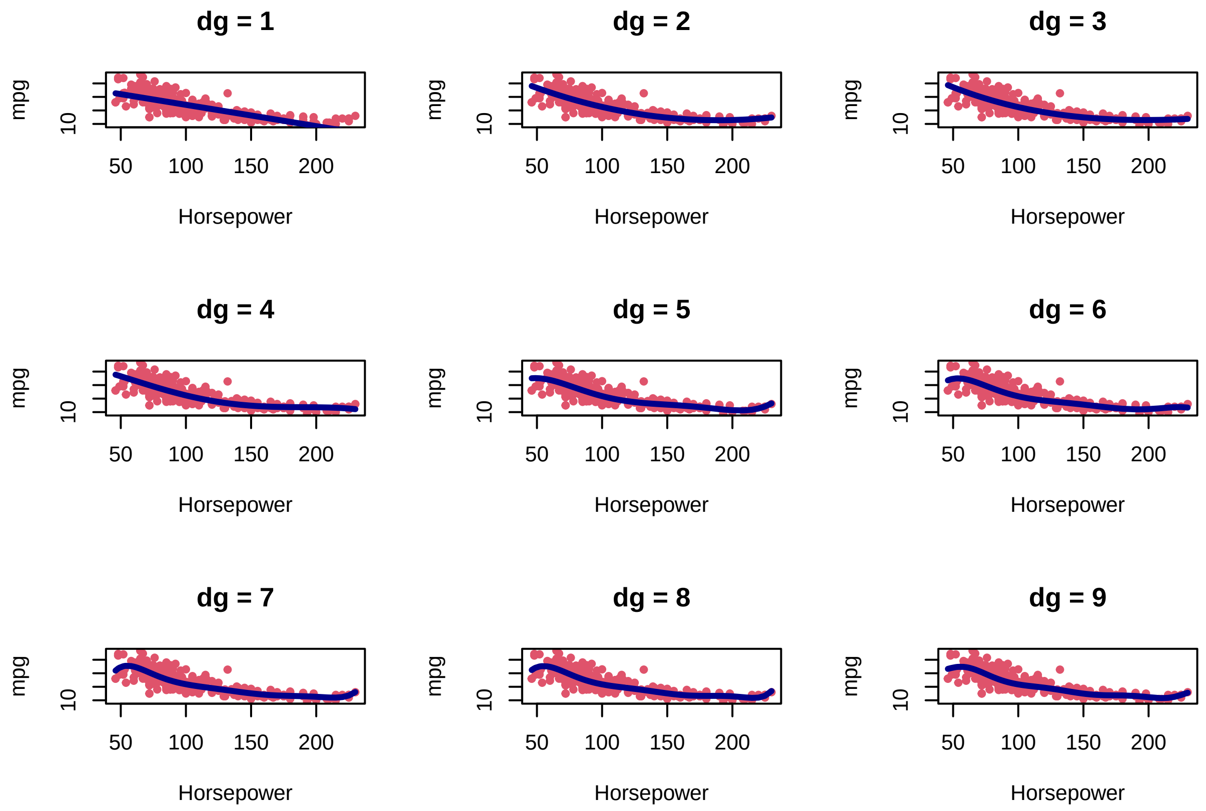
```r
data(Auto)
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241
```

```r
summary(Auto)
```

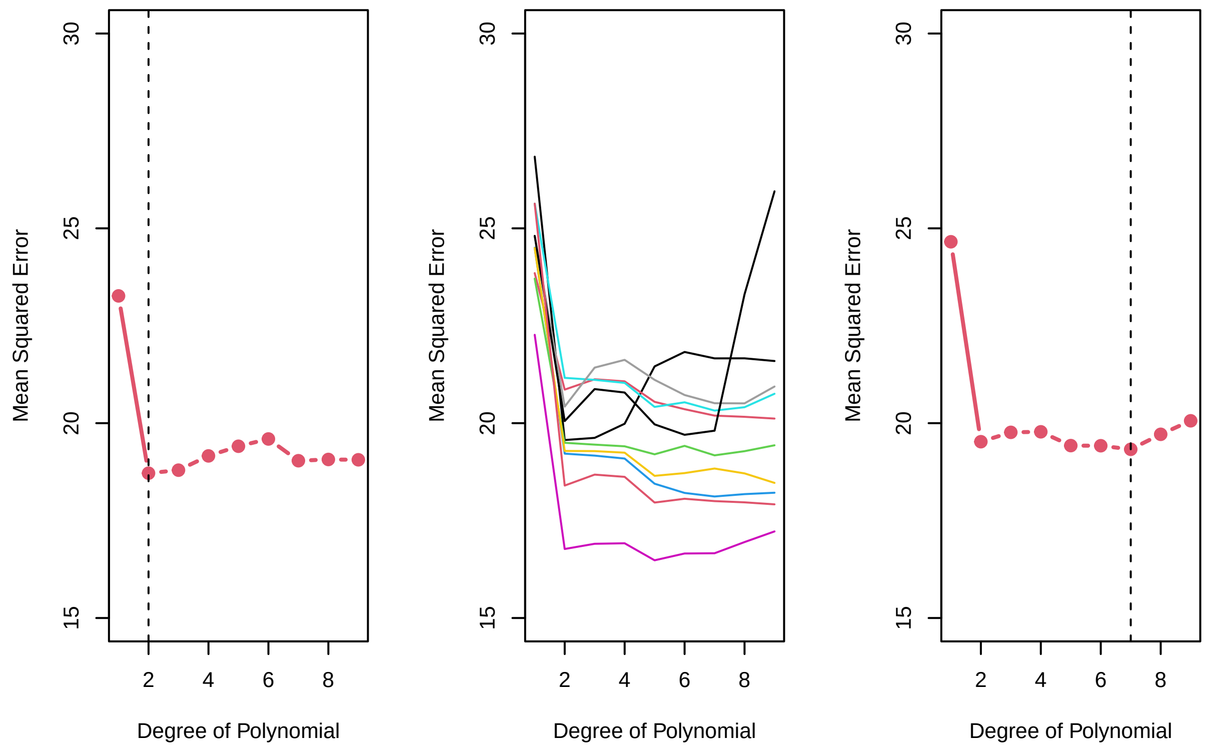
```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##
##   acceleration        year           origin                      name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5
##  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4
##                                                  (Other)           :365
```

```r
mpg <- Auto$mpg
horsepower <- Auto$horsepower
dg <- 1:9
u <- order(horsepower)

par(mfrow=c(3,3))
for (k in 1:length(dg)) {
  g <- lm(mpg ~ poly(horsepower, dg[k]))
  plot(mpg~horsepower, col=2, pch=20, xlab="Horsepower",
  ylab="mpg", main=paste("dg =", dg[k]))
  lines(horsepower[u], g$fit[u], col="darkblue", lwd=3)
}
```



```
set.seed(1)
n <- nrow(Auto)

## training set
tran <- sample(n, n/2) #
MSE <- NULL

for (k in 1:length(dg)) {
  g <- lm(mpg ~ poly(horsepower, dg[k]), subset=tran) # training set
  MSE[k] <- mean((mpg - predict(g, Auto))[-tran]^2) #    MSE
}

par(mfrow=c(1,3))
plot(dg, MSE, type="b", col=2, xlab="Degree of Polynomial",
ylab="Mean Squared Error", ylim=c(15,30), lwd=2, pch=19)
abline(v=which.min(MSE), lty=2)

K <- 10
MSE <- matrix(0, length(dg), K)

for (i in 1:K) {
  tran <- sample(392, 196) # K     training set     (K      )
  for (k in 1:length(dg)) { #
    g <- lm(mpg ~ poly(horsepower, dg[k]), subset=tran)
    MSE[k, i] <- mean((mpg - predict(g, Auto))[-tran]^2)
  }
```

```r
}

matplot(dg, MSE, type="l", xlab="Degree of Polynomial", lty=1,
ylab="Mean Squared Error", col=1:10, ylim=c(15,30))

avg <- apply(MSE, 1, mean) # K

plot(dg, avg, type="b", col=2, xlab="Degree of Polynomial",
ylab="Mean Squared Error", ylim=c(15,30), lwd=2, pch=19)
abline(v=which.min(avg), lty=2)
```



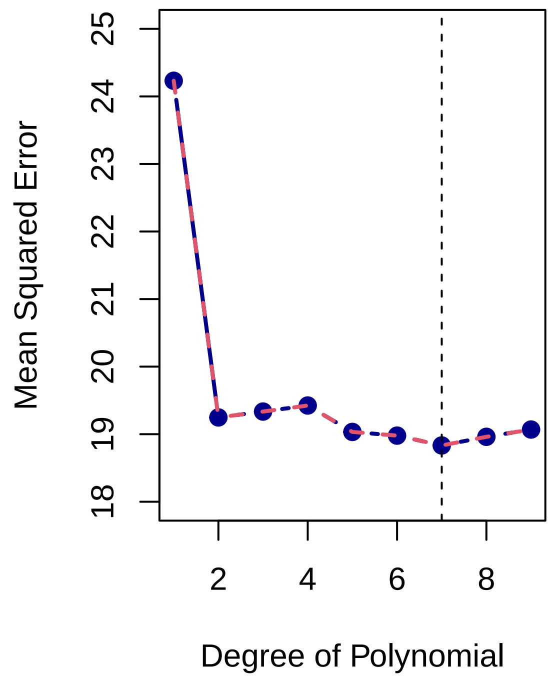
- training set   .

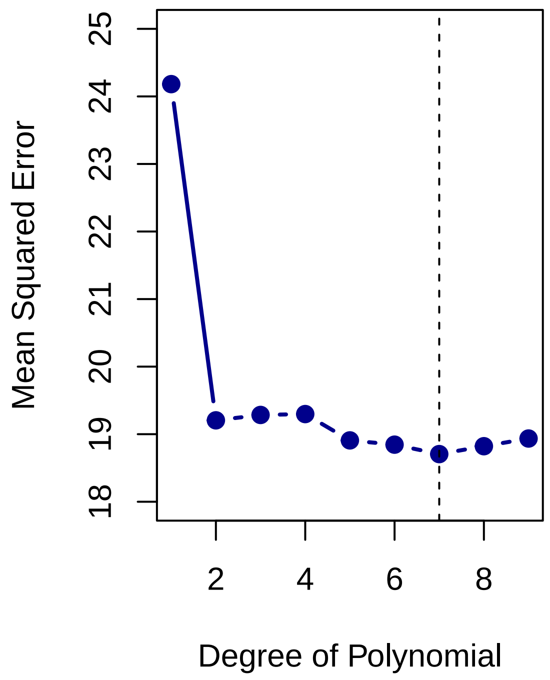
Leave one out cross validation

```r
n <- nrow(Auto)
dg <- 1:9
MSE <- matrix(0, n, length(dg))

for (i in 1:n) { # validation
  for (k in 1:length(dg)) {
    g <- lm(mpg ~ poly(horsepower, k), subset=(1:n)[-i]) # i
    MSE[i, k] <- mean((mpg - predict(g, Auto))[i]^2) # i        mse
  }
}

aMSE <- apply(MSE, 2, mean)

par(mfrow=c(1, 2))
plot(dg, aMSE, type="b", col="darkblue",
xlab="Degree of Polynomial", ylab="Mean Squared Error",
ylim=c(18,25), lwd=2, pch=19)
abline(v=which.min(aMSE), lty=2)

ncv <- NULL

for (k in 1:length(dg)) {
  g <- lm(mpg ~ poly(horsepower, k))
  ncv[k] <- mean((g$res/(1-influence(g)$hat))^2) # influence()        leverage
    # for loop  n*k                        .
}
lines(dg, ncv, col=2, lty=2, lwd=2)

K <- 10 ## 10-fold cross validation.     obs            10    .
MSE <- matrix(0, n, length(dg))

set.seed(54321)
u <- sample(rep(seq(K), length=n))
table(u)
```

```
## u
##  1  2  3  4  5  6  7  8  9 10
## 40 40 39 39 39 39 39 39 39 39
```

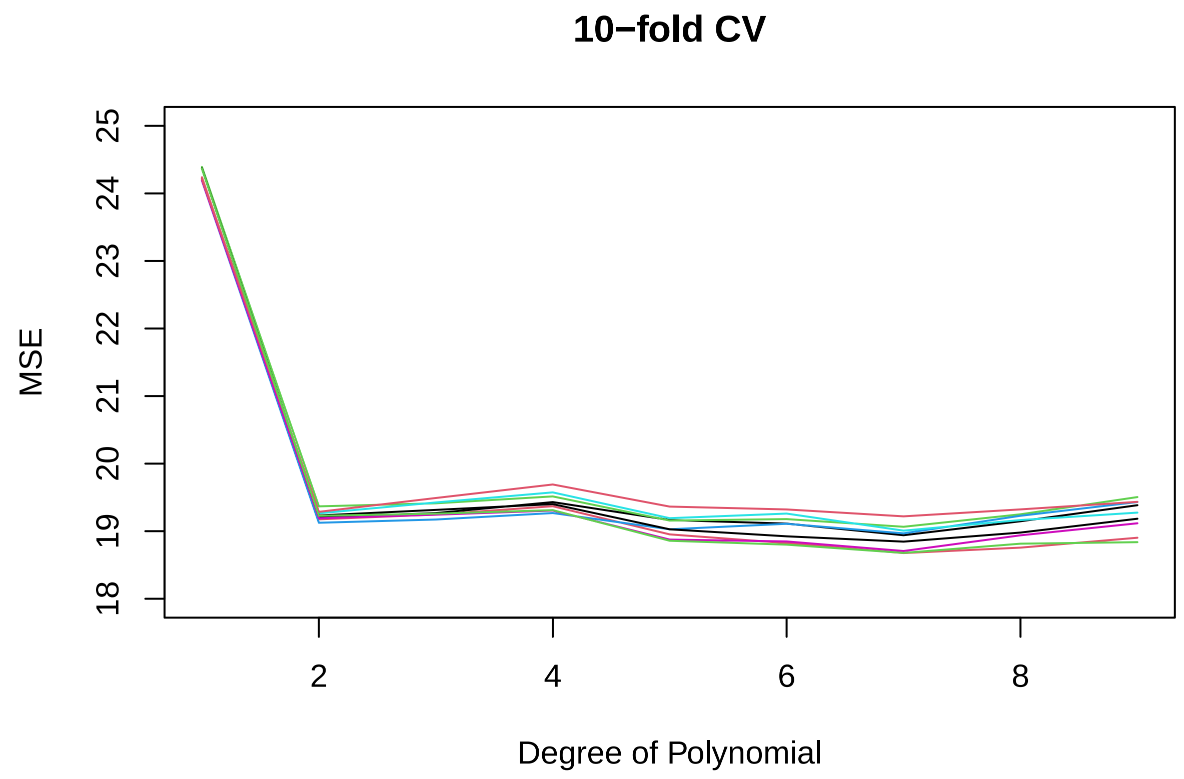
```r
for (k in 1:K) {
  tran <- which(u!=k)
  test <- which(u==k)
  for (i in 1:length(dg)) {
    g <- lm(mpg ~ poly(horsepower, i), subset=tran)
    MSE[test, i] <- (mpg - predict(g, Auto))[test]^2
  }
}
CVE <- apply(MSE, 2, mean)

plot(dg, CVE, type="b", col="darkblue",
xlab="Degree of Polynomial", ylab="Mean Squared Error",
ylim=c(18,25), lwd=2, pch=19)
abline(v=which.min(CVE), lty=2)
```





K-fold CV

```
N <- 9 ## Number of K-fold CV replications
KCV <- matrix(0, length(dg), N)

set.seed(1234)
for (j in 1:N) {
  MSE <- matrix(0, n, length(dg))
  u <- sample(rep(seq(K), length=n))
  for (k in 1:K) {
    tran <- which(u!=k)
    test <- which(u==k)
    for (i in 1:length(dg)) {
      g <- lm(mpg ~ poly(horsepower, i), subset=tran)
      MSE[test, i] <- (mpg - predict(g, Auto))[test]^2
    }
  }
  KCV[,j] <- apply(MSE, 2, mean)
}

matplot(dg, KCV, type="l", xlab="Degree of Polynomial", lty=1,
ylab="MSE", ylim=c(18,25), main="10-fold CV")
```



```
apply(KCV, 2, which.min)
```

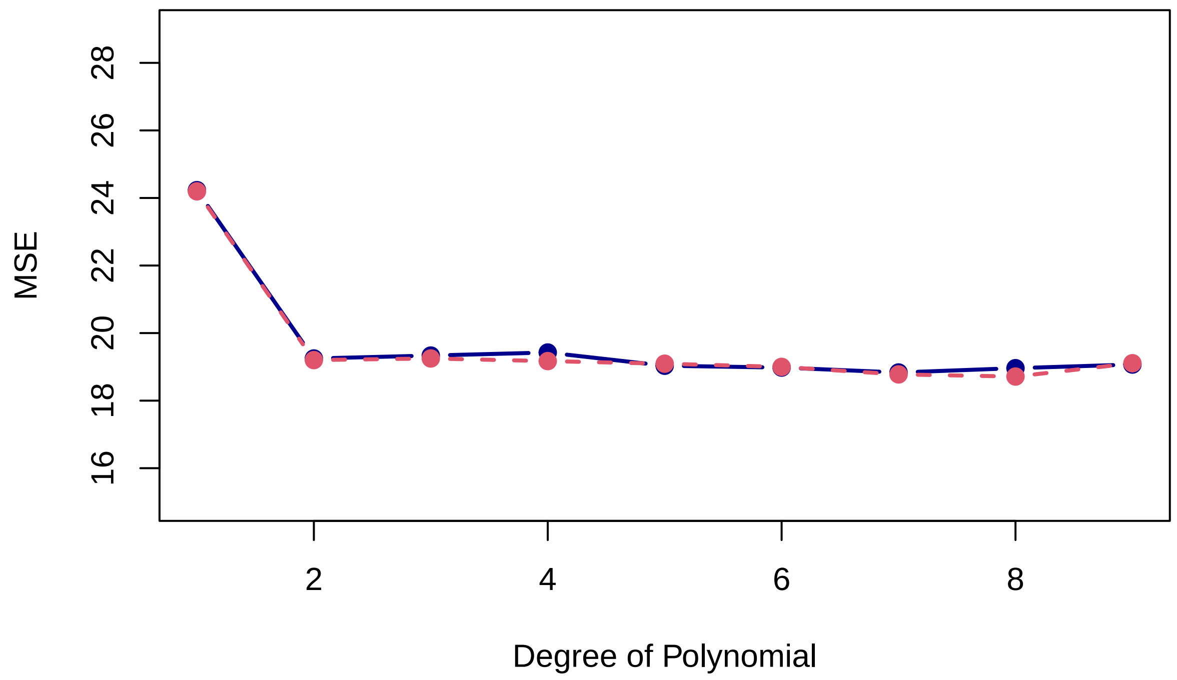
```
## [1] 7 7 7 7 7 7 7 7 7 7
```

- K   . => K-fold CV   .

boot package : CV

```
library(boot)
set.seed(101010)

## Leave-one-out CV
MSE <- NULL
for (i in 1:length(dg)) {
  glm.fit <- glm(mpg ~ poly(horsepower ,i))
  MSE[i] <- cv.glm(Auto, glm.fit)$delta[1]
}
plot(dg, MSE, type="b", col="darkblue", ylim=c(15,29),
xlab="Degree of Polynomial", ylab="MSE", lwd=2, pch=19)

## K-fold cross validation
K <- 10
KCV <- NULL
```

```r
for (i in 1:length(dg)) {
  glm.fit <- glm(mpg ~ poly(horsepower ,i))
  KCV[i] <- cv.glm(Auto, glm.fit, K=K)$delta[1]
}
lines(dg, KCV, col=2, lwd=2, type="b", pch=19, lty=2)
```



Cubic Model with K-fold Cross Validation

- training set df MSE , CV .

```r
set.seed(45678)
x <- runif(50,0,100)
y <- fun1(x) + rnorm(50) # cubic function + noise

K <- 5
df <- 2:40 #   39

MSE <- matrix(0, length(x), length(df))
u <- sample(rep(seq(K), length=length(x))) # 1:K

for (k in 1:K) {
  tr <- which(u!=k)
  te <- which(u==k)
  for (j in 1:length(df)) {
    fit <- smooth.spline(x[tr], y[tr], df=df[j])
    MSE[te, j] <- y[te] - predict(fit, x[te])$y
  }
}

CVE <- apply(MSE^2, 2, mean)
data.frame(DF=df, CVE=CVE)
```

```
##    DF        CVE
## 1   2   1.826783
## 2   3   1.149435
## 3   4   1.050967
## 4   5   1.082088
## 5   6   1.145289
## 6   7   1.225121
## 7   8   1.309276
## 8   9   1.388198
## 9  10   1.456269
## 10 11   1.513155
## 11 12   1.561547
## 12 13   1.605483
## 13 14   1.649503
## 14 15   1.697560
## 15 16   1.752622
## 16 17   1.816434
## 17 18   1.889711
## 18 19   1.973275
## 19 20   2.066752
## 20 21   2.169945
## 21 22   2.283464
## 22 23   2.408992
## 23 24   2.551177
## 24 25   2.717405
## 25 26   2.918233
## 26 27   3.171596
## 27 28   3.493487
## 28 29   3.901390
## 29 30   4.410251
## 30 31   5.014618
## 31 32   5.684375
## 32 33   6.370503
## 33 34   7.015428
## 34 35   7.603206
## 35 36   8.240476
## 36 37   9.688811
## 37 38  15.674305
## 38 39  40.679175
## 39 40 113.227168
```

Linear Model with K-fold Cross Validation

```
set.seed(45678)
x <- runif(50,0,100)
y <- fun2(x) + rnorm(50)

K <- 5
df <- 2:40

MSE <- matrix(0, length(x), length(df))
u <- sample(rep(seq(K), length=length(x)))

for (k in 1:K) {
  tr <- which(u!=k)
  te <- which(u==k)
  for (j in 1:length(df)) {
    fit <- smooth.spline(x[tr], y[tr], df=df[j])
    MSE[te, j] <- y[te] - predict(fit, x[te])$y
  }
}

CVE <- apply(MSE^2, 2, mean)
data.frame(DF=df, CVE=CVE)
```

```
##    DF       CVE
## 1   2  1.037058
## 2   3  1.047150
## 3   4  1.061881
## 4   5  1.109273
## 5   6  1.175957
## 6   7  1.253769
## 7   8  1.334445
## 8   9  1.410614
## 9  10  1.476878
## 10 11  1.532417
## 11 12  1.579600
## 12 13  1.622357
## 13 14  1.665218
## 14 15  1.712177
## 15 16  1.766228
## 16 17  1.829125
## 17 18  1.901586
## 18 19  1.984428
## 19 20  2.077256
## 20 21  2.179865
## 21 22  2.292861
## 22 23  2.417904
## 23 24  2.559636
## 24 25  2.725429
## 25 26  2.925834
## 26 27  3.178782
## 27 28  3.500271
## 28 29  3.907779
## 29 30   4.416252
## 30 31   5.020225
## 31 32   5.689573
## 32 33   6.375282
## 33 34   7.019800
## 34 35   7.607257
## 35 36   8.244439
## 36 37   9.693230
## 37 38  15.680525
## 38 39  40.689656
## 39 40 113.244389
```

Nonlinear Model with K-fold Cross Validation

```
set.seed(45678)
x <- runif(50,0,100)
y <- fun3(x) + rnorm(50)

K <- 5
df <- 2:40

MSE <- matrix(0, length(x), length(df))
u <- sample(rep(seq(K), length=length(x)))

for (k in 1:K) {
  tr <- which(u!=k)
  te <- which(u==k)
  for (j in 1:length(df)) {
    fit <- smooth.spline(x[tr], y[tr], df=df[j])
    MSE[te, j] <- y[te] - predict(fit, x[te])$y
  }
}

CVE <- apply(MSE^2, 2, mean)
data.frame(DF=df, CVE=CVE)
```

```
##    DF       CVE
## 1   2 34.722550
## 2   3 21.947697
## 3   4  6.961509
## 4   5  2.682327
## 5   6  1.751877
## 6   7  1.491226
## 7   8  1.434082
## 8   9  1.443439
## 9  10  1.472216
## 10 11  1.505855
## 11 12  1.540830
## 12 13  1.577728
## 13 14  1.618553
## 14 15  1.665584
## 15 16  1.720771
## 16 17   1.785246
## 17 18   1.859386
## 18 19   1.943828
## 19 20   2.038145
## 20 21   2.142112
## 21 22   2.256258
## 22 23   2.382320
## 23 24   2.524912
## 24 25   2.691465
## 25 26   2.892544
## 26 27   3.146058
## 27 28   3.468014
## 28 29   3.875961
## 29 30   4.384904
## 30 31   4.989569
## 31 32   5.660000
## 32 33   6.347225
## 33 34   6.993501
## 34 35   7.582228
## 35 36   8.218605
## 36 37   9.661143
## 37 38  15.628178
## 38 39  40.589758
## 39 40 113.069328
```