Homework 5 Smoothing

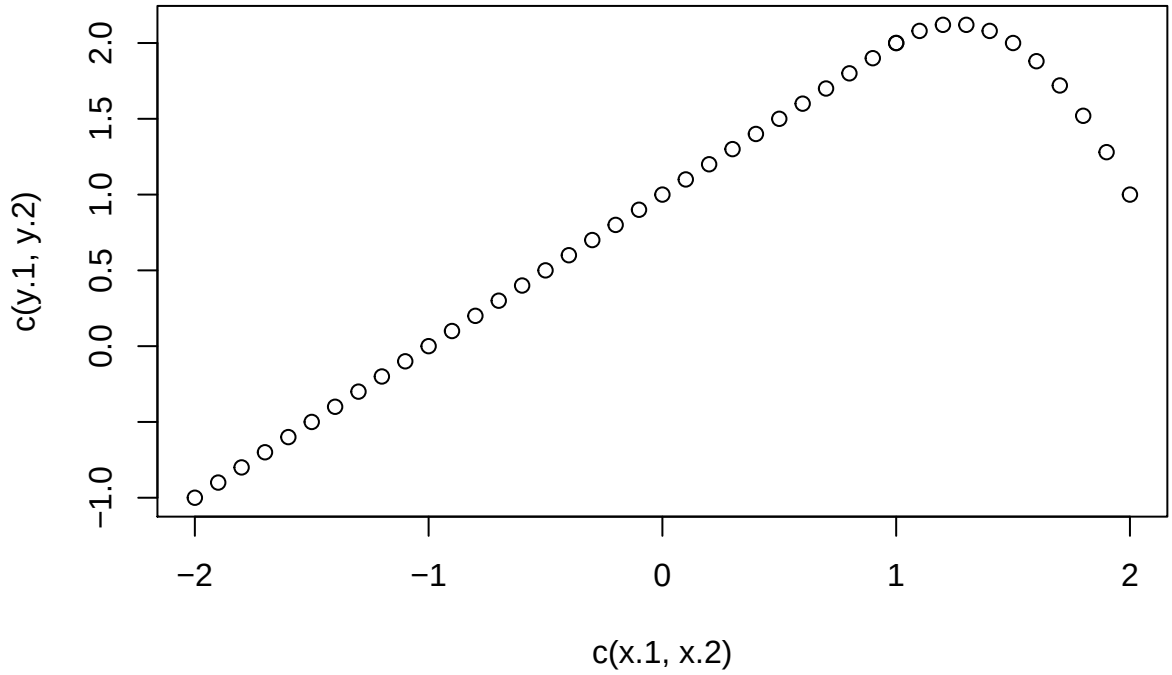
Jinfei Xue

March 1, 2019

7.3, 7.9, 7.10 and 7.11.

7.3

```
x.1 <- seq(-2,1,0.1)  # X<1
x.2 <- seq(1,2,0.1)   # X>=1
y.1 <- 1 + x.1
y.2 <- 1 + x.2 - 2*(x.2-1)^2
plot(c(x.1,x.2),c(y.1,y.2))
```



7.9

a

```
require(gam)
```

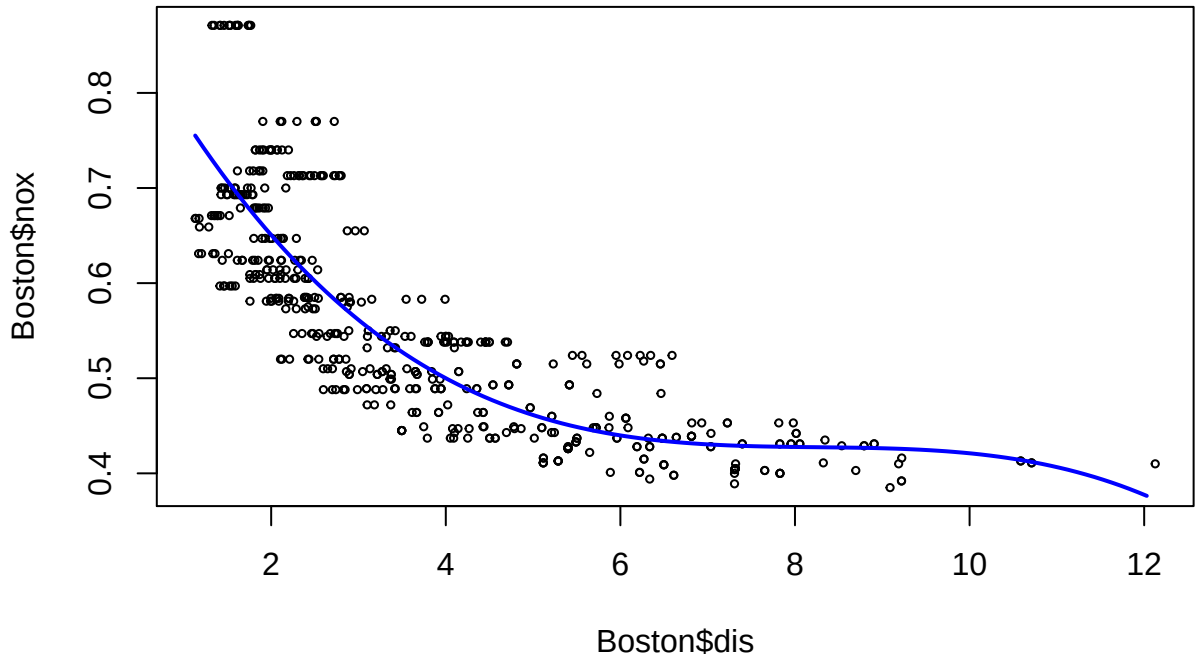
```
## Loading required package: gam
## Loading required package: splines
## Loading required package: foreach
```

```
## Loaded gam 1.16
```

```
require(MASS)
```

```
## Loading required package: MASS
```

```
data(Boston)
set.seed(1)
fit.poly <- lm(nox~poly(dis,3), data=Boston)
dislims <- range(Boston$dis)
dis.grid <- seq(dislims[1], dislims[2], 0.1)
preds <- predict(fit.poly, newdata=list(dis=dis.grid), se=TRUE)
se.bands <- preds$fit + cbind(2*preds$se.fit, -2*preds$se.fit)
par(mfrow=c(1,1), mar=c(4.5,4.5,1,1), oma=c(0,0,4,0))
plot(Boston$dis, Boston$nox, xlim=dislims, cex=0.5)
lines(dis.grid, preds$fit, lwd=2, col="blue")
```




```
summary(fit.poly)
```

```
##
## 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
```

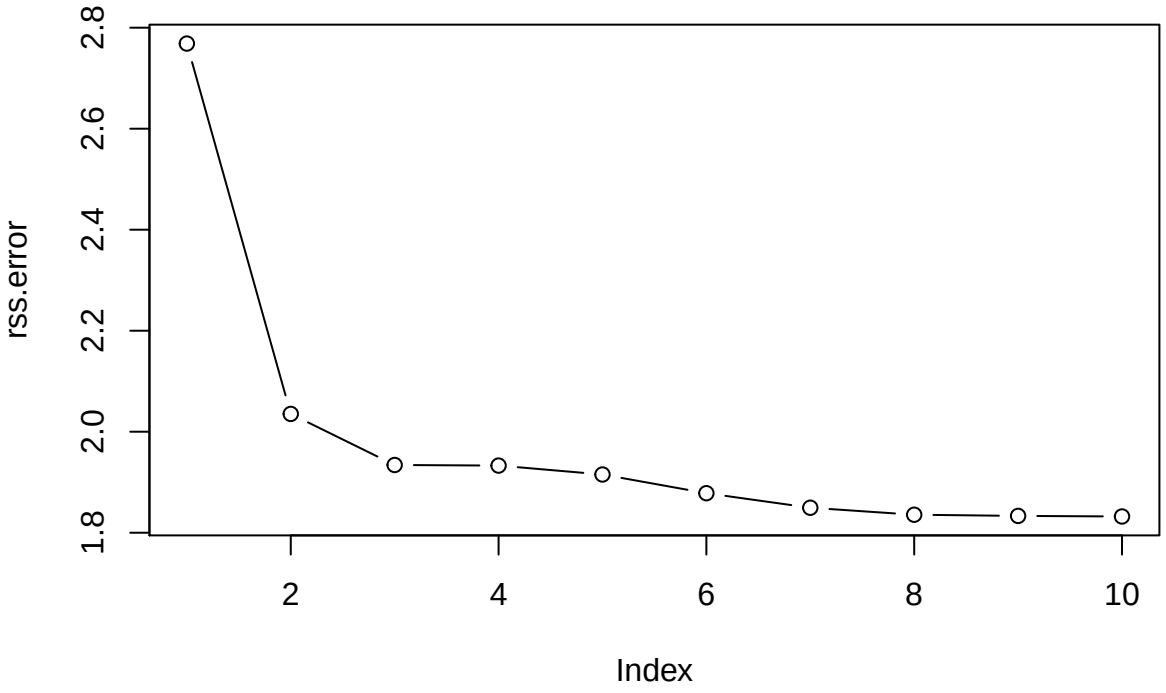
```
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

b

```
rss.error <- rep(0,10)
for (i in 1:10) {
  lm.fit <- lm(nox~poly(dis,i), data=Boston)
  rss.error[i] <- sum(lm.fit$residuals^2)
}
rss.error
```

```
##  [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484
##  [8] 1.835630 1.833331 1.832171
```

```
plot(rss.error, type="b")
```



c

```
require(boot)
```

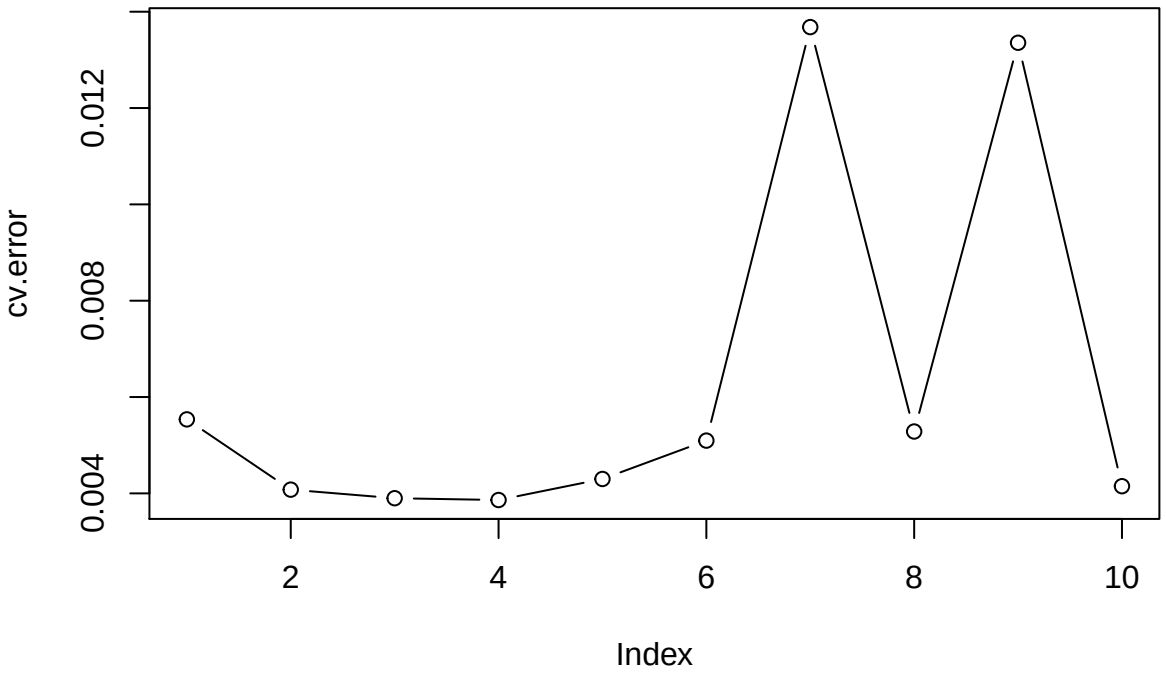
```
## Loading required package: boot
```

```
set.seed(1)
cv.error <- rep(0,10)
for (i in 1:10) {
  glm.fit <- glm(nox~poly(dis,i), data=Boston)
  cv.error[i] <- cv.glm(Boston, glm.fit, K=10)$delta[1]  #use 10 fold cv
```

```
}
cv.error
```

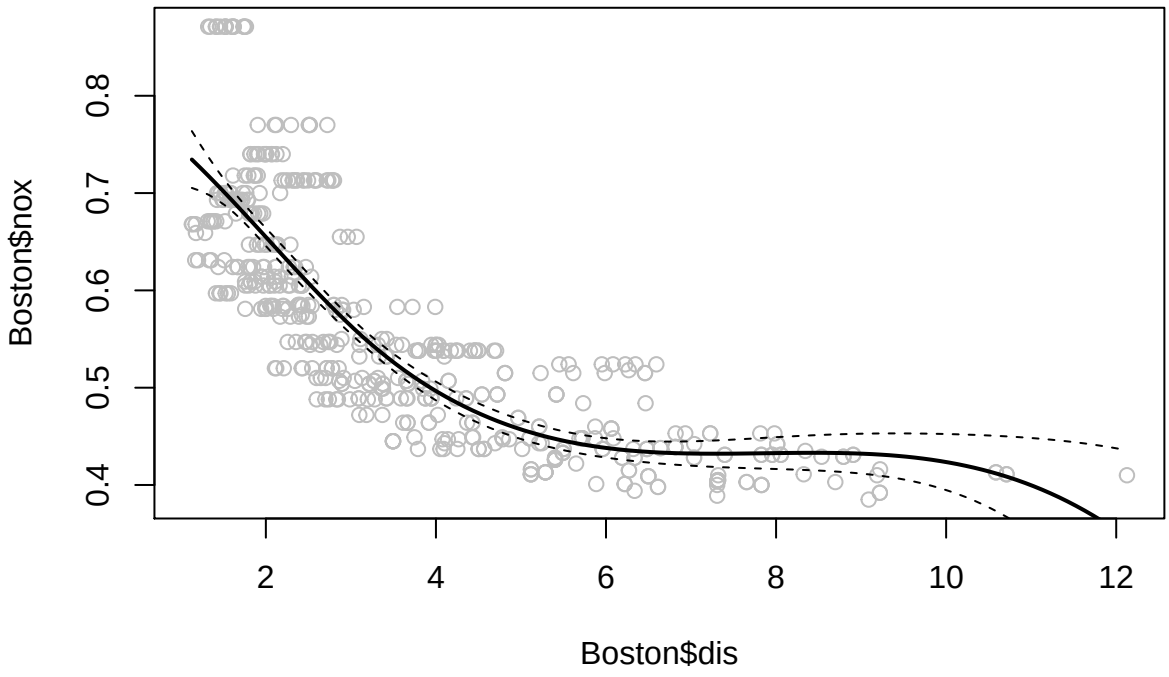
```
##  [1] 0.005536329 0.004077147 0.003899587 0.003862127 0.004298590
##  [6] 0.005095283 0.013680327 0.005284520 0.013355413 0.004148996
```

```
plot(cv.error, type="b")
```



d

```
require(splines)
fit.sp <- lm(nox~bs(dis, df=4), data=Boston)
pred <- predict(fit.sp, newdata=list(dis=dis.grid), se=T)
plot(Boston$dis, Boston$nox, col="gray")
lines(dis.grid, pred$fit, lwd=2)
lines(dis.grid, pred$fit+2*pred$se, lty="dashed")
lines(dis.grid, pred$fit-2*pred$se, lty="dashed")
```




```
# set df to select knots at uniform quantiles of `dis`
attr(bs(Boston$dis,df=4),"knots")  # only 1 knot at 50th percentile
```

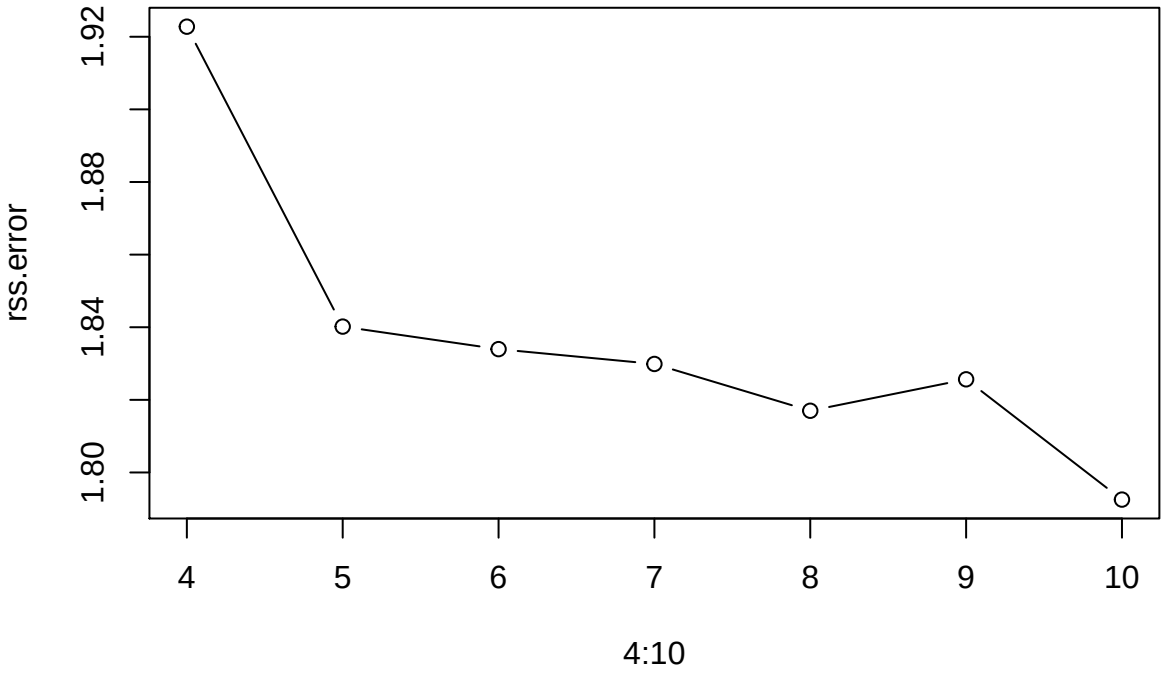
```
##     50%
## 3.20745
```

e

```
require(splines)
set.seed(1)
rss.error <- rep(0,7)
for (i in 4:10) {
  fit.sp <- lm(nox~bs(dis, df=i), data=Boston)
  rss.error[i-3] <- sum(fit.sp$residuals^2)
}
rss.error
```

```
## [1] 1.922775 1.840173 1.833966 1.829884 1.816995 1.825653 1.792535
```

```
plot(4:10, rss.error, type="b")  # RSS decreases on train set w more flexible fit
```



f

```
require(splines)
require(boot)
set.seed(1)
cv.error <- rep(0,7)
for (i in 4:10) {
  glm.fit <- glm(nox~bs(dis, df=i), data=Boston)
  cv.error[i-3] <- cv.glm(Boston, glm.fit, K=10)$delta[1]
}
```

```
## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.1523), 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%` = 3.1523), 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%` = 3.1992), 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%` = 3.1992), Boundary.knots =
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.38876666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.38876666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
```

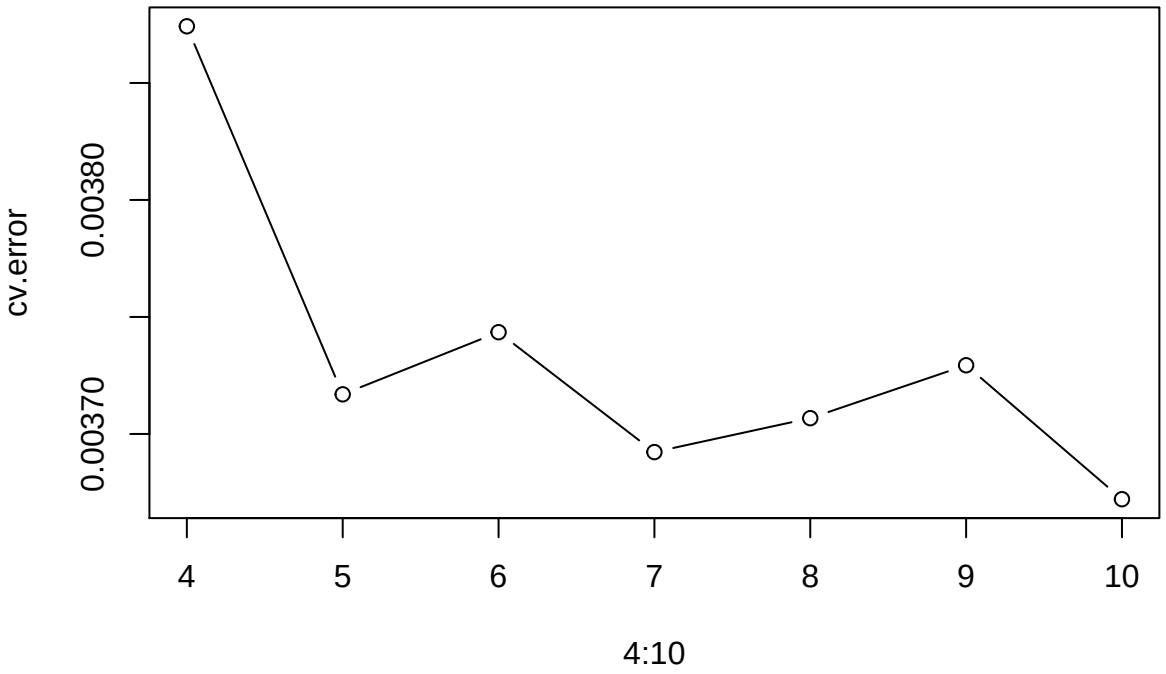
```
## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.34846666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.34846666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.0835, `50%` = 3.1323, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.0835, `50%` = 3.1323, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1084, `50%` = 3.2157, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1084, `50%` = 3.2157, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.9865, `40%` = 2.7147, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.9865, `40%` = 2.7147, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.92938, `40%` =
## 2.59774, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.92938, `40%` =
## 2.59774, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.86778333333333, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.86778333333333, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.83066666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.83066666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.79737142857143, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.79737142857143, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.7984, `28.57143%`
## = 2.206, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.7984, `28.57143%`
## = 2.206, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.755975, `25%` =
## 2.10675, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.755975, `25%` =
## 2.10675, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.7519375, `25%` =
## 2.08585, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.7519375, `25%` =
## 2.08585, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
cv.error
```

```
## [1] 0.003874210 0.003716933 0.003743481 0.003692253 0.003706746 0.003729357
## [7] 0.003672114
```

```
plot(4:10, cv.error, type="b")  # should use at least df=5
```

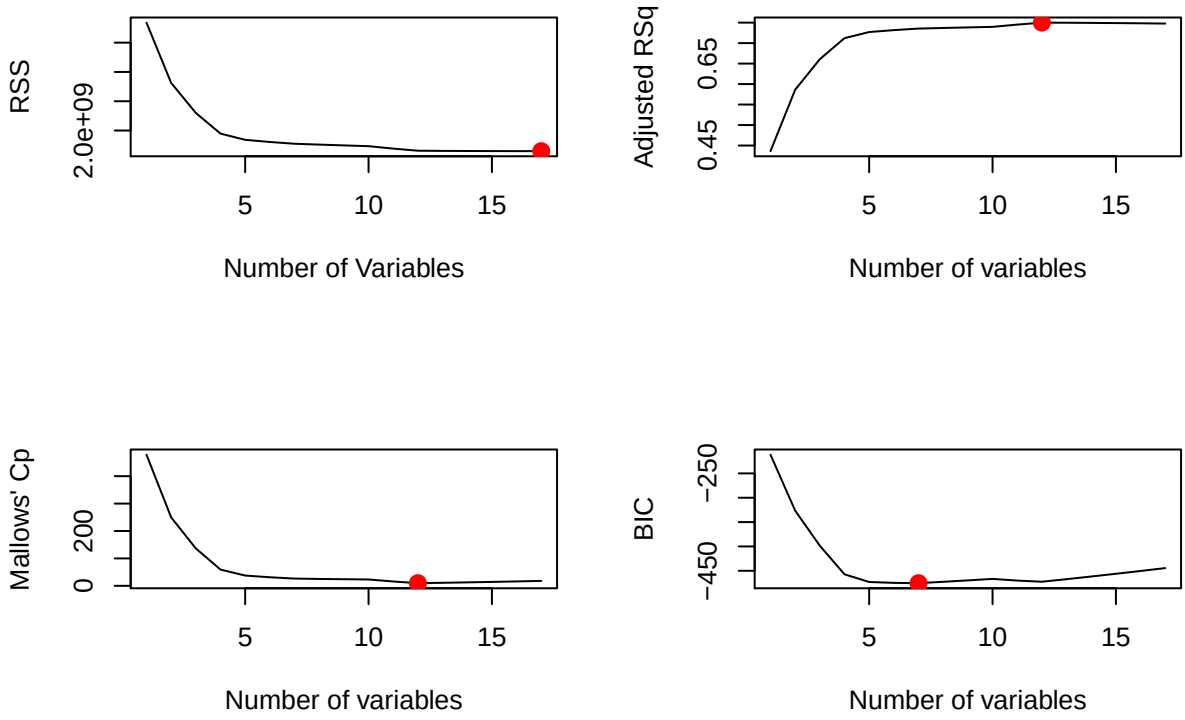


7.10

a

```
require(ISLR)
require(leaps)
data(College)
train = sample(nrow(College), nrow(College) / 2)
test = -train
```

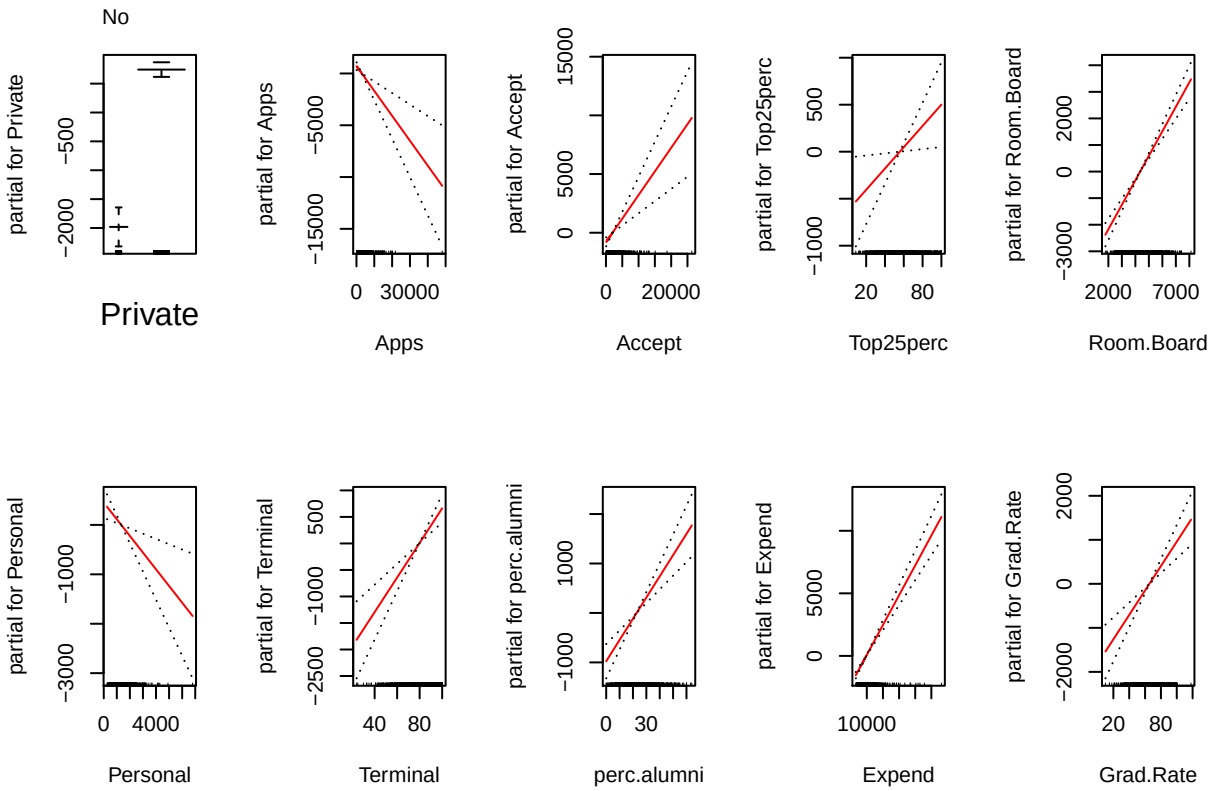
```
train.X = College[train, ]
test.X = College[test, ]
test.Y = College$Outstate[test]

regfit.fwd = regsubsets(Outstate~., data = train.X, nvmax = ncol(College) - 1, method = "forward")
regfit.fwd_summary = summary(regfit.fwd)
par(mfrow = c(2,2))
plot(regfit.fwd_summary$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
points(which.min(regfit.fwd_summary$rss), min(regfit.fwd_summary$rss), col = "red", cex = 2, pch = 20)
plot(regfit.fwd_summary$adjr2, xlab = "Number of variables", ylab = "Adjusted RSq", type = "l")
points(which.max(regfit.fwd_summary$adjr2), max(regfit.fwd_summary$adjr2), col = "red", cex = 2, pch = 2
plot(regfit.fwd_summary$cp, xlab = "Number of variables", ylab = "Mallows' Cp", type = "l")
points(which.min(regfit.fwd_summary$cp), min(regfit.fwd_summary$cp), col = "red", cex = 2, pch = 20)
plot(regfit.fwd_summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(regfit.fwd_summary$bic), min(regfit.fwd_summary$bic), col = "red", cex = 2, pch = 20)
```




```
par(mfrow = c(1,1))
```

b

```
library(gam)
par(mfrow = c(2,5))
gam.m = gam(Outstate ~ Private + Apps + Accept + Top25perc + Room.Board + Personal + Terminal + perc.alu
plot.Gam(gam.m, se = T, col = "red")
```




```
par(mfrow = c(1,1))
```

c

```
preds_all_linear = predict(gam.m, newdata = test.X)
mean((preds_all_linear - test.Y)^2)
```

```
## [1] 3256548
```

d

```
par(mfrow = c(2,5))
gam.full = gam(Outstate ~ Private + s(Apps, 4) + s(Accept, 4) + s(Top25perc, 4) + s(Room.Board, 4) + s(
summary(gam.full)
```

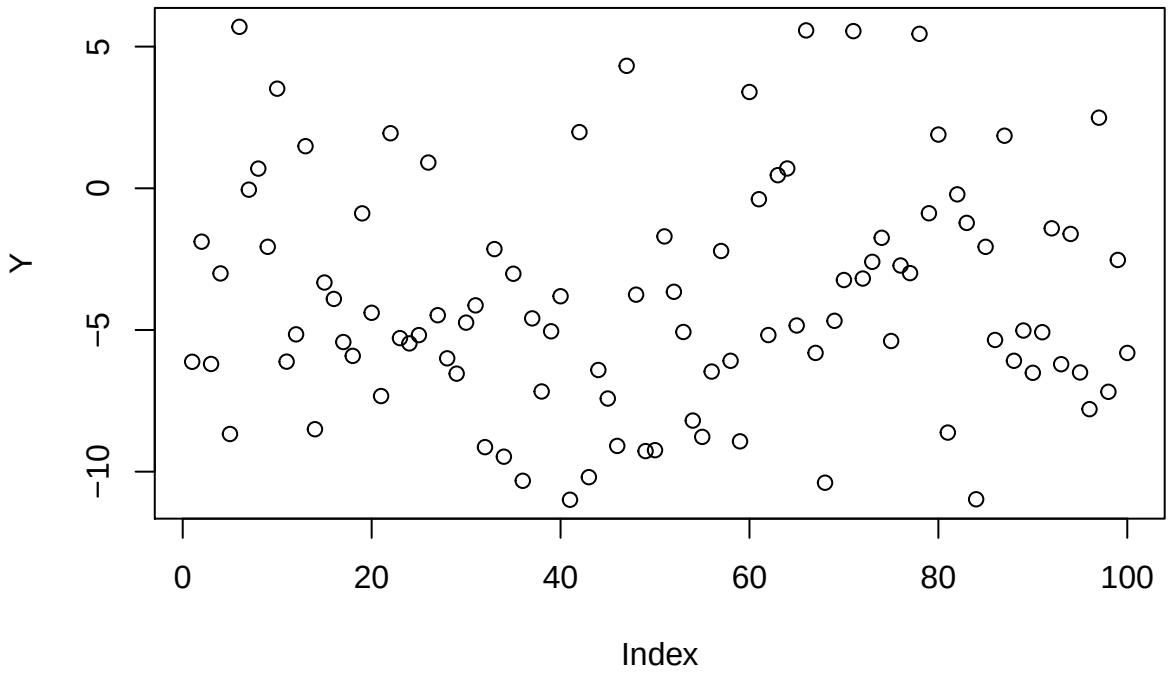
```
##
## Call: gam(formula = Outstate ~ Private + s(Apps, 4) + s(Accept, 4) +
##     s(Top25perc, 4) + s(Room.Board, 4) + s(Personal, 4) + s(Terminal,
##     4) + s(perc.alumni, 4) + s(Expend, 4) + s(Grad.Rate, 4),
##     data = College)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -7298.31 -1068.99    45.18  1255.38  7599.90
##
## (Dispersion Parameter for gaussian family taken to be 3299772)
##
```

```
##     Null Deviance: 12559297426 on 776 degrees of freedom
## Residual Deviance: 2438530706 on 738.9997 degrees of freedom
## AIC: 13906.35
##
## Number of Local Scoring Iterations: 3
##
## Anova for Parametric Effects
##                    Df     Sum Sq    Mean Sq  F value    Pr(>F)
## Private             1 3592090322 3592090322 1088.587 < 2.2e-16 ***
## s(Apps, 4)          1 1162147923 1162147923  352.190 < 2.2e-16 ***
## s(Accept, 4)        1  164618486  164618486   49.888 3.764e-12 ***
## s(Top25perc, 4)     1  815631920  815631920  247.178 < 2.2e-16 ***
## s(Room.Board, 4)    1 1110585697 1110585697  336.564 < 2.2e-16 ***
## s(Personal, 4)      1   76585072   76585072   23.209 1.764e-06 ***
## s(Terminal, 4)      1  205762437  205762437   62.357 1.037e-14 ***
## s(perc.alumni, 4)   1  303347973  303347973   91.930 < 2.2e-16 ***
## s(Expend, 4)        1  880501100  880501100  266.837 < 2.2e-16 ***
## s(Grad.Rate, 4)     1  101531678  101531678   30.769 4.049e-08 ***
## Residuals         739 2438530706    3299772
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                   Npar Df Npar F     Pr(F)
## (Intercept)
## Private
## s(Apps, 4)              3  2.287  0.077330 .
## s(Accept, 4)            3  5.444  0.001049 **
## s(Top25perc, 4)         3  0.556  0.644460
## s(Room.Board, 4)        3  2.359  0.070389 .
## s(Personal, 4)          3  3.527  0.014677 *
## s(Terminal, 4)          3  1.671  0.171916
## s(perc.alumni, 4)       3  1.213  0.303963
## s(Expend, 4)            3 38.879 < 2.2e-16 ***
## s(Grad.Rate, 4)         3  3.107  0.025911 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

7.11

a

```
set.seed(1)
X1 <- rnorm(100)
X2 <- rnorm(100)
beta_0 <- -3.8
beta_1 <- 0.3
beta_2 <- 4.1
eps <- rnorm(100, sd = 1)
Y <- beta_0 + beta_1*X1 + beta_2*X2 + eps
par(mfrow=c(1,1))
plot(Y)
```



b

```
# initialize beta hat 1
bhat_1 <- 1
```

c

```
a <- Y - bhat_1*X1
(bhat_2 <- lm(a~X2)$coef[2])
```

```
##       X2
## 4.047166
```

d

```
a <- Y - bhat_2*X2
(bhat_1 <- lm(a~X1)$coef[2])
```

```
##        X1
## 0.3211108
```

e

```
bhat_0 <- bhat_1 <- bhat_2 <- rep(0, 1000)
for (i in 1:1000) {
  a <- Y - bhat_1[i] * X1
```

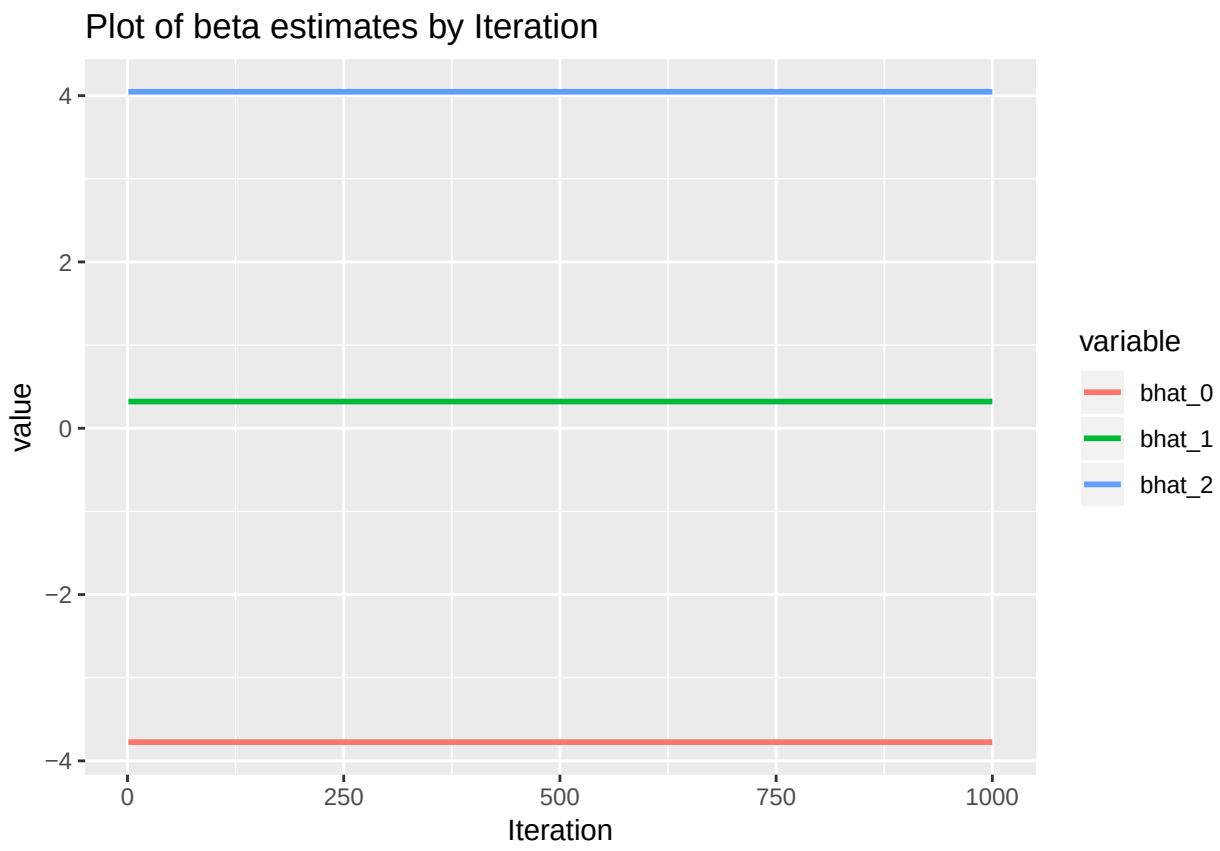
```
  bhat_2[i] <- lm(a ~ X2)$coef[2]
  a <- Y - bhat_2[i] * X2
  bhat_0[i] <- lm(a ~ X1)$coef[1]
  # bhat_1 will end up with 1001 terms
  bhat_1[i+1] <- lm(a ~ X1)$coef[2]
}
# make plots
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(reshape2)
```

```
## Loading required package: reshape2
```

```
mydf <- data.frame(Iteration=1:1000, bhat_0, bhat_1=bhat_1[-1], bhat_2)
mmydf <- melt(mydf, id.vars="Iteration")
ggplot(mmydf, aes(x=Iteration, y=value, group=variable, col=variable)) +
  geom_line(size=1) + ggtitle("Plot of beta estimates by Iteration")
```

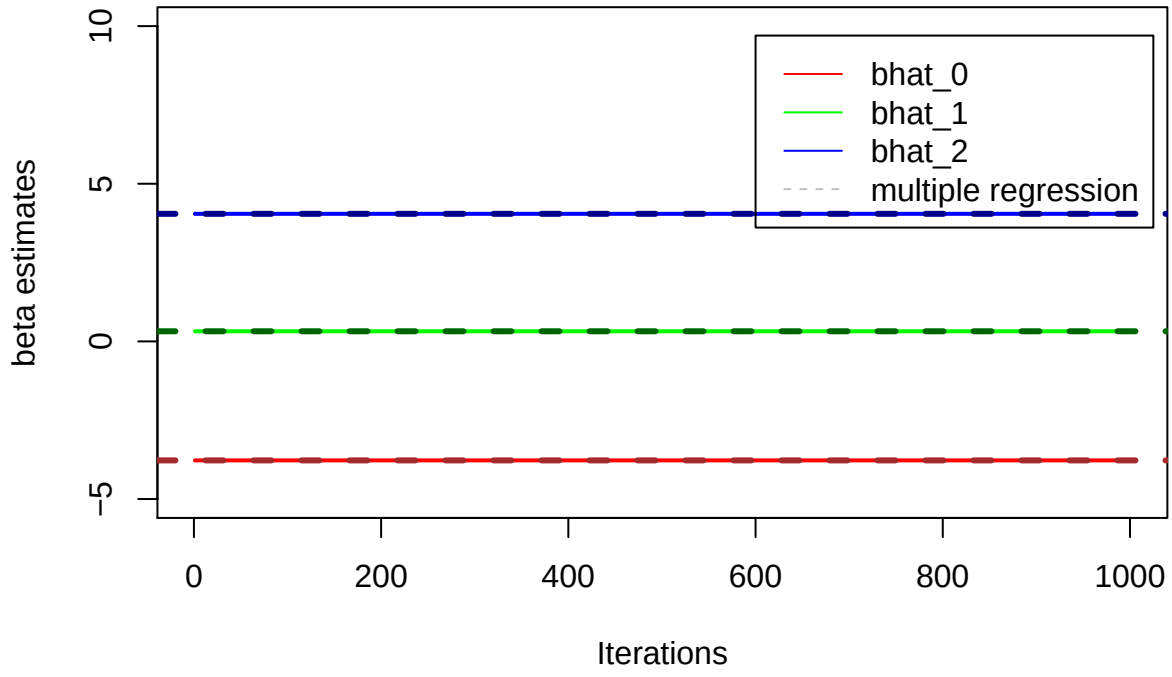


f

```
fit.lm <- lm(Y ~ X1 + X2)
coef(fit.lm)
```

```
## (Intercept)          X1          X2
##  -3.7746466   0.3211102   4.0465332
```

```
plot(bhat_0, type="l", col="red", lwd=2, xlab="Iterations",
     ylab="beta estimates", ylim=c(-5,10))
lines(bhat_1[-1], col="green", lwd=2)
lines(bhat_2, col="blue", lwd=2)
abline(h=coef(fit.lm)[1], lty="dashed", lwd=3, col="brown")
abline(h=coef(fit.lm)[2], lty="dashed", lwd=3, col="darkgreen")
abline(h=coef(fit.lm)[3], lty="dashed", lwd=3, col="darkblue")
legend(x=600,y=9.7, c("bhat_0", "bhat_1", "bhat_2", "multiple regression"),
       lty = c(1,1,1,2),
       col = c("red","green","blue","gray"))
```



g

```
head(mydf)
```

```
##   Iteration    bhat_0    bhat_1   bhat_2
## 1         1 -3.774658 0.3211098 4.046234
## 2         2 -3.774647 0.3211102 4.046533
## 3         3 -3.774647 0.3211102 4.046533
## 4         4 -3.774647 0.3211102 4.046533
## 5         5 -3.774647 0.3211102 4.046533
## 6         6 -3.774647 0.3211102 4.046533
```