Homework 3 Solutions

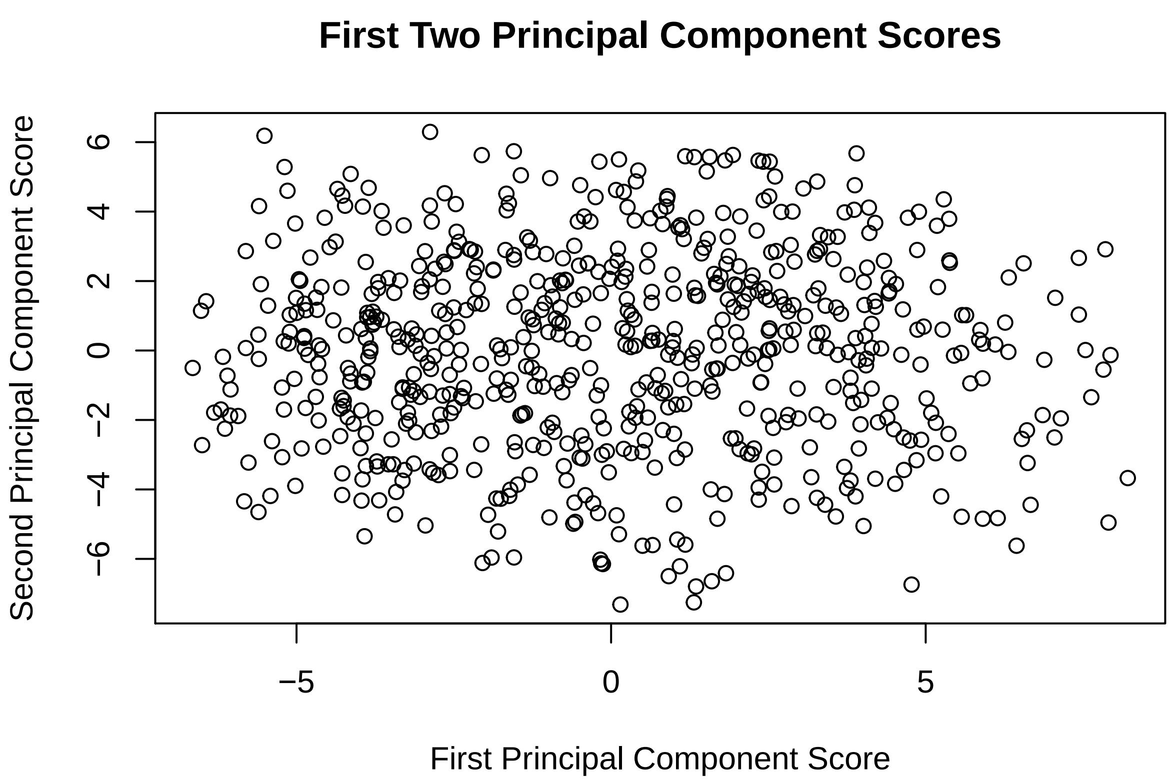
May 20, 2015

Problem 1

```
library(MASS)
library(classifly)
load("threes.Rdata")
```

(a)

```
pc = princomp(threes)
dirs = pc$loadings
score1 = pc$scores[ ,1]
score2 = pc$scores[ ,2]
plot(score2~score1, xlab = "First Principal Component Score",
     ylab = "Second Principal Component Score",
     main = "First Two Principal Component Scores")
```



(b)

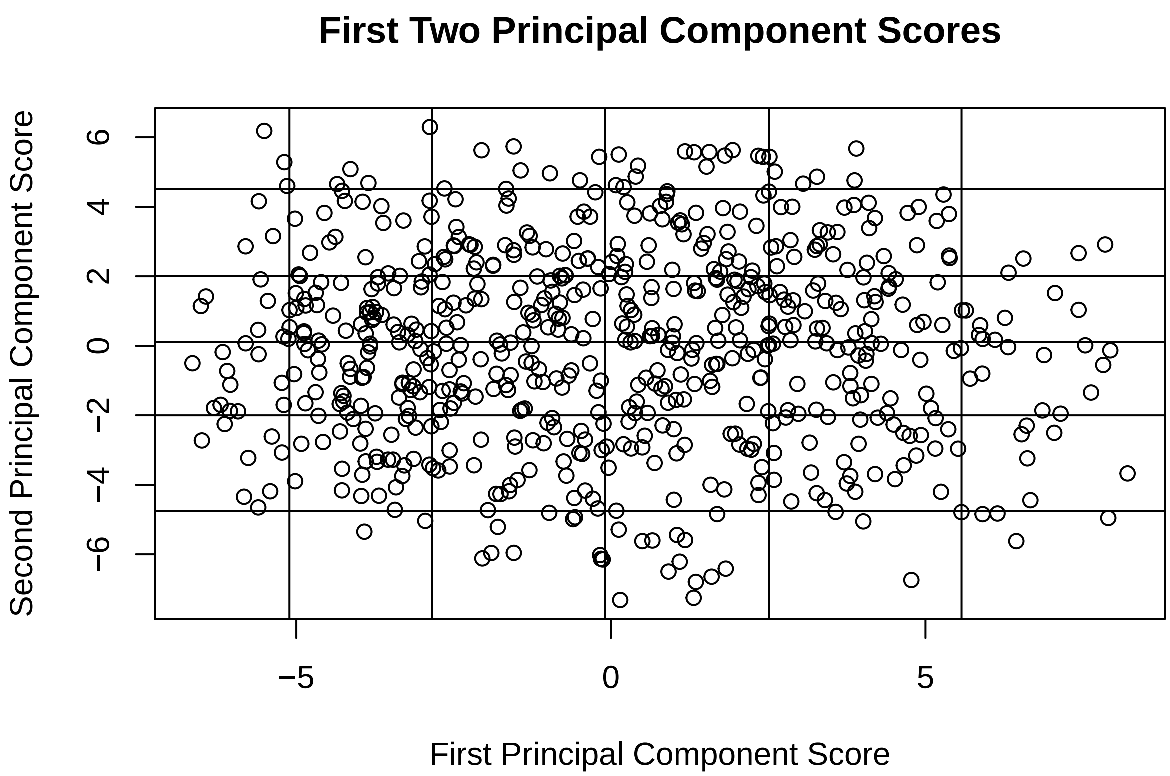
```
qScore1 = quantile(score1,  probs = c(.05, .25, .50, .75, .95))
qScore2 = quantile(score2,  probs = c(.05, .25, .50, .75, .95))
qScore1
```

```
##          5%         25%         50%         75%         95% 
## -5.10972461 -2.84481360 -0.09248278  2.51320157  5.57363992
```

```
qScore2
```

```
##         5%        25%        50%        75%        95% 
## -4.7512623 -2.0006136  0.1125415  2.0140153  4.5164373
```

```
plot(score2~score1, xlab = "First Principal Component Score",
     ylab = "Second Principal Component Score",
     main = "First Two Principal Component Scores")
abline(v = qScore1)
abline(h = qScore2)
```



(c)

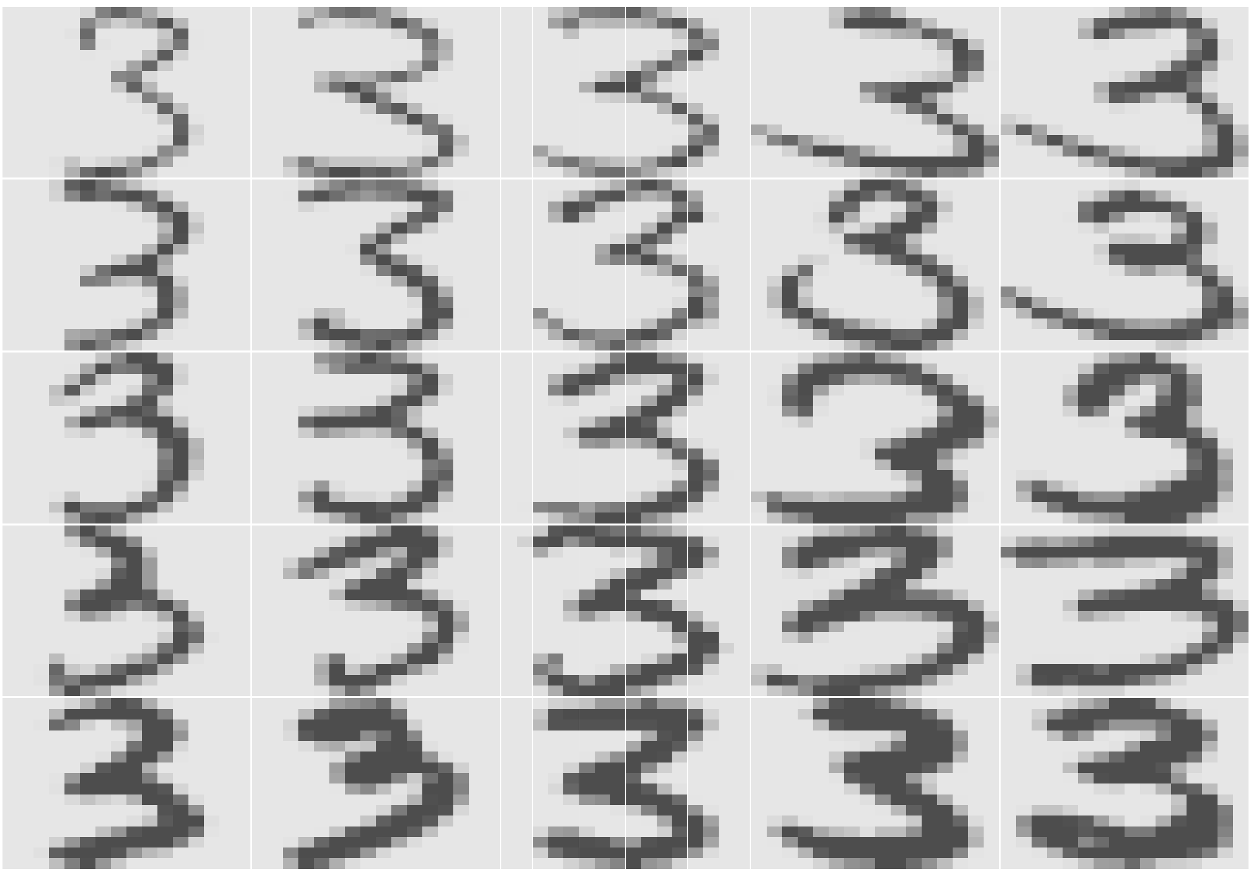
Indeces are listed below the code. Columns indicate first principal component, while rows indicate second principal component.

```
# index = replicate(25, identify(score1, score2, n=1))
load("index.rdata")
matrix(index, nrow=5, byrow=TRUE)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   73  238  568   82  640
## [2,]  284   84  133    4  322
## [3,]  392    6  554  220  500
## [4,]  247  430  142  146  649
## [5,]  184  149  234  375  176
```

(d)

```
plot.digit = function(x, zlim=c(-1,1)) {
  cols = gray.colors(100)[100:1]
  image(matrix(x, nrow=16)[ ,16:1], col=cols,
        zlim=zlim, axes=FALSE)
}
par(mfrow=c(5,5), mex = 0.01)
plot = sapply(index, function(x) plot.digit(threes[x, ]))
```



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

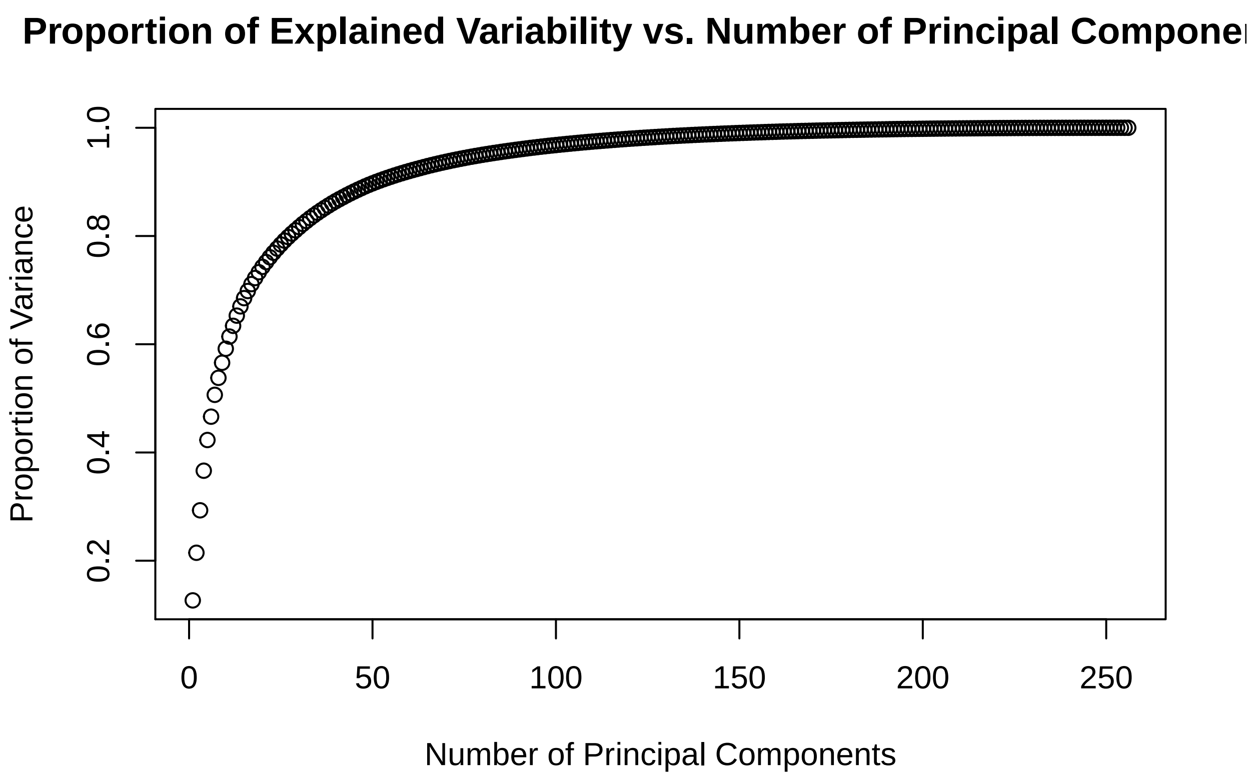
(e)

The previous plot shows that increases in the first principal component (left to right) make the bottom tail of the three a bit longer, and perhaps a bit thicker, as well.

The second principal component (top to bottom) seems to increase the thickness of the three, as well as making the threes more horizontally symmetric (except for a couple cases).

(f)

```
prop.var = pc$sdev^2 / sum(pc$sdev^2)

plot(x = 1:length(prop.var), y=cumsum(prop.var),
     main = "Proportion of Explained Variability vs. Number of Principal Components",
     xlab="Number of Principal Components", ylab="Proportion of Variance")
```




```
min(which(cumsum(prop.var) >= 0.5))
```

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

```
min(which(cumsum(prop.var) >= 0.9))
```

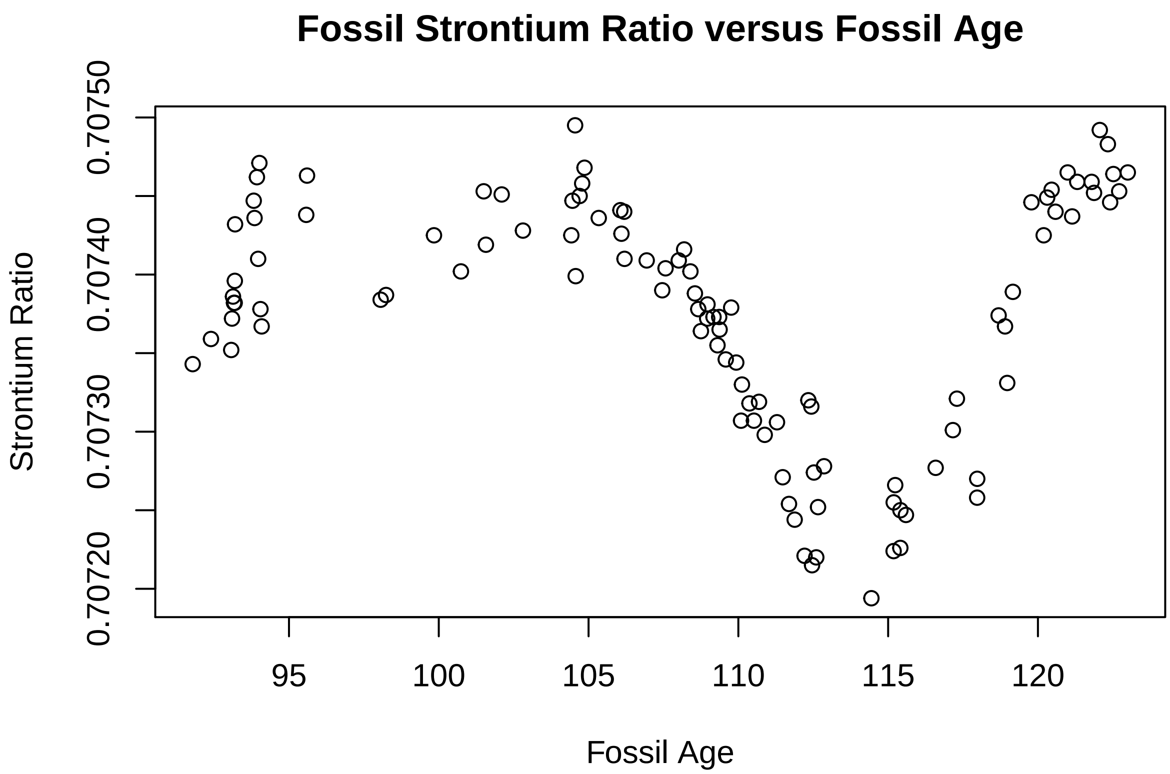
```
## [1] 52
```

We need 7 principal components to explain 50% of the variance, and 52 to explain 90%.

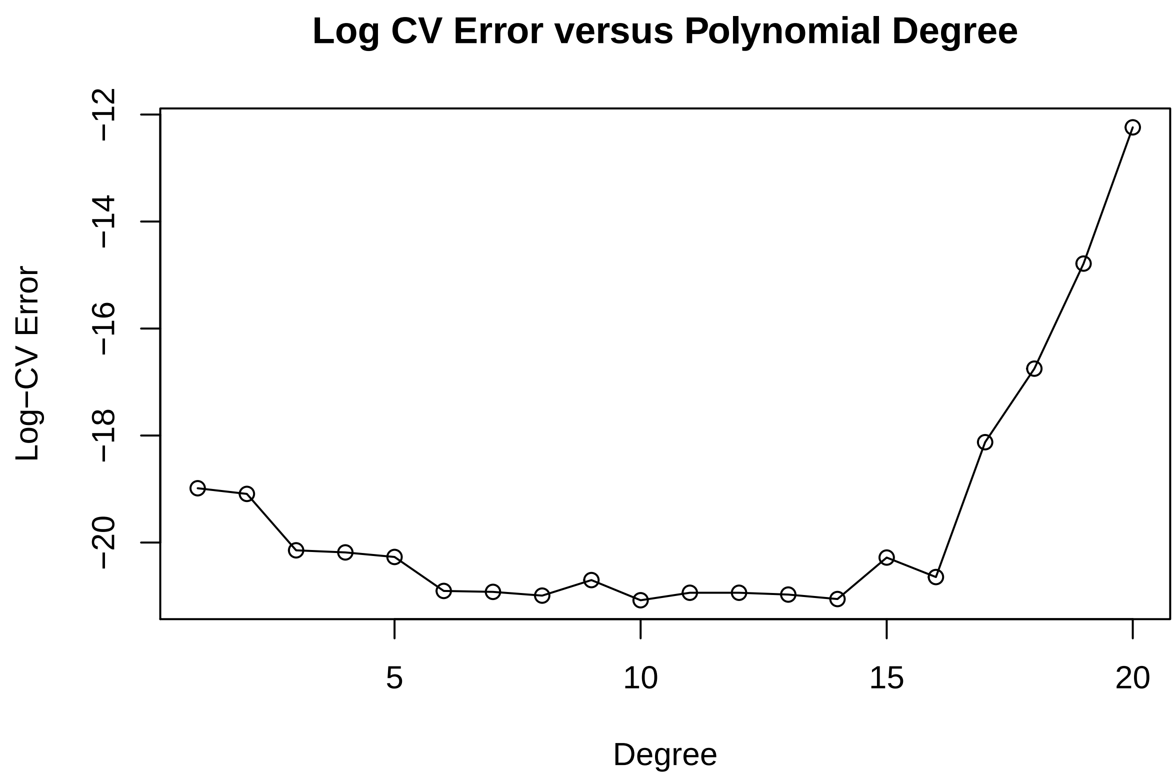
Problem 2

(a)

```
fossil = read.csv("fossil.csv")
plot(strontium.ratio ~ age, data = fossil, main="Fossil Strontium Ratio versus Fossil Age",
     xlab="Fossil Age", ylab="Strontium Ratio")
```




```
library(boot)
cv.error = sapply(1:20, function(i) {
  glmfit = glm(strontium.ratio ~ poly(age, degree=i), data=fossil)
  cv.glm(fossil, glmfit, K=5)$delta[1] } )

plot(log(cv.error), xlab="Degree", type="o", ylab="Log-CV Error",
     main="Log CV Error versus Polynomial Degree")
```




```
degree.opt = which.min(cv.error); degree.opt
```

```
## [1] 10
```

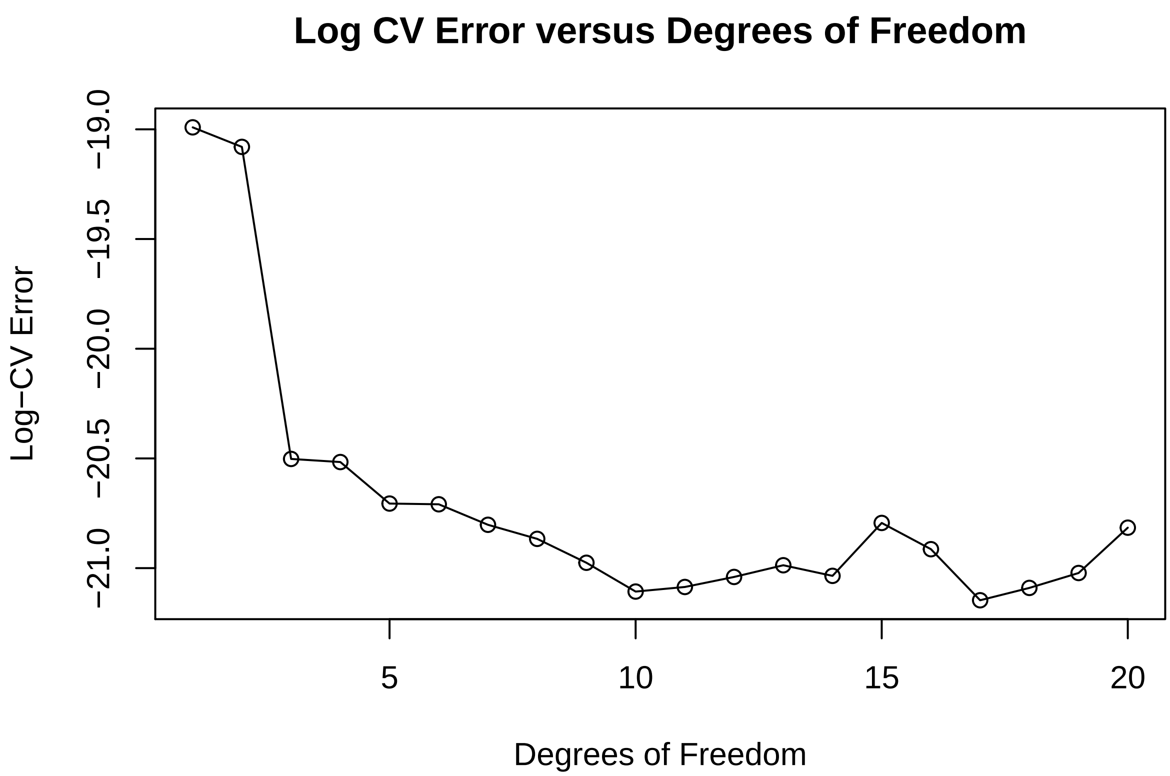
```
fit.poly = glm(strontium.ratio ~ poly(age, degree = degree.opt), data=fossil)
summary(fit.poly)
```

```
##
## Call:
## glm(formula = strontium.ratio ~ poly(age, degree = degree.opt),
##     data = fossil)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.940e-05  -1.192e-05   3.770e-07   1.584e-05   5.585e-05
##
## Coefficients:
##                                    Estimate Std. Error    t value Pr(>|t|)
## (Intercept)                       7.074e-01  2.457e-06 287908.592  < 2e-16
## poly(age, degree = degree.opt)1  -1.334e-04  2.530e-05     -5.275 8.33e-07
## poly(age, degree = degree.opt)2   2.639e-04  2.530e-05     10.433  < 2e-16
## poly(age, degree = degree.opt)3   5.872e-04  2.530e-05     23.213  < 2e-16
## poly(age, degree = degree.opt)4   1.071e-04  2.530e-05      4.233 5.32e-05
## poly(age, degree = degree.opt)5  -1.639e-04  2.530e-05     -6.481 4.03e-09
## poly(age, degree = degree.opt)6  -2.685e-04  2.530e-05    -10.613  < 2e-16
## poly(age, degree = degree.opt)7   1.096e-05  2.530e-05      0.433    0.666
## poly(age, degree = degree.opt)8   4.100e-05  2.530e-05      1.621    0.108
## poly(age, degree = degree.opt)9  -2.259e-05  2.530e-05     -0.893    0.374
```

```
## poly(age, degree = degree.opt)10  3.879e-05  2.530e-05      1.534    0.128
##
## (Intercept)                      ***
## poly(age, degree = degree.opt)1  ***
## poly(age, degree = degree.opt)2  ***
## poly(age, degree = degree.opt)3  ***
## poly(age, degree = degree.opt)4  ***
## poly(age, degree = degree.opt)5  ***
## poly(age, degree = degree.opt)6  ***
## poly(age, degree = degree.opt)7
## poly(age, degree = degree.opt)8
## poly(age, degree = degree.opt)9
## poly(age, degree = degree.opt)10
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.398747e-10)
##
##     Null deviance: 6.0729e-07  on 105  degrees of freedom
## Residual deviance: 6.0788e-08  on  95  degrees of freedom
## AIC: -1930.8
##
## Number of Fisher Scoring iterations: 2
```

(b)

```
library(splines)
cv.error = sapply(1:20, function(i) {
  glmfit = glm(strontium.ratio ~ ns(age, df=i), data=fossil)
  cv.glm(fossil, glmfit, K=5)$delta[1] } )

plot(log(cv.error), xlab="Degrees of Freedom", type="o", ylab="Log-CV Error",
     main="Log CV Error versus Degrees of Freedom")
```




```
df.opt = which.min(cv.error); df.opt
```

```
## [1] 17
```

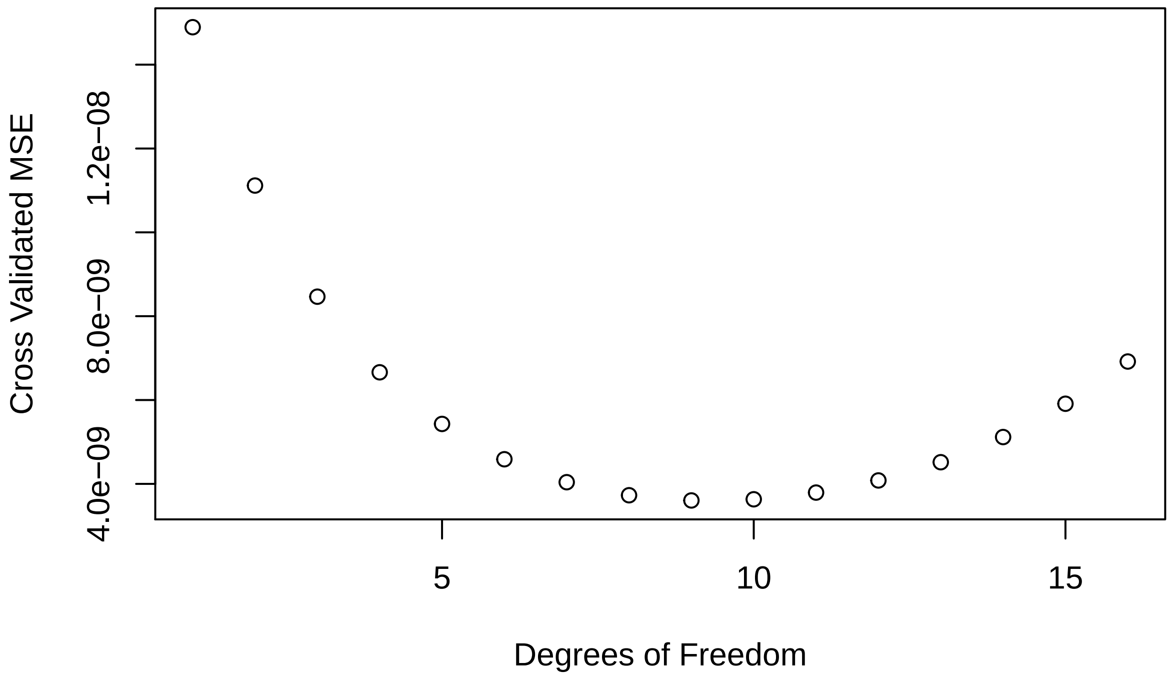
```
fit.ns = glm(strontium.ratio ~ ns(age, df=df.opt), data=fossil)
summary(fit.ns)
```

```
##
## Call:
## glm(formula = strontium.ratio ~ ns(age, df = df.opt), data = fossil)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -6.251e-05  -1.302e-05  -2.001e-06   1.428e-05   5.713e-05
##
## Coefficients:
##                            Estimate Std. Error   t value Pr(>|t|)
## (Intercept)               7.073e-01  2.281e-05 31012.260  < 2e-16 ***
## ns(age, df = df.opt)1     1.514e-04  3.127e-05     4.842 5.46e-06 ***
## ns(age, df = df.opt)2    -5.856e-06  4.433e-05    -0.132 0.895191
## ns(age, df = df.opt)3     1.151e-04  3.366e-05     3.419 0.000954 ***
## ns(age, df = df.opt)4     1.129e-04  2.847e-05     3.966 0.000148 ***
## ns(age, df = df.opt)5     8.026e-05  3.194e-05     2.513 0.013809 *
## ns(age, df = df.opt)6     6.490e-05  3.247e-05     1.999 0.048695 *
## ns(age, df = df.opt)7     2.844e-05  2.786e-05     1.021 0.310272
## ns(age, df = df.opt)8    -3.412e-06  3.057e-05    -0.112 0.911382
## ns(age, df = df.opt)9    -7.291e-05  3.260e-05    -2.236 0.027874 *
## ns(age, df = df.opt)10   -7.003e-05  3.120e-05    -2.245 0.027303 *
```

```
## ns(age, df = df.opt)11 -1.540e-04  3.902e-05    -3.946 0.000159 ***
## ns(age, df = df.opt)12 -4.175e-05  3.983e-05    -1.048 0.297444
## ns(age, df = df.opt)13 -6.121e-05  3.512e-05    -1.743 0.084865 .
## ns(age, df = df.opt)14  1.384e-04  3.093e-05     4.475 2.28e-05 ***
## ns(age, df = df.opt)15  1.030e-04  2.657e-05     3.877 0.000203 ***
## ns(age, df = df.opt)16  1.531e-04  4.845e-05     3.161 0.002157 **
## ns(age, df = df.opt)17  1.066e-04  2.114e-05     5.040 2.47e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.267257e-10)
##
##     Null deviance: 6.0729e-07  on 105  degrees of freedom
## Residual deviance: 5.5152e-08  on  88  degrees of freedom
## AIC: -1927.1
##
## Number of Fisher Scoring iterations: 2
```

(c)

```r
strontium.ratio = fossil$strontium.ratio
age = fossil$age
cv.smoothspline = function(data,df) {
  data.split = split(data, cut(data$age, 5))

  MSE = sapply(1:5, function(l){
    train = do.call(rbind, data.split[-l])
    test = data.split[[l]]
    fit = smooth.spline(x=train$age, y=train$strontium.ratio, df=df)
    mean((predict(fit, test$age,se=T)$y - test$strontium.ratio)^2)
    })
  mean(MSE)
}
CV = sapply(5:20, function(df) cv.smoothspline(fossil, df))
plot(CV, xlab="Degrees of Freedom", ylab= "Cross Validated MSE")
```




```r
which.min(CV)
```

```
## [1] 9
```

```r
fit.smooth = smooth.spline(age, strontium.ratio, df=which.min(CV))
```

(d)

```r
library(bisoreg)
```

```
## Loading required package: bootstrap
## Loading required package: monreg
## Loading required package: R2WinBUGS
## Loading required package: coda
```

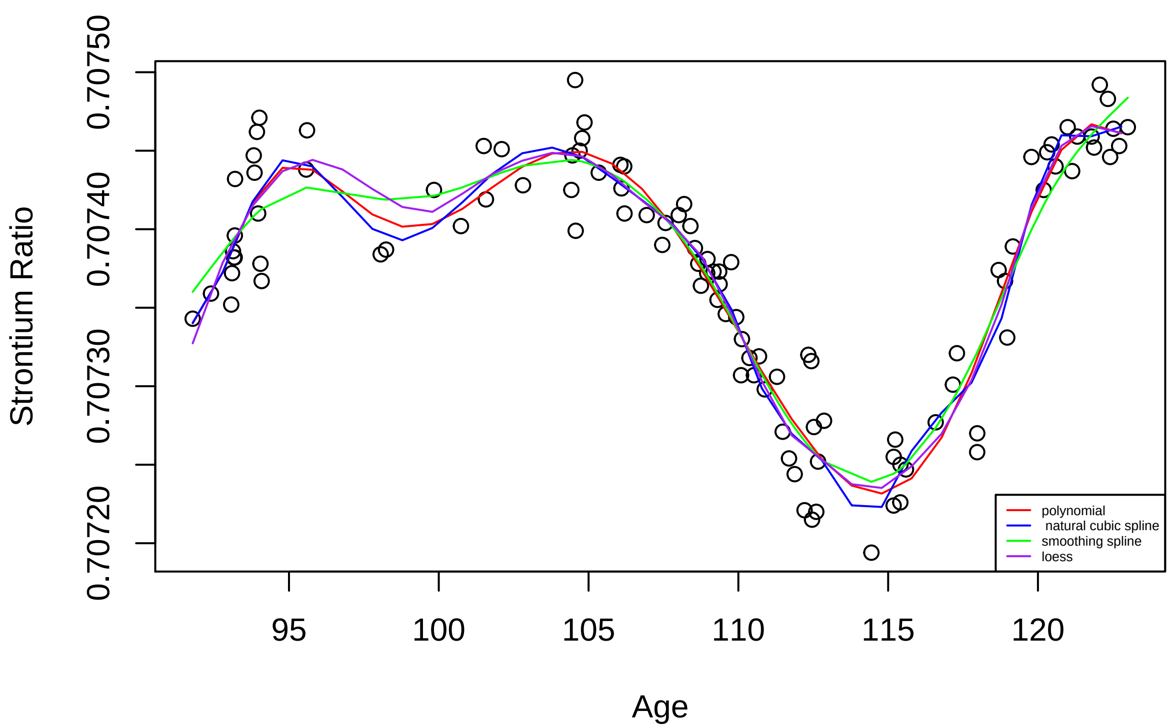
```r
span.opt = summary(loess.wrapper(age, strontium.ratio,
                                 span.vals = seq(.1, 1, by = 0.01), folds = 5))$pars$span

fit.loess=loess(strontium.ratio~age, span = span.opt)
summary(fit.loess)
```

```
## Call:
## loess(formula = strontium.ratio ~ age, span = span.opt)
##
## Number of Observations: 106
## Equivalent Number of Parameters: 13.63
## Residual Standard Error: 2.566e-05
## Trace of smoother matrix: 15.08
```

```
##
## Control settings:
##   normalize:  TRUE
##   span       :  0.21
##   degree    :  2
##   family    :  gaussian
##   surface   :  interpolate      cell = 0.2
```

(e)

```
ageRange=range(age)
ageSeq = seq(from=ageRange[1],to=ageRange[2])
predict.poly = predict(fit.poly, newdata = list(age=ageSeq),se=T)
predict.ns = predict(fit.ns, newdata = list(age=ageSeq),se=T)
predict.smooth = predict(fit.smooth, newdata = data.frame(age=ageSeq),se=T)
predict.loess = predict(fit.loess, newdata = data.frame(age=ageSeq),se=T)

plot(strontium.ratio ~ age, xlab="Age", ylab="Strontium Ratio", data=fossil)
lines(ageSeq, predict.poly$fit, col="red")
lines(ageSeq, predict.ns$fit, col="blue")
lines(predict.smooth, col="green")
lines(ageSeq, predict.loess$fit, col="purple")

legend("bottomright", col=c('red','blue','green','purple'), lty=1, cex=0.4,
       legend=c("polynomial", " natural cubic spline",'smoothing spline','loess'))
```



Problem 3.

(a)

```r
library(gam)
```

```
## Loading required package: foreach
## Loaded gam 1.12
```

```r
income = read.csv('http://www-bcf.usc.edu/~gareth/ISL/Income2.csv')
Income.LM = lm(Income ~ Education + Seniority, data=income)
coefficients(Income.LM)
```

```
## (Intercept)   Education   Seniority
## -50.0856388   5.8955560   0.1728555
```

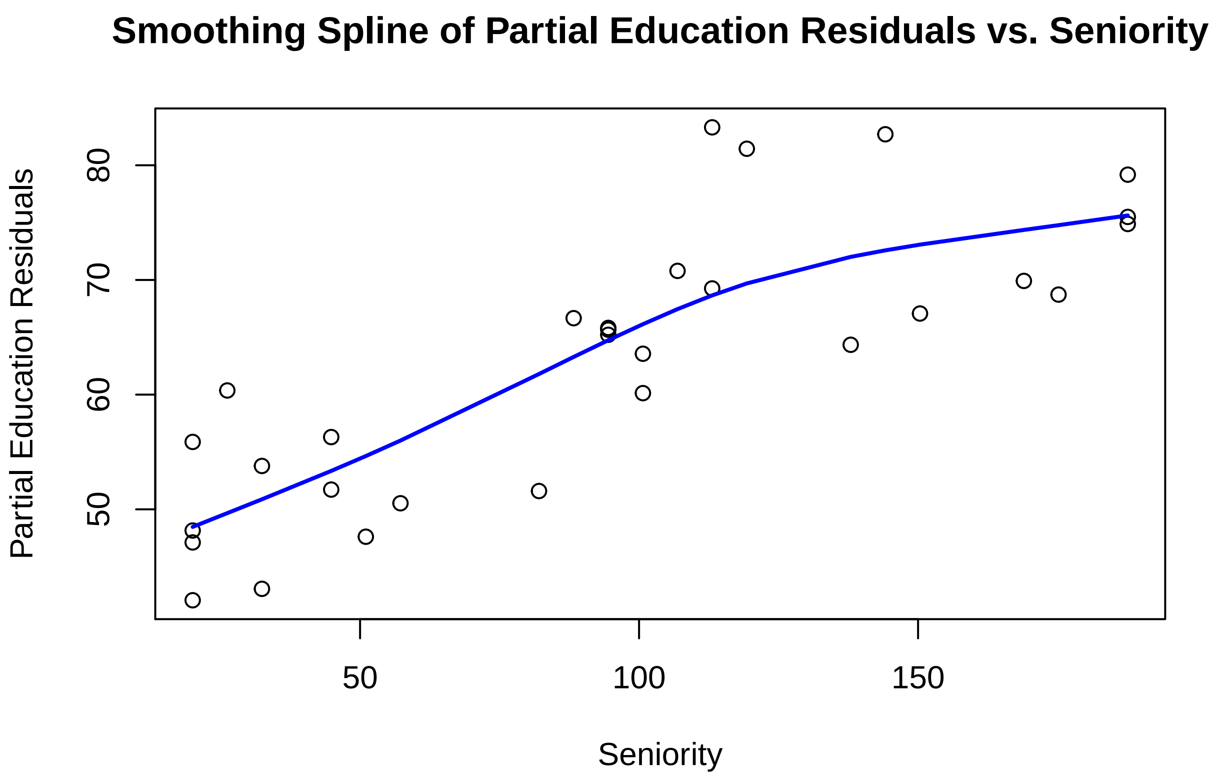
```r
f1 = with(income, Income.LM$coefficient[2]*Education-mean(Income.LM$coefficient[2]*Education))
```

(b)

```r
z1 = income$Income - f1
plot(income$Seniority, z1, xlab = "Seniority", ylab = "Partial Education Residuals",
     main = "Smoothing Spline of Partial Education Residuals vs. Seniority")
s2 = smooth.spline(income$Seniority, z1, cv=TRUE)
```

```
## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful
```

```r
lines(s2, lwd=2, col='blue')
```



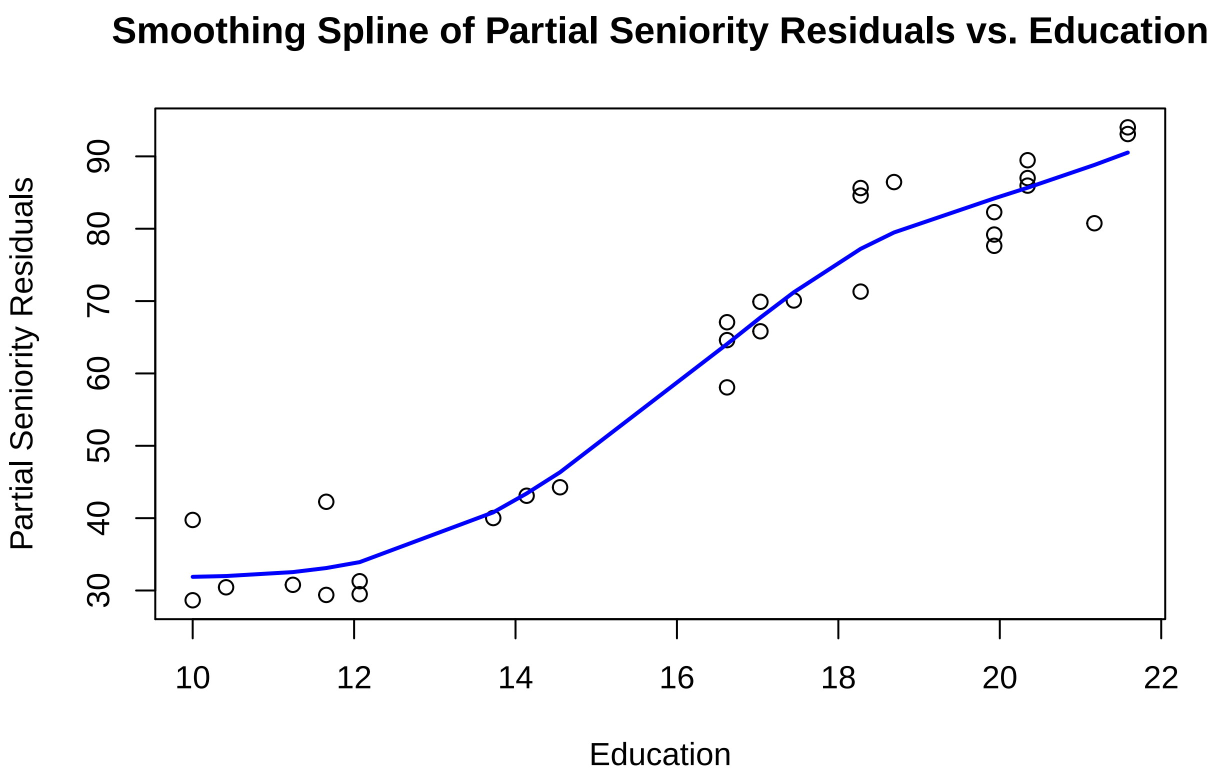

```
f2 = predict(s2,income$Seniority)$y
f2 = f2 - mean(f2)
```

(c)

```
z2 = income$Income - f2
plot(income$Education, z2, xlab = "Education", ylab = "Partial Seniority Residuals",
     main = "Smoothing Spline of Partial Seniority Residuals vs. Education")
s1 = smooth.spline(income$Education, z2, cv=TRUE)
```

```
## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful
```

```
lines(s1, lwd=2, col=' blue')
```



(d)

```
niter = 0
tol = 1E-4
L2dist = Inf

while (L2dist >= tol){
    f1.old = f1
    f2.old = f2

    z1 = income$Income - f1
    s2 = smooth.spline(income$Seniority,z1, cv=TRUE)
    f2 = predict(s2,income$Seniority)$y   - mean(predict(s2,income$Seniority)$y  )

    z2 = income$Income - f2
    s1 = smooth.spline(income$Education,z2, cv=TRUE)
    f1 = predict(s1, income$Education)$y - mean(predict(s1, income$Education)$y)
    L2dist = sqrt(sum(((f1+f2) - (f1.old + f2.old))^2))
    niter = niter + 1
}
```

```
## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
```

```
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Seniority, z1, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful

## Warning in smooth.spline(income$Education, z2, cv = TRUE): cross-validation
## with non-unique 'x' values seems doubtful
```

```r
cat("Convergence reached after ", niter, "iterations (tolerance=", tol, ")")
```

```
## Convergence reached after  10 iterations (tolerance= 1e-04 )
```

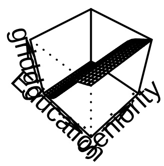
(e)

```r
par(mfrow=c(2,2))
EducationGrid = with(income, seq(min(Education), max(Education), len=20))
SeniorityGrid = with(income, seq(min(Seniority), max(Seniority), len=20))
griddf = with(income, expand.grid(Education=EducationGrid, Seniority=SeniorityGrid))

pred <- function(grid){
    (predict(s1,grid[,1])$y - mean(f1)) + (predict(s2,grid[,2])$y - mean(f2))
}

griddg <- matrix(pred(griddf), 20, 20)

f1_hat = predict(s1, EducationGrid)$y - mean(predict(s1, EducationGrid)$y)
f2_hat = predict(s2, SeniorityGrid)$y - mean(predict(s2, SeniorityGrid)$y)
griddg = matrix(f1_hat + f2_hat, 20, 20)
persp(EducationGrid, SeniorityGrid, griddg, phi=45, theta=45, d=2, main = "GAM", xlab='Education', ylab=

library(fields)
```

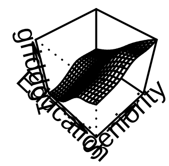
```
## Loading required package: spam
## Loading required package: grid
## Spam version 1.0-1 (2014-09-09) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
##
## Attaching package: 'spam'
##
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
##
## Loading required package: maps
```

```r
fit = Tps(income[,2:3], income$Income)
griddg1 = matrix(predict(fit, griddf), 20, 20)
persp(EducationGrid, SeniorityGrid, griddg1, phi=45, theta=45, d=2, main = "Thin Plate Spline", xlab='Ed

#Loess
loc.fit = loess(Income~Education+Seniority, data=income)
griddg <- matrix(predict(loc.fit, griddf), 20, 20)
persp(EducationGrid, SeniorityGrid, griddg, phi=45, theta=45, d=2, main = "Loess",
      xlab='Education', ylab='Seniority')
```

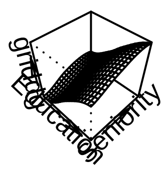
GAM



Thin Plate Spline



Loess



The Backfit GAM is the most smooth fit, while the thin plate has the most amount of bumpiness.