STA207 homework7

Juanjuan Hu, Zhen Zhang

March 8, 2016

14.9

```
## Loading required package: rJava
## Loading required package: xlsxjars
```

(a)

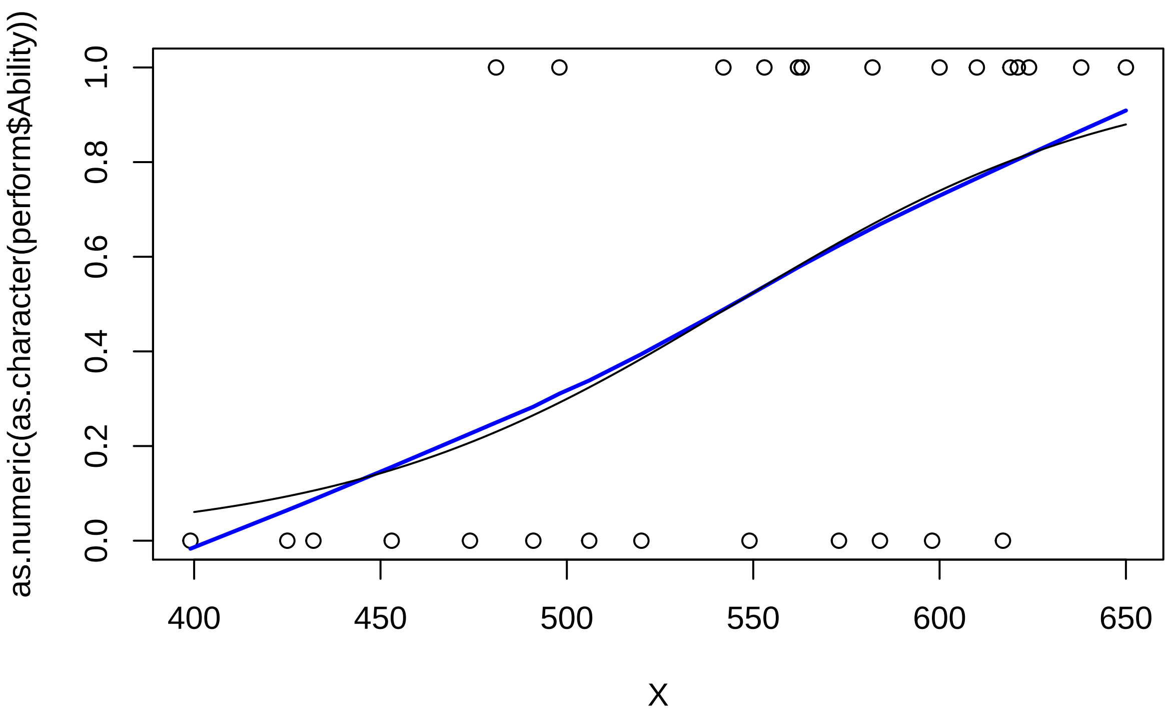
```r
# fit the simple logistic regression
fit1 = glm(Ability~Stability, data = perform, family = binomial)
summary(fit1)
```

```
##
## Call:
## glm(formula = Ability ~ Stability, family = binomial, data = perform)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7845  -0.8350   0.5065   0.8371   1.7145
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.308925   4.376997  -2.355   0.0185 *
## Stability     0.018920   0.007877   2.402   0.0163 *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 37.393  on 26  degrees of freedom
## Residual deviance: 29.242  on 25  degrees of freedom
## AIC: 33.242
##
## Number of Fisher Scoring iterations: 4
```

The MLE of β_0 and β_1 are: -10.31, 0.02. The fitted response function is: $\hat{\pi} = \frac{exp(-10.31+0.02X)}{1+exp(-10.31+0.02X)}$

(b)

```
b0 = -10.308925
b1 = 0.018920
X=perform$Stability
prediction = predict(fit1,type = 'response')
f = function(x) exp(b0+b1*x)/(1+exp(b0+b1*x))
par(mfrow =c(1,1))
plot(X,y = as.numeric(as.character(perform$Ability)))
lines(lowess(X, prediction),col="blue",lwd=2)
curve(f, 400, 650, add=TRUE)
```



It fits well.

(c)

```
exp(b1)
```

```
## [1] 1.0191
```

exp(b1) is 1.0191, which means that the estimated oddes are multiplied by 1.0191 for any unit increase in x.

(d)

```r
f(550)
```

```
## [1] 0.5242497
```

The estimated probability is 0.5242.

(e)

```r
p=0.7
x = (log(p/(1-p))-b0)/b1
x
```

```
## [1] 589.6524
```

The estimated emotional stability test score is 589.6542.

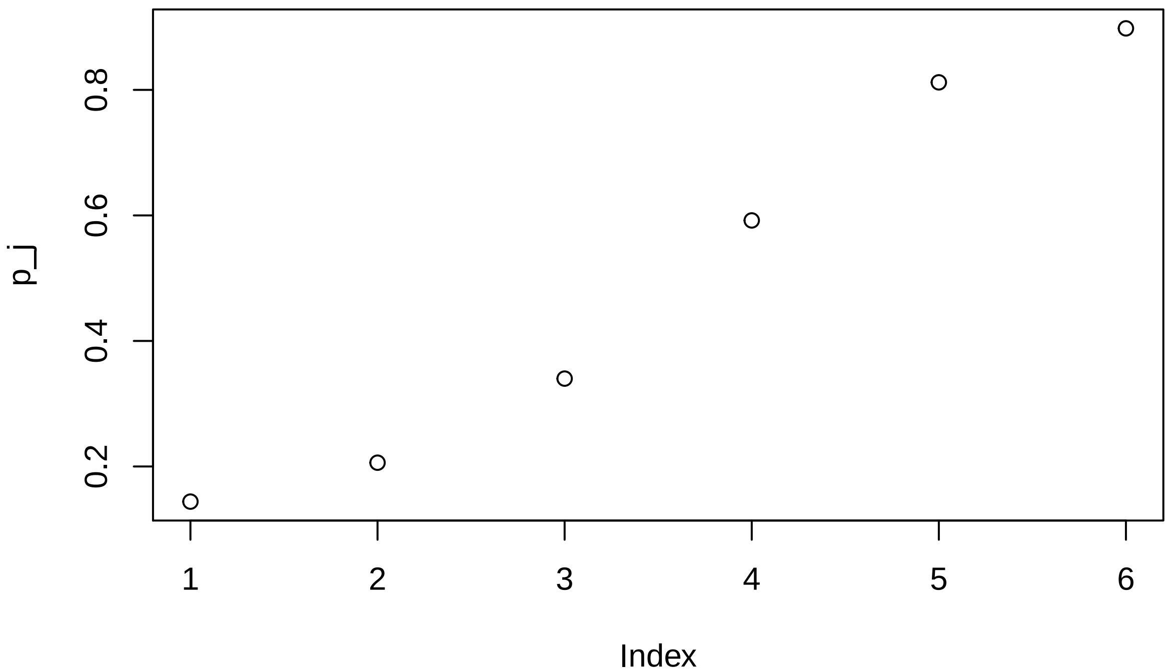
14.11

(a)

```r
x_j = c(2,5,10,20,25,30)
n_j = rep(500, 6)
Y_j = c(72, 103, 170, 296, 406, 449)
p_j = Y_j/n_j
p_j
```

```
## [1] 0.144 0.206 0.340 0.592 0.812 0.898
```

```r
plot(p_j)
```



The plot has a sigmoid shape within a range (0, 1). It suggests that the logistic response function is appropriate.

(b)

```r
fit2 = glm(p_j~x_j, family = binomial)
```

```
## Warning: non-integer #successes in a binomial glm!
```

```r
summary(fit2)
```

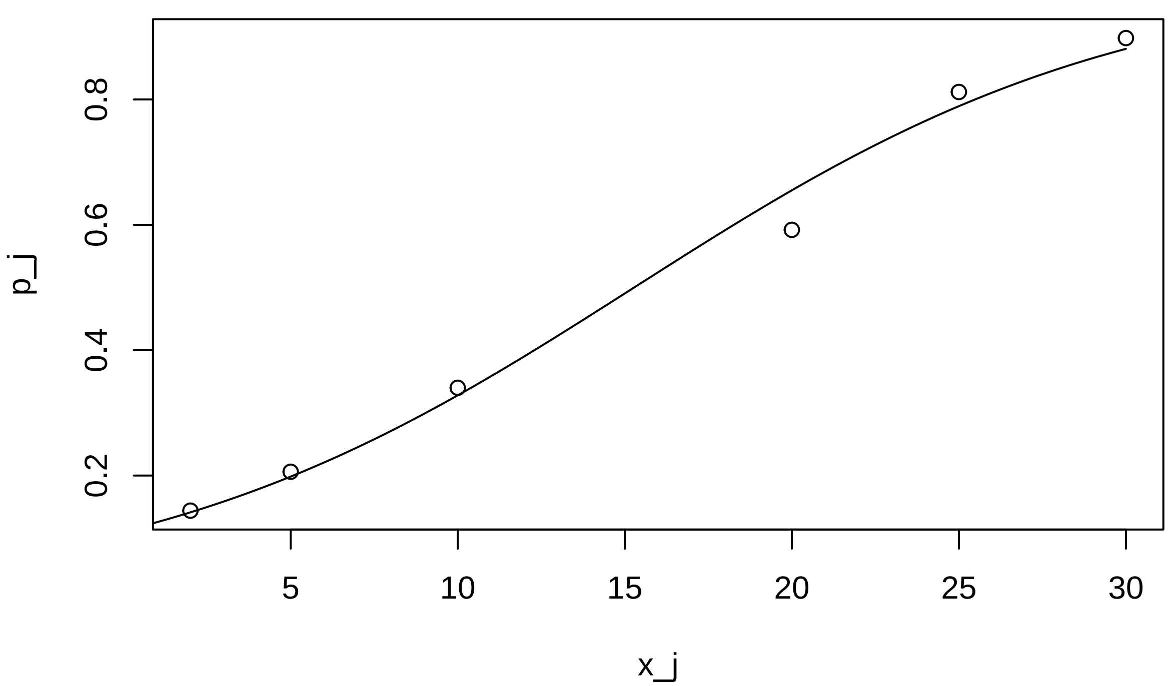
```
##
## Call:
## glm(formula = p_j ~ x_j, family = binomial)
##
## Deviance Residuals:
##         1          2          3          4          5
##  0.007846   0.019363   0.025865  -0.130556   0.056842
##         6
##  0.054601
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  -2.0766     1.8970  -1.095    0.274
## x_j           0.1359     0.1067   1.273    0.203
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.216343  on 5  degrees of freedom
## Residual deviance: 0.024363  on 4  degrees of freedom
## AIC: 7.1154
##
## Number of Fisher Scoring iterations: 4
```

The MLE of β_0 and β_1 are: -2.0766, 0.1359. The fitted response function is: $\hat{p} = \frac{exp(-2.0766+0.1359X)}{1+exp(-2.0766+0.1359X)}$

(c)

```
f = function(x) exp(-2.0766+0.1359*x)/(1+exp(-2.0766+0.1359*x))
par(mfrow =c(1,1))
plot(x_j,p_j)
curve(f, 0, 30, add=TRUE)
```



The fit looks pretty good.

(d)

```r
b1=0.1359
exp(b1)
```

```
## [1] 1.145567
```

exp(b1) is 1.1456, which means that the estimated oddes are multiplied by 1.1456 for any unit increase in x.

(e)

```r
f(15)
```

```
## [1] 0.4904762
```

The estimated probability is 0.4905.

(f)

```r
b0=-2.0766
b1-0.1359
```

```
## [1] 0
```

```r
p=0.75
x = (log(p/(1-p))-b0)/b1
x
```

```
## [1] 23.36433
```

The amount of deposit is estimated to be 23.3643.

14.14

(a)

```r
flu = read.table("http://www.stat.ufl.edu/~rrandles/sta4210/Rclassnotes/data/textdatasets/KutnerData/Ch
names(flu) = c("Y", "X1", "X2", "X3")
flu$X3 = as.factor(flu$X3)
flu$Y = as.numeric(flu$Y)
flu$X1 = as.numeric(flu$X1)
flu$X2 = as.numeric(flu$X2)

fit3 = glm(Y~X1+X2+X3, data = flu, family = binomial)
summary(fit3)
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2 + X3, family = binomial, data = flu)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4037  -0.5637  -0.3352  -0.1542   2.9394
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.17716    2.98242  -0.395  0.69307
## X1           0.07279    0.03038   2.396  0.01658 *
## X2          -0.09899    0.03348  -2.957  0.00311 **
## X31          0.43397    0.52179   0.832  0.40558
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 134.94  on 158  degrees of freedom
## Residual deviance: 105.09  on 155  degrees of freedom
## AIC: 113.09
##
## Number of Fisher Scoring iterations: 6
```

The MLE of β_0, β_1, β_2 and β_3 are: -1.17716, 0.07279, -0.09899, 0.43397. The fitted response function is: $\hat{\pi} = \frac{exp(-1.17716+0.07279X_1-0.09899X_2+0.43397X_3)}{1+exp(-1.17716+0.07279X_1-0.09899X_2+0.43397X_3)}$

(b)

```
b0 = -1.17716
b1 = 0.07279
b2 = -0.09899
b3 = 0.43397
exp(b1)
```

```
## [1] 1.075505
```

```
exp(b2)
```

```
## [1] 0.9057518
```

```
exp(b3)
```

```
## [1] 1.543373
```

exp(b1), exp(b2) and exp(b3) are 1.0755, 0.9058, 1.5434. It means that holding other variables constant, unit increse in x1 leads to the estimated odds multiplied by 1.0755; unit increse in x2 leads to the estimated odds multiplied by 0.9058; unit increse in x3 leads to the estimated odds multiplied by 1.5434.

(c)

```r
X_1 = 55
X_2 = 60
X_3 = 1
f = exp(-1.17716+0.07279*X_1-0.09899*X_2+0.43397*X_3)/(1+exp(-1.17716+0.07279*X_1-0.09899*X_2+0.43397*X_
f
```

```
## [1] 0.06421554
```

The estimated probabity is 0.642.

14.20

(c)

```r
flu$X3=as.numeric(as.character(flu$X3))
b0 = -1.17717-0.1^3
b1 = 0.07279
b2 = -0.09899
b3 = 0.43397
sum1 = 0
for (i in 1:159) {
sum1=sum1+flu[i,]$Y*(b0+b1*flu[i,]$X1+b2*flu[i,]$X2+b3*flu[i,]$X3)
}
sum2 = 0
for (i in 1:159){
sum2=sum2+log(1+exp(b0+b1*flu[i,]$X1+b2*flu[i,]$X2+b3*flu[i,]$X3))
}
sum1-sum2
```

```
## [1] -52.5466
```

```r
fit4 = glm(Y~X1+X2, data = flu, family = binomial)
b0 = -1.45778
b1=0.07787
b2 = -0.09547
sum1 = 0
for (i in 1:159) {
sum1=sum1+flu[i,]$Y*(b0+b1*flu[i,]$X1+b2*flu[i,]$X2)
}
sum2 = 0
for (i in 1:159){
sum2=sum2+log(1+exp(b0+b1*flu[i,]$X1+b2*flu[i,]$X2))
}
sum1-sum2
```

```
## [1] -52.89769
```

7

```r
library(pls)
```

```
##
## Attaching package: 'pls'
##
## The following object is masked from 'package:stats':
##
##     loadings
```

```r
apartment <- read.table("data/apartment.txt", header = T)
apartment <- as.data.frame(apply(apartment, 2, scale))
```

(a)

```r
set.seed(100)
plsr_model1 <- plsr(Y ~ 0 + ., 5, data = apartment, validation = 'CV')
# scores
scores(plsr_model1)
```

```
##          Comp 1      Comp 2      Comp 3       Comp 4
## 1  -1.43809414  0.31257891 -0.66465691 -0.238439173
## 2   1.65010464 -1.22067017 -0.64427193  0.663274381
## 3  -1.05278777  0.23602481 -0.31050170 -0.020291335
## 4   2.04913072  1.58847399  1.43430866  0.056609088
## 5  -1.13936477  0.29816779  0.09920242 -0.077974453
## 6   0.43417694 -0.26004572 -1.00952515  0.650608213
## 7  -1.22623584 -0.17772037 -1.20001375 -0.068573074
## 8  -0.65767426  0.36330099  1.03967609 -0.268877399
## 9   1.42140486 -1.39664727  1.71050459 -0.349167349
## 10  5.48855488  0.07922404 -1.42064358 -0.711330743
## 11  1.98211289  1.07654279 -0.19360273  0.247458345
## 12  0.04614307 -0.64535432  0.61846253 -0.480354282
## 13 -0.71466027  0.75759510 -0.14154088 -0.168335371
## 14 -0.89066011 -0.24878365  1.32675710 -0.194652611
## 15  0.41103483 -1.04629327  0.48064460  0.246649352
## 16 -1.02148397 -0.21942028  1.30789234 -0.154167563
## 17  0.77416093 -0.20479803 -0.70800362 -0.161374329
## 18 -1.24116621  0.12113528 -0.52207912 -0.071041829
## 19 -0.36723658 -0.32047183  0.43733622  0.267294081
## 20 -1.18318143  0.27366081  0.02535909 -0.014259103
## 21 -1.21222218 -0.07162508 -1.09128123 -0.058437672
## 22 -1.43423671  0.17603426  0.16663071 -0.203393131
## 23  1.20087805  0.46813038  1.30158137  0.899514178
## 24 -1.14387561 -0.16195504 -1.19339577 -0.004177028
## 25 -0.73482194  0.22291589 -0.84883933  0.213438810
##           Comp 5
## 1   0.159569798
```

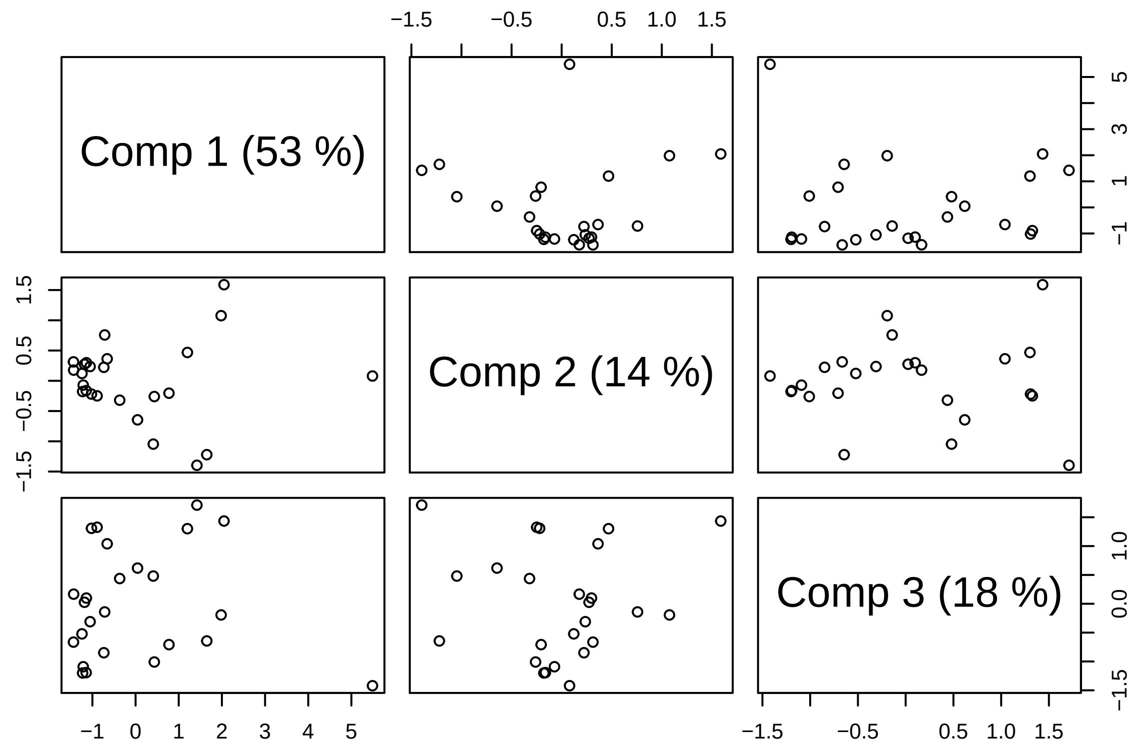
```
## 2  -0.865798881
## 3   0.008296669
## 4   0.998283179
## 5  -0.315614625
## 6   0.643434804
## 7   0.280347608
## 8  -1.370326788
## 9   1.208806076
## 10 -0.865767632
## 11  0.426744426
## 12  1.057377814
## 13  0.116132220
## 14 -0.421211010
## 15  0.306896982
## 16 -0.564857509
## 17 -0.174850186
## 18  0.036778165
## 19 -0.910453651
## 20 -0.184750720
## 21  0.309847588
## 22 -0.594314019
## 23 -0.313846232
## 24  0.359346058
## 25  0.669929866
## attr(,"class")
## [1] "scores"
## attr(,"explvar")
##    Comp 1    Comp 2    Comp 3    Comp 4    Comp 5
## 52.582242 14.361573 18.228133  6.180191  8.647861
```

```
# loadings
loadings(plsr_model1)[, 1:3]
```

```
##         Comp 1     Comp 2      Comp 3
## X1 -0.07983743  0.6903977 -0.76418606
## X2  0.59805736  0.1008482 -0.08164449
## X3  0.54164949 -0.5440785 -0.25282151
## X4  0.16890957 -0.7416964  0.59414237
## X5  0.56738864  0.5744813  0.04300711
```

```
n <- nrow(apartment)

apartment_mean <- apply(apartment, 2, mean)
SSTO <- sum((apartment$Y - apartment_mean[1])^2)
SSE <- apply((residuals(plsr_model1)[,,])^2, 2, sum)
# R square
R2 <- 1 - SSE / SSTO
# adjusted R square
R2_adjusted <- 1 - (1 - R2) * (n - 1) / (n - 1:5 - 1)
plot(plsr_model1, plottype = 'scores', comps = 1:3)
```



The model with 3 components is the best one, since more components does not give us more information.

(b)

```
seq_F <- (n - 2:5 - 1) * (R2[2:5] - R2[1:4]) / (1 - R2[2:5])
seq_F_1 <- (n - 1 - 1) * (R2[1] - 0) / (1 - R2[1])
seq_F <- c(seq_F_1, seq_F)
q_F <- sapply(1:5, function(x) qf(0.95, 1, n - x - 1))

summary(plsr_model1)
```

```
## Data:    X dimension: 25 5
##  Y dimension: 25 1
## Fit method: kernelpls
## Number of components considered: 5
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV           1.021   0.3411   0.2744   0.2002   0.1901
## adjCV        1.021   0.3382   0.2635   0.1928   0.1854
##        5 comps
## CV      0.2597
## adjCV   0.2508
```

```
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps
## X    52.58    66.94    85.17    91.35   100.00
## Y    92.14    96.53    97.92    98.01    98.05
```

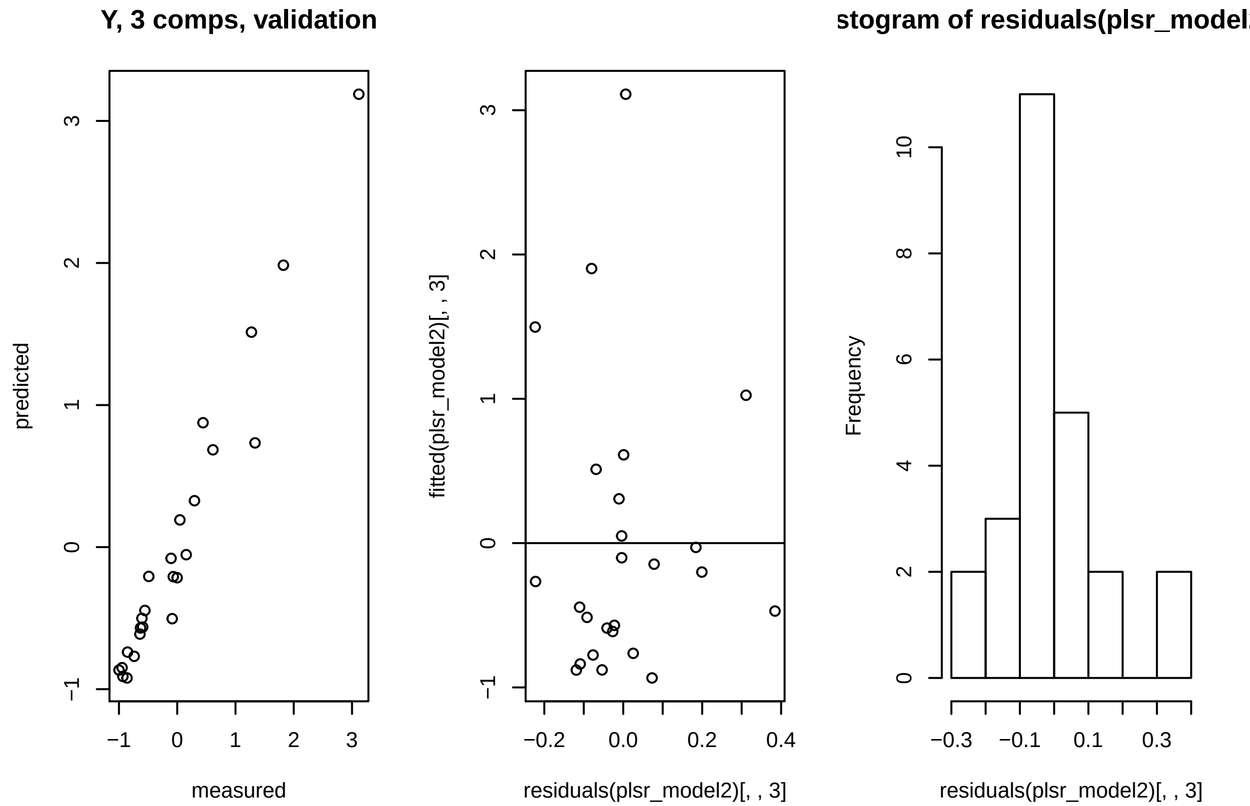
We should select model with 3 components under sequential F test. But under CV, 3 components and 4 components do not differ from each other very much.

(c)

```r
plsr_model2 <- plsr(Y ~ 0 + ., 3, data = apartment, validation = 'CV')
# coefficients
coef(plsr_model2)
```

```
## , , 3 comps
##
##               Y
## X1 -0.11356364
## X2  0.34543343
## X3 -0.02384503
## X4  0.05143543
## X5  0.67482746
```

```r
par(mfrow = c(1,3))
# fitted vs observed
plot(plsr_model2)
# residual vs fitted
plot(residuals(plsr_model2)[,,3], fitted(plsr_model2)[,,3])
abline(h = 0)
# histogram of residuals
hist(residuals(plsr_model2)[,,3])
```



This model is appropriate.

8

(a)

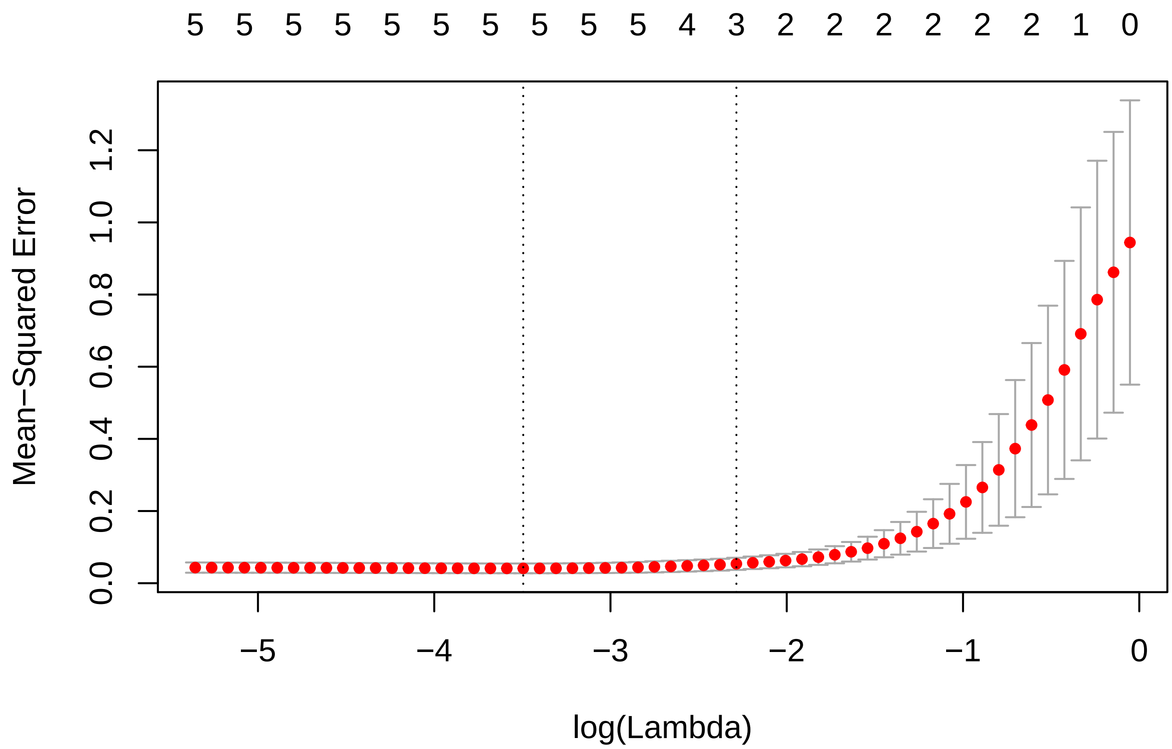
```r
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-3
```

```r
glmnet_model1 <- cv.glmnet(as.matrix(apartment[,-1]), apartment$Y, intercept = F)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since <
## 3 observations per fold
```

```r
# plot
plot(glmnet_model1)
```




```
# the value of penalty at which the cv is the smallest
glmnet_model1$lambda.min
```

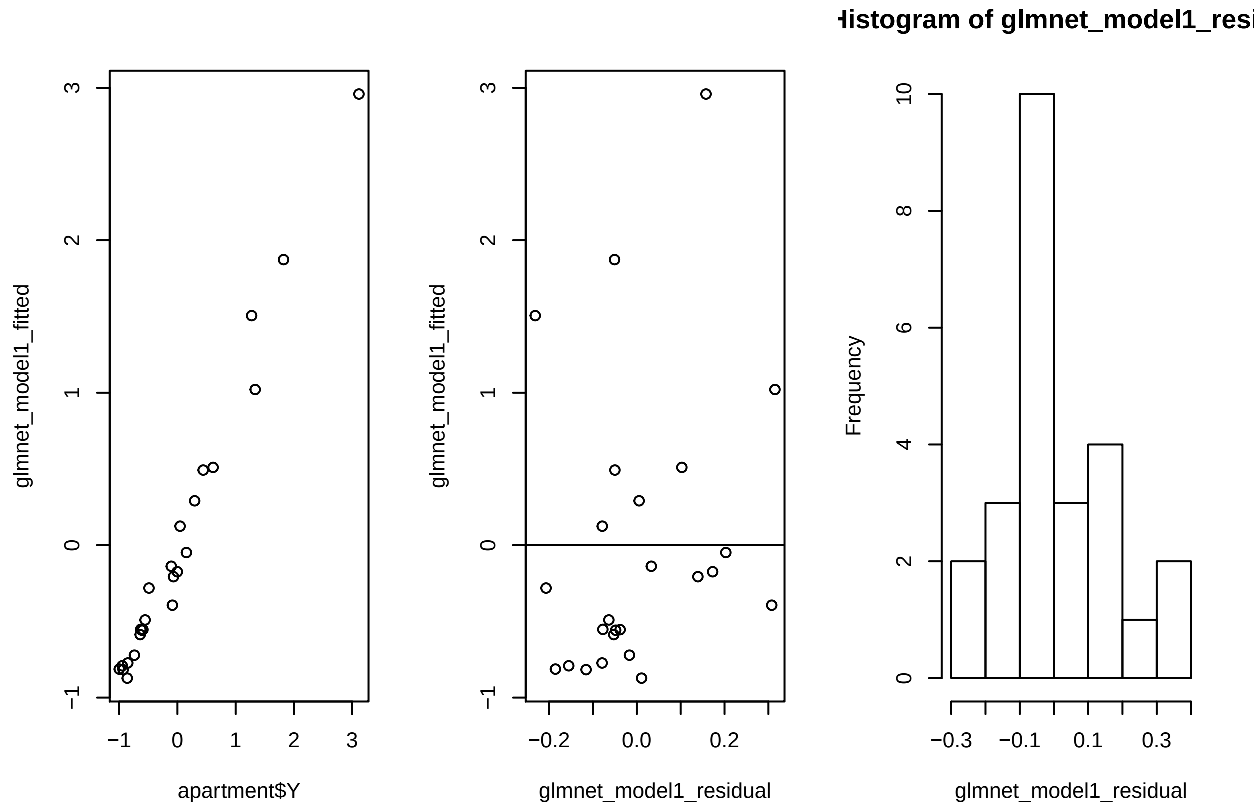
```
## [1] 0.03034732
```

(b)

```
# coefficient
coef(glmnet_model1, s = 'lambda.min')@x
```

```
## [1] -0.081952895  0.256348869  0.008781196  0.042445660
## [5]  0.704634852
```

```
glmnet_model1_fitted <- as.matrix(apartment[,-1]) %*% coef(glmnet_model1, s = 'lambda.min')@x
glmnet_model1_residual <- apartment$Y - glmnet_model1_fitted
par(mfrow = c(1,3))
# the observed against the fitted values
plot(apartment$Y, glmnet_model1_fitted)
# residuals against fitted
plot(glmnet_model1_residual, glmnet_model1_fitted)
abline(h = 0)
# histogram of residuals
hist(glmnet_model1_residual)
```



The model is a little left skewed.

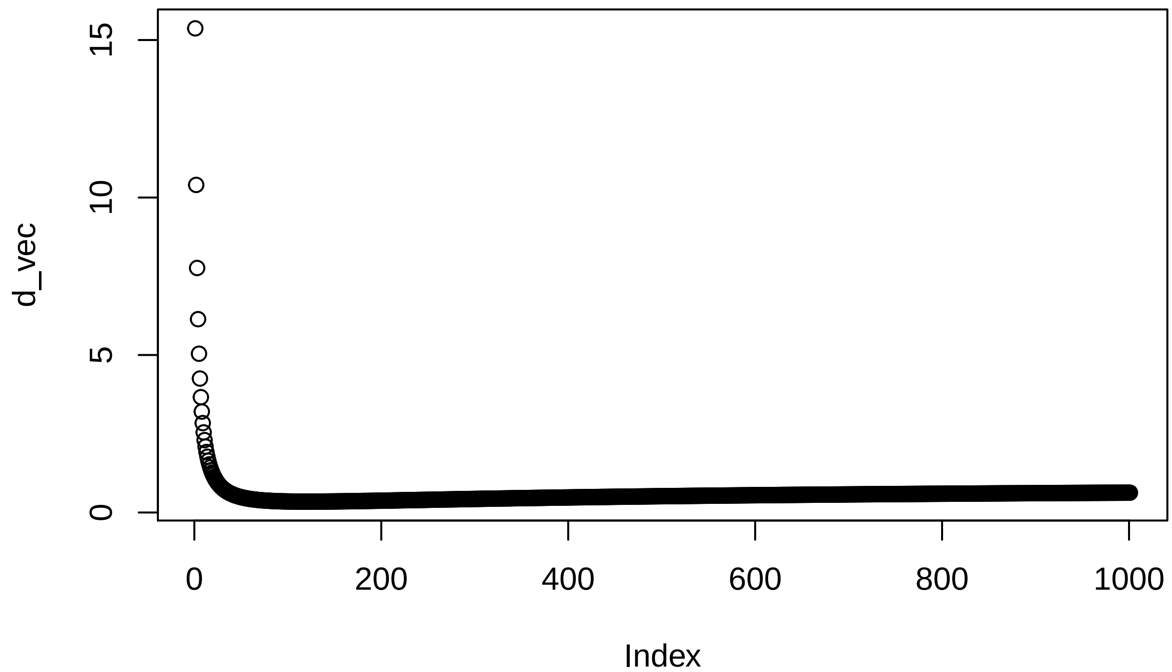
10

(a)

```
lambda <- c(19, 3, 1, 0.7, 0.3)
e_beta <- c(0.8, 0.3, 0.2, 0.2, 0.1)
sigma_square <- 2.5
k <- seq(0, 100, 0.1)

calculate_d <- function(k)
  sigma_square * sum(lambda / (k + lambda)^2) + k^2 * sum((e_beta)^2 / (k + lambda)^2)

d_vec <- sapply(k, calculate_d)
plot(d_vec)
```




```
which.min(d_vec) / 10
```

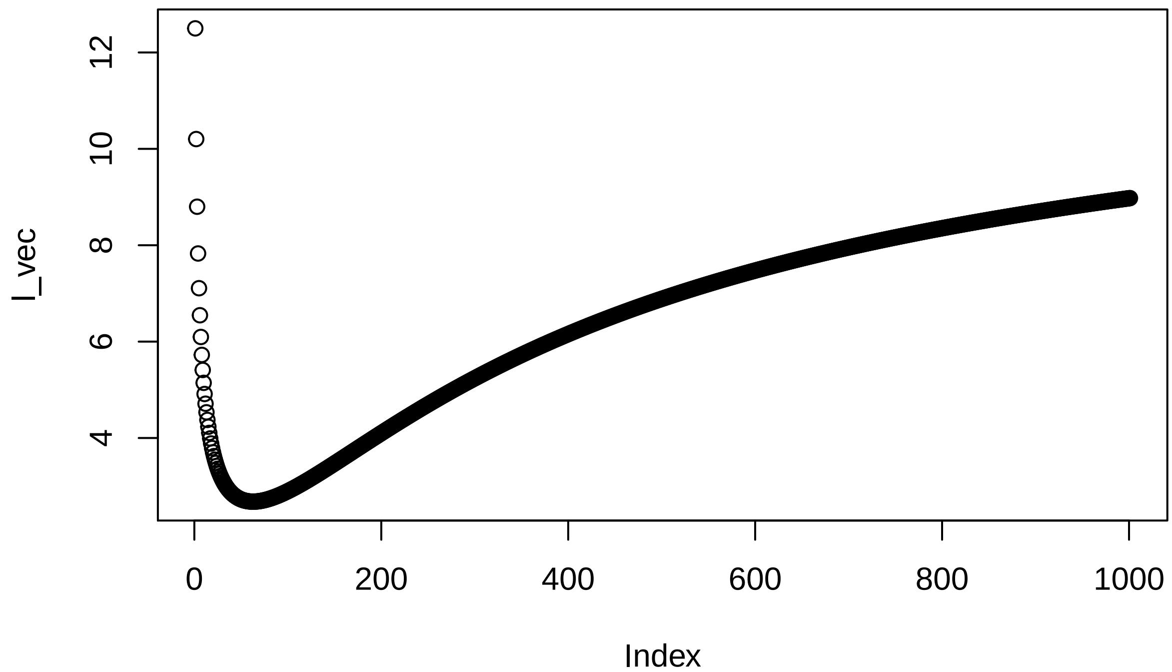
```
## [1] 12
```

So the value is 12

(b)

```
calculate_L <- function(k)
  sigma_square * sum(lambda^2 / (k + lambda)^2) + k^2 * sum((e_beta)^2 * lambda / (k + lambda)^2)

l_vec <- sapply(k, calculate_L)
plot(l_vec)
```




```r
which.min(l_vec) / 10
```

```
## [1] 6.3
```

So the value is 6.3

(c)

```r
calculate_d(0)
```

```
## [1] 15.36967
```

```r
calculate_d(12)
```

```
## [1] 0.3461745
```

```r
calculate_L(0)
```

```
## [1] 12.5
```

```r
calculate_L(6.3)
```

```
## [1] 2.680293
```