Assignment 7

STAT 32950

Ki Hyun

Due: 09:00 (CT) 2023-05-16

```
library(dplyr)
library(ggplot2)
library(MASS)
library(glmnet)
library(elasticnet)
library(fastICA)
```

Problem 1.

```
x1 = rnorm(30)
x2 = x1 + rnorm(30, sd = 0.01)
Y = rnorm(30, mean = 3 + x1 + x2)
```

(a)

```
OLS_model <- lm(Y ~ x1 + x2)
betas <- OLS_model$coefficients
summary(OLS_model)
```

```
##
## Call:
## lm(formula = Y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7170 -0.4208  0.1368  0.5474  1.4689
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.2373     0.1737  18.633   <2e-16 ***
## x1          -14.0323    16.1516  -0.869    0.393
## x2           15.8051    16.1457   0.979    0.336
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.9306 on 27 degrees of freedom
## Multiple R-squared:  0.8276, Adjusted R-squared:  0.8148
## F-statistic: 64.79 on 2 and 27 DF,  p-value: 4.95e-11
```

From the Least Square method, the fitted model with estimated parameters is as below:

$$\begin{aligned}\mathbf{E}[\hat{Y}] &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \\ &\approx 3.24 + (-14.03)x_1 + (15.81)x_2\end{aligned}$$

(b)

From the given code, the true model with the true β_i's are:

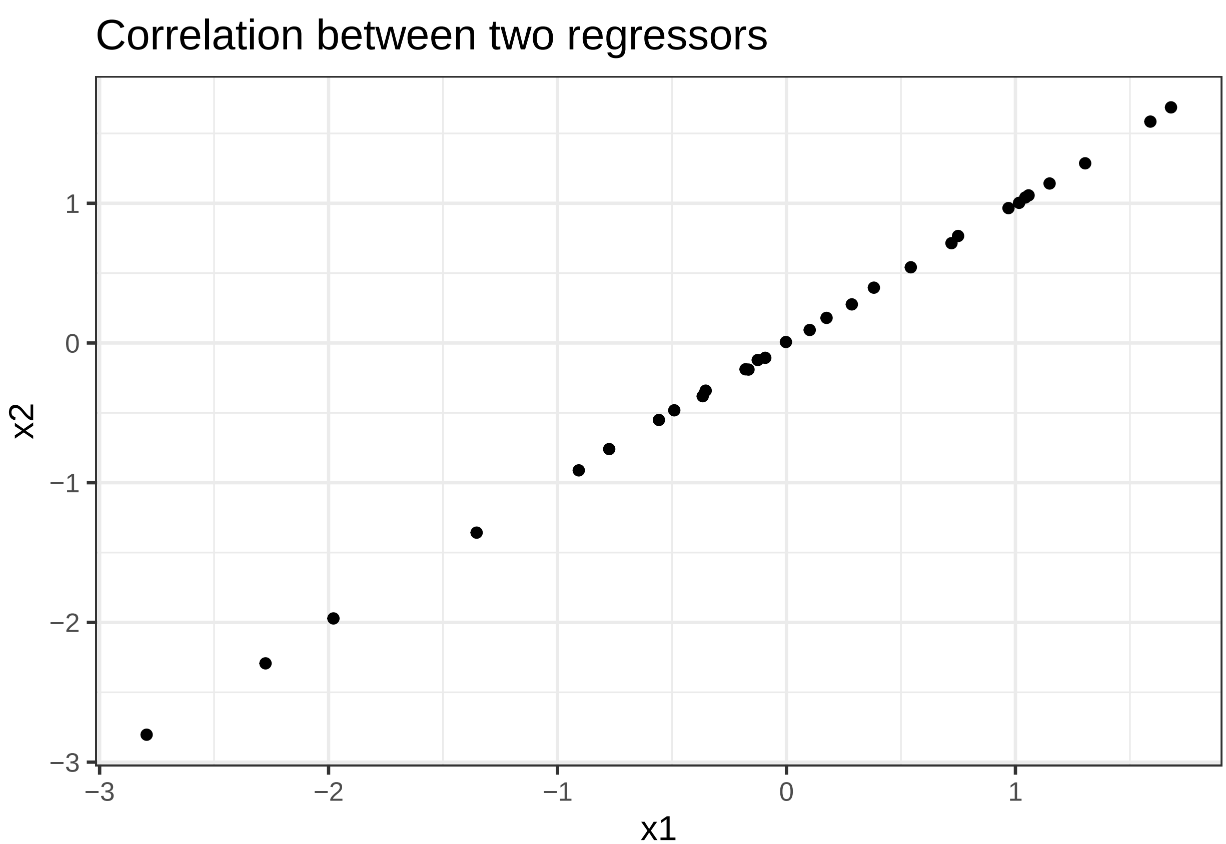
$$Y = 3 + x_1 + x_2 + \epsilon$$

where $\beta_0 = 3$, $\beta_1 = 1$, and $\beta_2 = 1$.

Compared to this true value, the LS model in (a) is **very bad**.

From the code, it appears that x_1 and x_2 are highly correlated. If we actually plot the two values:

```r
ggplot(data = tibble(x1 = x1, x2 = x2)) +
  geom_point(mapping = aes(x = x1, y = x2)) +
  labs(title = "Correlation between two regressors") +
  theme_bw(base_size = 13)
```



We can clearly see that the two independent variables are highly correlated. This would result in coefficient estimates that are far away from the true values.

(c)

```r
RSS_true = sum((Y - 3 - x1 - x2)^2)
print(paste0("The RSS of the true model: ", RSS_true))
```

```
## [1] "The RSS of the true model: 27.0703864638038"
```

```r
RSS_LS = sum(OLS_model$residuals^2)
print(paste0("The RSS of the LS model: ", RSS_LS))
```

```
## [1] "The RSS of the LS model: 23.3819260437571"
```

The two RSS are indeed comparable. In fact, the RSS of the "bad" LS model is lower than the true model.

This is the case since the LS parameter values are chosen to minimize the RSS value. Therefore, the optimized LS coefficients will result in not only comparable, but also the lowest in-sample RSS value.

(d)

```r
Ridge_model <- lm.ridge(Y ~ x1 + x2, lambda = 1, model = TRUE)
betas_ridge <- coef(Ridge_model)
summary(Ridge_model)
```

```
##        Length Class  Mode
## coef   2      -none- numeric
## scales 2      -none- numeric
## Inter  1      -none- numeric
## lambda 1      -none- numeric
## ym     1      -none- numeric
## xm     2      -none- numeric
## GCV    1      -none- numeric
## kHKB   1      -none- numeric
## kLW    1      -none- numeric
```

The fitted Ridge model is:

$$\begin{aligned}\mathbf{E}[\hat{Y}] &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \\ &\approx 3.2 + (0.85)x_1 + (0.9)x_2\end{aligned}$$

The parameter estimates are much closer to the true model.

(e)

The criterion of the LS method is the RSS. In mathematical expression:

$$\min_{\boldsymbol{\beta}} \sum_{j=1}^{n} \left[y_j - (\hat{\beta}_0 + \hat{\beta}_1 x_{1,j} + \hat{\beta}_2 x_{2,j}) \right]^2$$

The criterion of the Ridge method is the RSS and a l_2 penalty term on the coefficients. In mathematical expression:

$$\min_{\boldsymbol{\beta}} \left(\sum_{j=1}^{n} \left[y_j - (\hat{\beta}_0 + \hat{\beta}_1 x_{1,j} + \hat{\beta}_2 x_{2,j}) \right]^2 + \sum_{k=1}^{3} |\beta_k|^2 \right)$$

To recap, the result in (a) was:

```
OLS_model$coefficients
```

```
## (Intercept)          x1          x2
##     3.23727   -14.03226    15.80512
```

The result in (d) was:

```
Ridge_model
```

```
##                   x1        x2
## 3.2040985 0.8536845 0.8954694
```

As shown by comparing the absolute values of the coefficients for the results of (a) and (d), the Ridge regression reduces the magnitude of the coefficients.

Problem 2.

```r
data(Boston)
colnames(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

(a)

```r
Tdata = Boston[1:300,]
Cdata = Boston[301:506,]
X=as.matrix(Tdata[,1:13])
Y=Tdata[,14]
```

```r
trainfit = glmnet(X, Y)
nx = as.matrix(Cdata[, 1:13])
ny = Cdata[, 14]
calibrate_mse = colMeans((predict(trainfit, newx = nx) - ny)^2)
lambda_star <- trainfit$lambda[which.min(calibrate_mse)]
```

```r
betas_LASSO <- coef(trainfit, s = lambda_star)
betas_LASSO
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -19.190331147
## crim          .
## zn            .
## indus         .
## chas          0.131996136
## nox           .
## rm            9.193505503
## age          -0.024883019
## dis          -0.358735094
## rad           .
## tax          -0.009113364
## ptratio      -0.586462537
## black         0.008388966
## lstat        -0.119856133
```

The optimal model after calibration is:

$$
\begin{aligned}
\mathbf{E}[Y_{MEDV}] \approx & -19.19+ \\
& (0.13)X_{chas} + (9.19)X_{rm} + (-0.02)X_{age} + (-0.36)X_{dis}+ \\
& (-0.01)X_{tax} + (-0.59)X_{ptratio} + (0.01)X_{black} + (-0.12)X_{lstat}
\end{aligned}
$$

The independent variables crim, zn, indus, nox, rad were excluded from the model as their coefficients were optimized at 0 after the l_1 penalty.

(b)

```r
OLS_model2 <- lm(medv ~ ., data = Boston)
betas2 <- OLS_model2$coefficients
summary(OLS_model2)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

The OLS model result is:

$$
\begin{aligned}
\mathbf{E}[Y_{MEDV}] \approx\, & 36.46 + \\
& (-0.11)X_{crim} + (0.046)X_{zn} + (0.021)X_{indus} + (2.69)X_{chas} + \\
& (-17.77)X_{nox} + (3.81)X_{rm} + (6.9 \times 10^{-4})X_{age} + (-1.48)X_{dis} + \\
& (0.31)X_{rad} + (-0.012)X_{tax} + (-0.95)X_{ptratio} + (0.009)X_{black} + \\
& (-0.52)X_{lstat}
\end{aligned}
$$

As expected, there is no dimension reduction for the OLS model. Moreover, the intercept and the coefficients for chas, rm, age, dis, tax, ptratio, black, lstat are very different between the two models.

We expect the in-sample MSE to be lower for the OLS model but the out-of-sample MSE to be much lower for the LASSO model.

Problem 3.

```r
data = read.csv("hearlossData.csv")
colnames(data)=c("Left5c","Left1k","Left2k","Left4k",
                 "Right5c","Right1k","Right2k","Right4k")
```

(a)

```r
# variance of each column
diag(cov(data))
```

```
##    Left5c    Left1k    Left2k    Left4k   Right5c   Right1k   Right2k   Right4k
##  41.40899  57.59637 120.25871 388.28592  51.19563  41.00186  87.06452 377.06143
```

```r
summary(princomp(data))
```

```
## Importance of components:
##                           Comp.1     Comp.2      Comp.3     Comp.4     Comp.5
## Standard deviation     26.5815753 13.3295905 10.55117531 9.31933963 5.41682080
## Proportion of Variance  0.6132885  0.1542187  0.09662846 0.07538304 0.02546785
## Cumulative Proportion   0.6132885  0.7675071  0.86413560 0.93951864 0.96498649
##                           Comp.6     Comp.7      Comp.8
## Standard deviation     4.45325679 3.61865582 2.722773605
## Proportion of Variance 0.01721309 0.01136575 0.006434672
## Cumulative Proportion  0.98219958 0.99356533 1.000000000
```

For the un-scaled data, there are a total of 8 principal components. The decomposition of each 8 columns in terms of the principal components are:

```r
princomp(data)$loadings
```

```
##
## Loadings:
##         Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## Left5c          0.294         0.384  0.130         0.550  0.656
## Left1k   0.110  0.398         0.316  0.287  0.602        -0.534
## Left2k   0.223  0.556        -0.447  0.521 -0.387 -0.119
## Left4k   0.678 -0.114  0.712        -0.109
## Right5c         0.279         0.495 -0.317 -0.642  0.134 -0.377
## Right1k         0.310         0.276 -0.262  0.105 -0.774  0.374
## Right2k  0.171  0.375 -0.274 -0.450 -0.660  0.227  0.255
## Right4k  0.656 -0.344 -0.643  0.155  0.111
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.125  0.125  0.125  0.125  0.125  0.125  0.125  0.125
## Cumulative Var  0.125  0.250  0.375  0.500  0.625  0.750  0.875  1.000
```

In terms of the first 2 principal components:

```
princomp(data)$loadings[, 1:2]
```

```
##             Comp.1     Comp.2
## Left5c  0.08379682  0.2943489
## Left1k  0.10966429  0.3976918
## Left2k  0.22304507  0.5563308
## Left4k  0.67791950 -0.1139214
## Right5c 0.06647239  0.2788279
## Right1k 0.08959874  0.3102494
## Right2k 0.17107268  0.3746639
## Right4k 0.65567943 -0.3440134
```

We may repeat the same after scaling the data.

```
summary(princomp(data), cor = T)
```

```
## Importance of components:
##                            Comp.1     Comp.2      Comp.3     Comp.4     Comp.5
## Standard deviation     26.5815753 13.3295905 10.55117531 9.31933963 5.41682080
## Proportion of Variance  0.6132885  0.1542187  0.09662846 0.07538304 0.02546785
## Cumulative Proportion   0.6132885  0.7675071  0.86413560 0.93951864 0.96498649
##                            Comp.6     Comp.7      Comp.8
## Standard deviation     4.45325679 3.61865582 2.722773605
## Proportion of Variance 0.01721309 0.01136575 0.006434672
## Cumulative Proportion  0.98219958 0.99356533 1.000000000
```

For the scaled data, there are a total of 8 principal components. The decomposition of each 8 scaled columns in terms of the principal components are:

```
princomp(data, cor = T)$loadings
```

```
##
## Loadings:
##         Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## Left5c   0.401  0.320  0.153  0.329         0.443  0.332  0.546
## Left1k   0.420  0.226         0.482 -0.381               -0.621
## Left2k   0.366 -0.243 -0.469  0.283  0.439        -0.524  0.188
## Left4k   0.283 -0.470  0.433  0.160  0.345 -0.421  0.427
## Right5c  0.343  0.389  0.254 -0.485  0.501  0.194 -0.158 -0.343
## Right1k  0.411  0.232        -0.377 -0.355 -0.609         0.361
## Right2k  0.311 -0.320 -0.562 -0.391 -0.110  0.265  0.478 -0.148
## Right4k  0.256 -0.509  0.431 -0.159 -0.390  0.374 -0.412
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.125  0.125  0.125  0.125  0.125  0.125  0.125  0.125
## Cumulative Var  0.125  0.250  0.375  0.500  0.625  0.750  0.875  1.000
```

In terms of the first 2 principal components:

```r
princomp(data, cor = T)$loadings[, 1:2]
```

```
##            Comp.1     Comp.2
## Left5c  0.4005783  0.3197367
## Left1k  0.4204255  0.2261167
## Left2k  0.3657558 -0.2430834
## Left4k  0.2830992 -0.4695360
## Right5c 0.3426632  0.3893248
## Right1k 0.4109096  0.2322277
## Right2k 0.3111725 -0.3200733
## Right4k 0.2564449 -0.5090537
```

(b)

Problem 4.

```r
X = read.table("tableICA.txt")
```

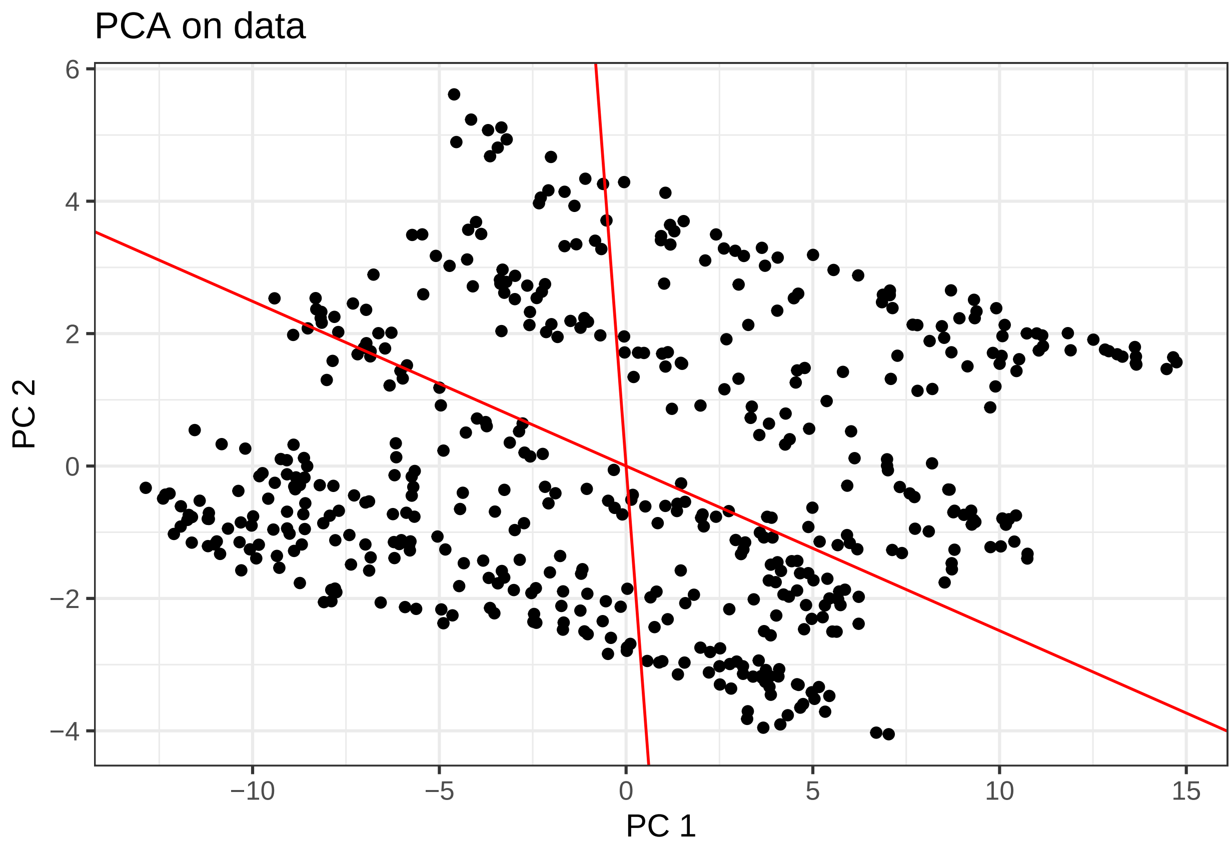
(a)

```r
Mpca = princomp(X)
summary(Mpca)
```

```
## Importance of components:
##                           Comp.1     Comp.2      Comp.3
## Standard deviation     6.5057138 2.10975775 0.513957832
## Proportion of Variance 0.8997603 0.09462417 0.005615545
## Cumulative Proportion  0.8997603 0.99438446 1.000000000
```

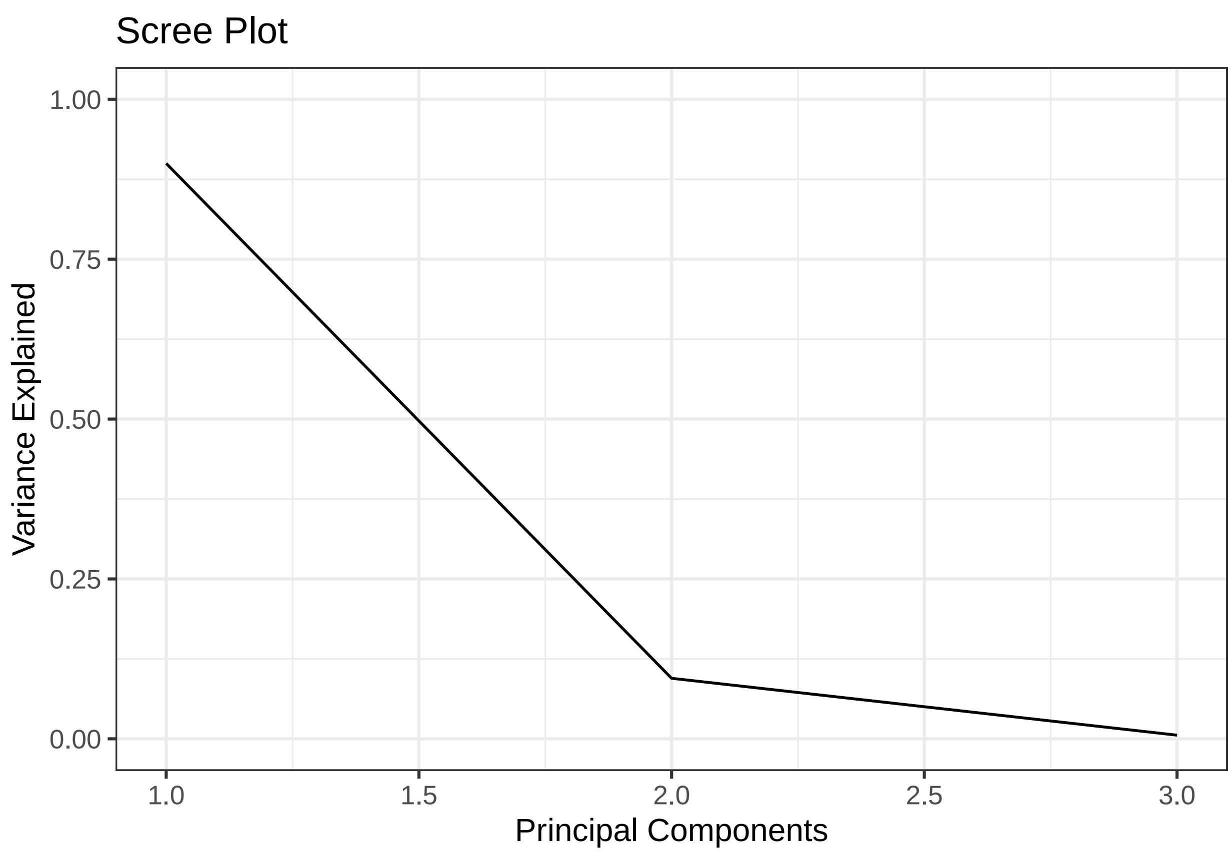
The plotting of the observations by the first two principal components

```r
ggplot(data = as_tibble(Mpca$scores)) +
  geom_point(mapping = aes(x = Comp.1, y = Comp.2)) +
  geom_abline(slope = Mpca$loading[1,1]/Mpca$loading[2,1],
              intercept = 0, col = "red") +
  geom_abline(slope = Mpca$loading[1,2]/Mpca$loading[2,2],
              intercept = 0, col = "red") +
  labs(title = "PCA on data",
       x = "PC 1", y = "PC 2") +
  theme_bw(base_size = 12)
```



The scree plot is shown below:

```
ggplot(tibble(x = c(1:3), y = Mpca$sdev^2 / sum(Mpca$sdev^2))) +
  geom_line(mapping = aes(x, y)) +
  labs(x = "Principal Components",
       y = "Variance Explained",
       title = "Scree Plot")+
  ylim(0, 1) +
  theme_bw(base_size = 12)
```



The scree plot shows that the first two principal components explain most of the variance within the data. Nevertheless, the plot of the observations with the first two PCs as axis shows that the combination of the two principal components would better explain the data as the plot resembles a linear pattern in layers.

(b)

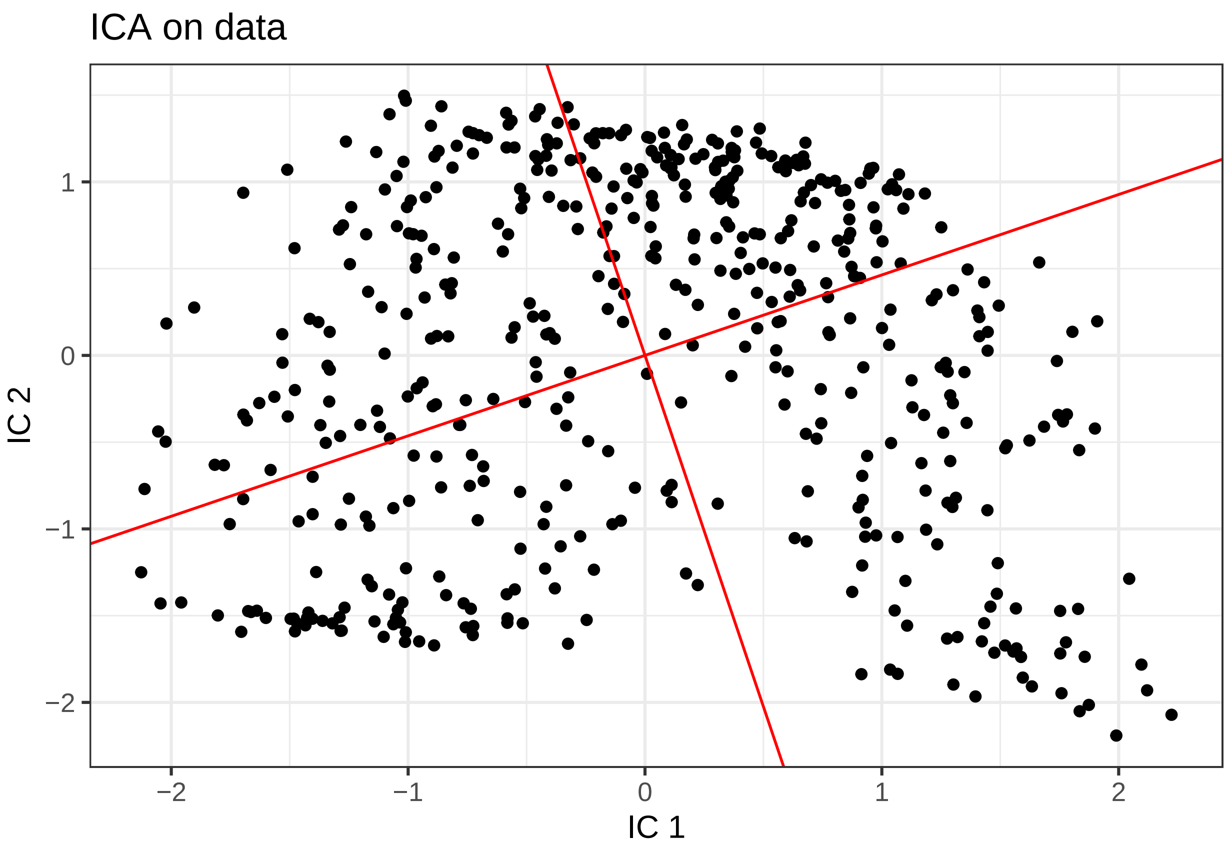
```
Mica = fastICA(X, 3)
summary(Mica$S)
```

```
##        V1                 V2                 V3
##  Min.   :-2.1271   Min.   :-2.1910   Min.   :-1.786949
##  1st Qu.:-0.8599   1st Qu.:-0.8403   1st Qu.:-0.989343
##  Median : 0.0223   Median : 0.1919   Median :-0.005626
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000000
##  3rd Qu.: 0.8175   3rd Qu.: 0.9236   3rd Qu.: 0.947140
##  Max.   : 2.2232   Max.   : 1.4967   Max.   : 1.711369
```

If we plot the Independent Components on the data:

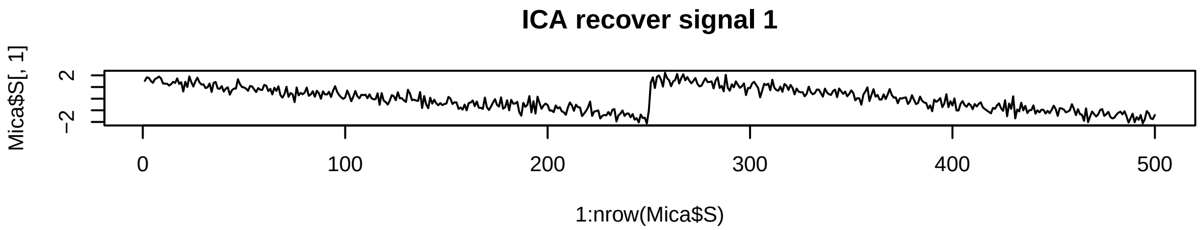
```
ggplot(data = tibble(IC1 = Mica$S[, 1], IC2 = Mica$S[, 2])) +
  geom_point(mapping = aes(x = IC1, y = IC2)) +
  geom_abline(slope = Mica$W[1,1]/Mica$W[2,1],
              intercept = 0, col = "red") +
```

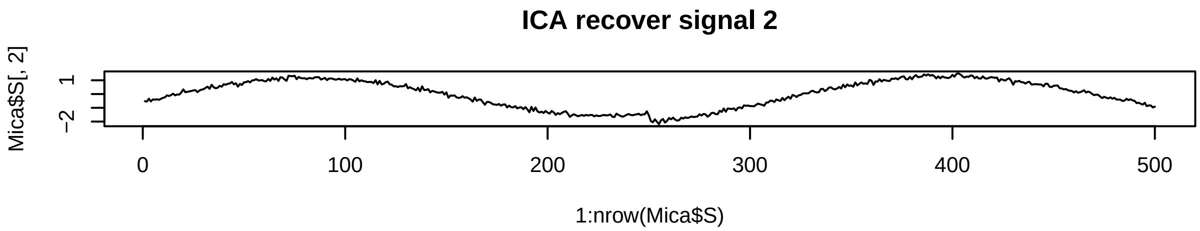
```r
  geom_abline(slope = Mica$W[1,2]/Mica$W[2,2],
              intercept = 0, col = "red") +
  labs(title = "ICA on data",
       x = "IC 1", y = "IC 2") +
  theme_bw(base_size = 12)
```

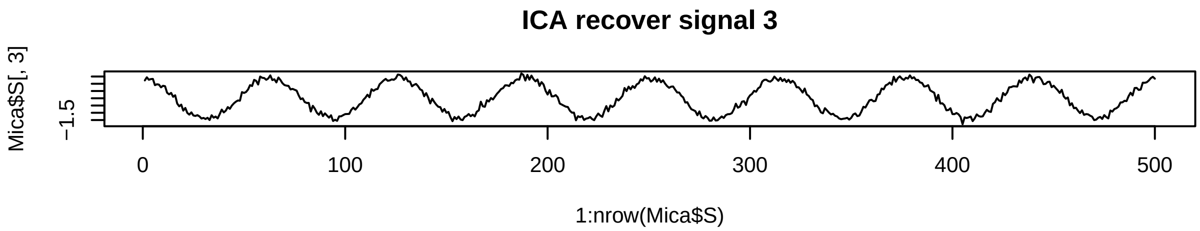


The plots of ICA recovered signals are included below:

```r
par(mfrow = c(3, 1))
plot(1:nrow(Mica$S),Mica$S[,1],cex=.5,type="l"); title("ICA recover signal 1")
plot(1:nrow(Mica$S),Mica$S[,2],cex=.5,type="l"); title("ICA recover signal 2")
plot(1:nrow(Mica$S),Mica$S[,3],cex=.5,type="l"); title("ICA recover signal 3")
```

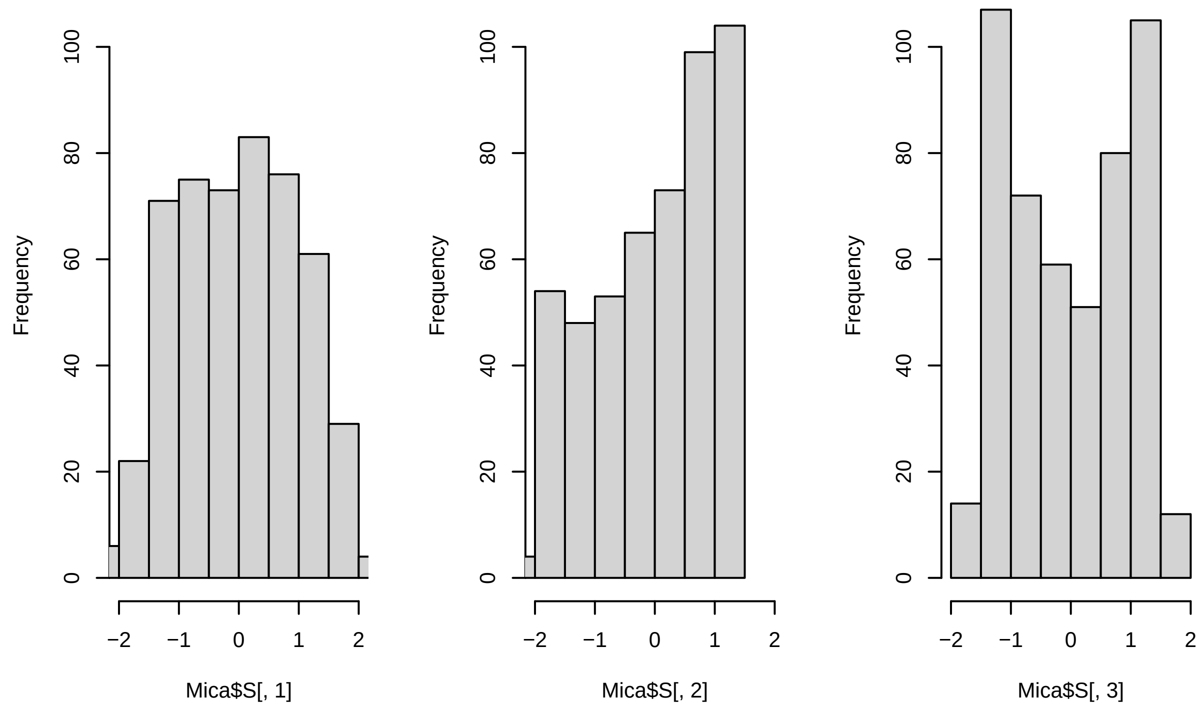






The plot of the three independent components recovered are included below:

```
par(mfrow = c(1, 3))
hist(Mica$S[,1],main="", xlim=c(-2,2),ylim=c(0,110))
hist(Mica$S[,2],main="", xlim=c(-2,2),ylim=c(0,110))
hist(Mica$S[,3],main="", xlim=c(-2,2),ylim=c(0,110))
```



(c)

The IC analysis seems to separate the signals and decompose the data into different dimensions better. The different "layers" present in the PC analysis plot is no longer present in the IC analysis. Moreover, comparing (a) and (b) may suggest the existence of independent source that is far from a Gaussian distribution.

Problem 5.

(a)

$$
\begin{aligned}
H(X) &= -\int_{\mathbf{R}} \phi(x)\log\phi(x)dx \\
&= -\int_{\mathbf{R}} \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}\left(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2 - \log(\sigma\sqrt{2\pi})\right)dx \\
&= \log(\sigma\sqrt{2\pi})\int_{\mathbf{R}} \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}dx + \int_{\mathbf{R}} \frac{1}{2}(\frac{x-\mu}{\sigma})^2\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}dx \\
&= \log(\sigma\sqrt{2\pi}) + \frac{1}{2\sigma^2}\int_{\mathbf{R}} x^2\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}dx - \frac{\mu}{\sigma^2}\int_{\mathbf{R}} x\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}dx + \frac{\mu^2}{2\sigma^2}\int_{\mathbf{R}} \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}dx \\
&= \log(\sigma\sqrt{2\pi}) + \frac{\sigma^2+\mu^2}{2\sigma^2} - \frac{\mu^2}{\sigma^2} + \frac{\mu^2}{2\sigma^2} \\
&= \log(\sigma\sqrt{2\pi}) + \frac{1}{2} \\
&= \log(\sigma\sqrt{2\pi}\times e^{\frac{1}{2}}) \\
&= \log(\sigma\sqrt{2\pi e})
\end{aligned}
$$

Q.E.D.

(b)

$$
\begin{aligned}
H(X) &= -\int_{\mathbf{R}} f(x)\log\phi(x)dx \\
&= -\int_{\mathbf{R}} f(x)\left(-\frac{1}{2}(\frac{x}{\sigma})^2 - \log(\sigma\sqrt{2\pi})\right)dx \\
&= \log(\sigma\sqrt{2\pi})\int_{\mathbf{R}} f(x)dx + \int_{\mathbf{R}} \frac{1}{2}(\frac{x}{\sigma})^2 f(x)dx \\
&= \log(\sigma\sqrt{2\pi}) + \frac{1}{2\sigma^2}\int_{\mathbf{R}} x^2 f(x)dx \\
&= \log(\sigma\sqrt{2\pi}) + \frac{\sigma^2+0^2}{2\sigma^2} \\
&= \log(\sigma\sqrt{2\pi}) + \frac{1}{2} \\
&= \log(\sigma\sqrt{2\pi}e^{-\frac{1}{2}}) \\
&= \log(\sigma\sqrt{2\pi e})
\end{aligned}
$$

Q.E.D.

(c)

If we look at the function

$$\log\phi(x) = -\frac{1}{2}(\frac{x}{\sigma})^2 - \log(\sigma\sqrt{2\pi})$$

it takes a convex form.

Therefore, if we let $h(x) = \log\phi(x)$ then the differential entropy examined in (b) can be expressed as $-\mathbf{E}[h(x)]$.

$$-\int_{\mathbf{R}} f(x)\log\phi(x)dx = -\mathbf{E}[h(X)]$$

Since $h(x)$ is a convex function, we know from Jensen's Inequality that

$$\mathbf{E}[h(X)] \leq h(\mathbf{E}[X])$$

Therefore,

$$-\int_{\mathbf{R}} f(x)\log\phi(x)dx = -\mathbf{E}[h(X)] \geq -h(\mathbf{E}[X])$$

It was also defined that $\mathbf{E}[X] = 0$. From this,

$$-\int_{\mathbf{R}} f(x)\log\phi(x)dx \geq -h(0) = -\log\phi(0) = \log(\sigma\sqrt{2\pi})$$

(d)

First since Y is defined by $X_1 + X_2$, Y is also a continuous random variable on the real line.

Now if we examine the mean of Y

$$\mathbf{E}[Y] = \mathbf{E}[X_1 + X_2] = \mathbf{E}[X_1] + \mathbf{E}[X_2] = 0$$

For the variance,

$$Var[Y] = Var[X_1 + X_2] = Var[X_1] + Var[X_2] + 2Cov[X_1, X_2]$$

If we denote the variance of Y as σ_Y^2 and the correlation between X_1 and X_2 as $\rho_{1,2}$,

$$\sigma_Y^2 = \sigma_1^2 + \sigma_2^2 + 2\rho_{1,2}\sigma_1\sigma_2$$

We know from (c) that

$$H(Y) \leq \log(\sigma_Y\sqrt{2\pi e})$$

Moreover, we know that the equation only holds when Y follows a normal distribution. This would mean that, first, to maximize the differential entropy, both X_1 and X_2 would need to follow the normal distribution in order for Y to follow a normal distribution.

Secondly, looking at the break down of σ_Y, the $\log(\sigma_Y\sqrt{2\pi e})$ value itself would be maximized when $\rho 1, 2 = 1$.

Therefore, my choice of X_1 and X_2 would be:

$$X_1 = \frac{\sigma_1}{\sigma_2}X_2 \sim N(0, \sigma_1^2)$$

Problem 6.

(a)

$$\mathbf{E}[Y] = \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_p \end{pmatrix} \in \mathbf{R}^p$$

(b)

Since Y_i are independent, the covariance matrix becomes:

$$Cov(Y) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & 1 \end{bmatrix} \in \mathbf{R}^{p \times p}$$

(c)

i)

$$\boldsymbol{\mu} = \mathbf{E}[Y] = \sum_{c=1}^{K} \pi_c \ \boldsymbol{\mu}_c = \begin{pmatrix} \sum_{c=1}^{K} \pi_c \mu_{c1} \\ \vdots \\ \sum_{c=1}^{K} \pi_c \mu_{cp} \end{pmatrix}$$

ii)

$$\mathbf{P}(\mathbf{y} \mid \boldsymbol{\mu}_c) = \prod_{i=1}^{p} \mu_{ci}^{y_i} (1 - \mu_{ci})^{1-y_i}$$

iii)

$$\begin{aligned} \mathbf{P}(\mathbf{y} \mid \boldsymbol{\pi}, \boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K) &= \sum_{k=1}^{K} \mathbf{P}(\mathbf{y} \mid \boldsymbol{\mu}_c = \boldsymbol{\mu}_k) \times \mathbf{P}(\boldsymbol{\mu}_k \mid \boldsymbol{\pi}) \\ &= \sum_{c=1}^{K} \left(\pi_c \prod_{i=1}^{p} \mu_{ci}^{y_i} (1 - \mu_{ci})^{1-y_i} \right) \end{aligned}$$

iv)

First and foremost, we should note

$$Cov(Y) = \mathbf{E}(Cov(Y \mid C)) + Cov(\mathbf{E}(Y \mid C))$$

If we focus on the first part:

$$\mathbf{E}(Cov(Y \mid C)) = \mathbf{E}(\Sigma_c) = \sum_{c=1}^{K} \pi_c \Sigma_c$$

Now for the second part:

$$Cov(\mathbf{E}(Y \mid C)) = Cov(\boldsymbol{\mu}_c)$$

Let's look at μ_{ci} and μ_{cj}.

$$Cov(\mu_{ci}, \mu_{cj}) = \sum_{c=1}^{K} \mu_{ci}\mu_{cj}\pi_c - \left(\sum_{c=1}^{K} \mu_{ci}\pi_c\right)\left(\sum_{c=1}^{K} \mu_{cj}\pi_c\right)$$

Therefore, $Cov(\boldsymbol{\mu}_c)$ may be expressed as:

$$Cov(\boldsymbol{\mu}_c) = \sum_{c=1}^{K} \pi_c \boldsymbol{\mu}_c \boldsymbol{\mu}_c^T - \left(\sum_{c=1}^{K} \pi_c \boldsymbol{\mu}_c\right)\left(\sum_{c=1}^{K} \pi_c \boldsymbol{\mu}_c\right)^T$$

Ultimately,

$$Cov(Y) = \sum_{c=1}^{K} \pi_c \Sigma_c + \sum_{c=1}^{K} \pi_c \boldsymbol{\mu}_c \boldsymbol{\mu}_c^T - \left(\sum_{c=1}^{K} \pi_c \boldsymbol{\mu}_c\right)\left(\sum_{c=1}^{K} \pi_c \boldsymbol{\mu}_c\right)^T$$

(d)

i)

Using the expression in (c) iii):

$$L(\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K, \boldsymbol{\pi} \mid \mathbf{Y}) = \mathbf{P}(\mathbf{Y} \mid \boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K, \boldsymbol{\pi}) = \prod_{i=1}^{n} \left(\sum_{c=1}^{K} \left(\pi_c \prod_{j=1}^{p} \mu_{cj}^{y_j^{(i)}} (1 - \mu_{cj})^{1 - y_j^{(i)}}\right)\right)$$

ii)