ASSIGNMENT 4

1.(a)

model1 is created as an original before stepwise

Here is the code:

```
#1.(a)
wine <- read.csv("/Users/apple/Downloads/winequality-red (2).csv", sep=";")
wine$quality <- factor(wine$quality)
levels(wine$quality) <- c("bad","bad","average","average","good",'good')
library(nnet)
model1 <- multinom(quality~fixed.acidity + volatile.acidity + citric.acid +
                     residual.sugar + chlorides + free.sulfur.dioxide +
                     total.sulfur.dioxide + density + pH + sulphates + alcohol,
                   data = wine, trace = FALSE)
summary(model1)
```

Here is the output:

```
> summary(model1)
Call:
multinom(formula = quality ~ fixed.acidity + volatile.acidity +
    citric.acid + residual.sugar + chlorides + free.sulfur.dioxide +
    total.sulfur.dioxide + density + pH + sulphates + alcohol,
    data = wine, trace = FALSE)

Coefficients:
        (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
average   -178.8060   -0.26900223        -4.462724  -1.2742330   -0.259419065  -5.485951
good       122.3626    0.05907234        -6.782318  -0.6097511    0.009571902 -13.977148
        free.sulfur.dioxide total.sulfur.dioxide   density        pH sulphates   alcohol
average          0.01736726         0.0161405294  197.9551 -4.534165  1.199091 0.4310564
good             0.02721645        -0.0004499205 -119.7577 -3.941803  5.012159 1.1267605

Std. Errors:
        (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
average    2.705097     0.1543634         0.835823    1.218232     0.08282364  3.101619
good       3.129535     0.1706778         1.110980    1.444455     0.09777426  4.459550
        free.sulfur.dioxide total.sulfur.dioxide  density       pH sulphates   alcohol
average          0.02292630          0.008267819 2.646911 1.391239  1.269845 0.1758899
good             0.02565747          0.009496124 3.052356 1.600228  1.358037 0.1949868

Residual Deviance: 1295.352
AIC: 1343.352
```

model2 is created after stepwise

Here is the code:

```
model2 <- step(model1, scope=~., trace=0)
summary(model2)
```

Here is the output:

```
> summary(model2)
Call:
multinom(formula = quality ~ fixed.acidity + volatile.acidity +
    citric.acid + residual.sugar + chlorides + total.sulfur.dioxide +
    density + pH + sulphates + alcohol, data = wine, trace = FALSE)

Coefficients:
        (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
average   -177.1713   -0.26252634        -4.565347  -1.4550010    -0.24141265  -5.434441
good       125.7384    0.07207673        -6.952064  -0.8516209     0.02213188 -13.805282
        total.sulfur.dioxide   density        pH sulphates   alcohol
average          0.020732275  196.0641 -4.486351  1.255191 0.4420856
good             0.006790962 -123.6146 -3.799881  5.050087 1.1345209

Std. Errors:
        (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
average    2.704340     0.1545768        0.8275218    1.201010     0.07916868  3.111854
good       3.125144     0.1708142        1.1010330    1.427641     0.09566638  4.468744
        total.sulfur.dioxide  density       pH sulphates   alcohol
average          0.006059025 2.649079 1.393743  1.273329 0.1756397
good             0.006894894 3.048687 1.597277  1.360258 0.1946937

Residual Deviance: 1296.663
AIC: 1340.663
```

1.(b)

model3 is created before stepwise

Here is the code:

```
#1.(b)
library(MASS)
model3 <- polr(quality~fixed.acidity + volatile.acidity + citric.acid +
                 residual.sugar + chlorides + free.sulfur.dioxide +
                 total.sulfur.dioxide + density + pH + sulphates + alcohol,
               data = wine)
summary(model3)
```

Here is the output:

```
> summary(model3)

Re-fitting to get Hessian

Call:
polr(formula = quality ~ fixed.acidity + volatile.acidity + citric.acid +
    residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + pH + sulphates + alcohol, data = wine)

Coefficients:
                        Value Std. Error   t value
fixed.acidity          0.098341   0.068831   1.42874
volatile.acidity      -3.879626   0.544055  -7.13094
citric.acid            0.053926   0.660585   0.08163
residual.sugar         0.091284   0.054391   1.67830
chlorides             -6.706752   1.966230  -3.41097
free.sulfur.dioxide    0.004318   0.009655   0.44719
total.sulfur.dioxide  -0.003905   0.003244  -1.20383
density              -86.533740   1.361587 -63.55357
pH                    -1.523710   0.709843  -2.14654
sulphates              2.683266   0.457355   5.86692
alcohol                0.774485   0.078961   9.80847

Intercepts:
             Value    Std. Error t value
bad|average  -87.1941   1.3953   -62.4934
average|good -80.5404   1.4091   -57.1567

Residual Deviance: 1373.733
AIC: 1399.733
```

model4 is created after stepwise:

Here is the code:

```
model4 <- step(model3, scope=~., trace=FALSE)
summary(model4)
```

Here is the output:

```
> summary(model4)

Re-fitting to get Hessian

Call:
polr(formula = quality ~ volatile.acidity + chlorides + pH +
    sulphates + alcohol, data = wine)

Coefficients:
                  Value Std. Error t value
volatile.acidity -4.0170     0.4688  -8.569
chlorides        -6.5630     1.8940  -3.465
pH               -2.1087     0.5221  -4.039
sulphates         2.4779     0.4547   5.450
alcohol           0.8712     0.0758  11.494

Intercepts:
             Value   Std. Error t value
bad|average  -3.0044  1.7780    -1.6898
average|good  3.6146  1.7748     2.0366

Residual Deviance: 1377.604
AIC: 1391.604
```

Comments on difference: The residual deviance for multinomial model is 1296.663, and the residual deviance for ordinal model is 1377.604. So the residual deviance of ordinal model is greater than multinomial model, the AIC for residual deviance of ordinal model is also greater than multinomial model, which means model2 (multinomial model) should be more accurate than model4 (ordinal model). But there are more coefficients in multinomial model, which means this multinomial model is more complicated.

1.(c)

Here is the value predicted by multinomial model

Here is the code:

```
#1.(c)
new_wine <- list(fixed.acidity=7.9, volatile.acidity=0.5, citric.acid=0.26,
                 residual.sugar=2.25, chlorides=0.09, free.sulfur.dioxide=14,
                 total.sulfur.dioxide=36, density=0.997, pH=3.3, sulphates=0.63,
                 alcohol=10.2)
multi_predict <- predict(model2, newdata=new_wine, type="probs")
multi_predict
```

Here is the output:

```
> multi_predict
       bad    average       good
0.02143751 0.93821263 0.04034987
```

Conclusion: The probability of good wine is 0.04035

Here is the value predicted by ordinal model

Here is the code:

```
ordinal_predict <- predict(model4, newdata=new_wine, type="probs")
ordinal_predict
```

Here is the output:

```
> ordinal_predict
       bad    average       good
0.01995793 0.91852481 0.06151726
```

Conclusion: The probability of good wine is 0.06152.

1.(d)

1.cd)

We can prove the odds ratio in ordinal regression:

X_{ch} should defined as chlorides value, so

$$X_A = \begin{bmatrix} X_1 \\ X_{ch1} \\ X_2 \\ \vdots \\ X_n \end{bmatrix}, \quad X_B = \begin{bmatrix} X_1 \\ X_{ch2} \\ X_2 \\ \vdots \\ X_n \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_{ch} \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix},$$

$X_{ch1} = 0.07$

$X_{ch2} = 0.09$

$\beta_{ch} = -6.563$

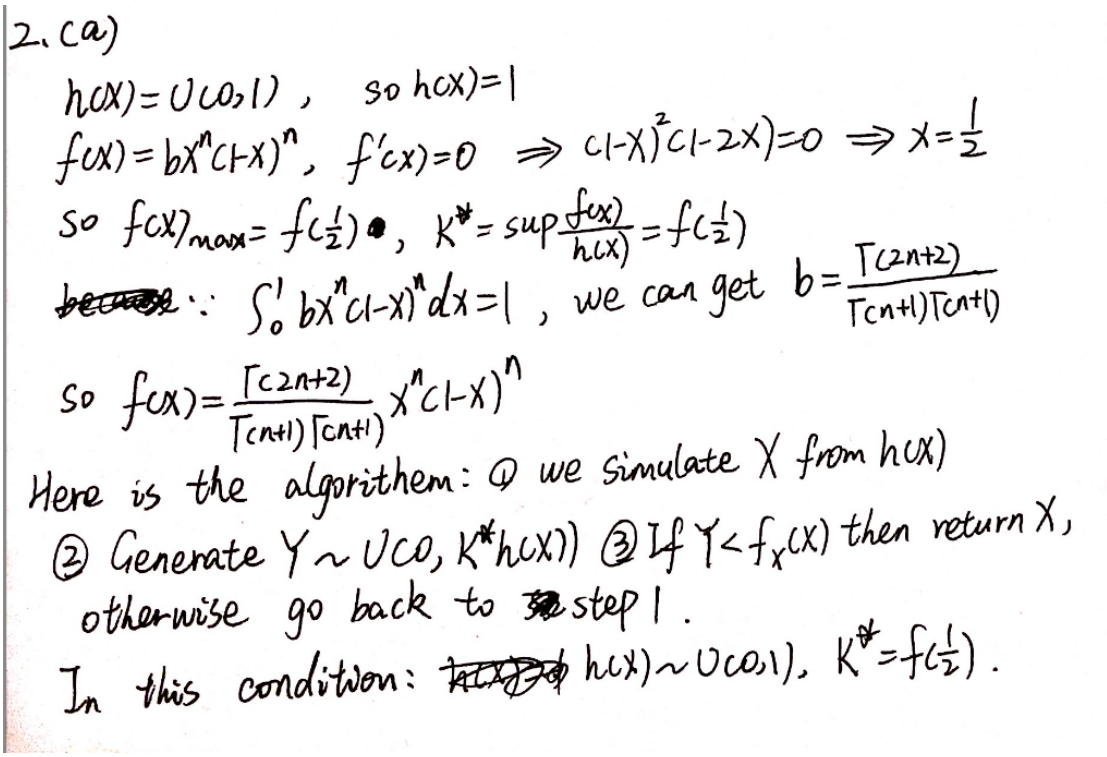
the odds ratio: $\dfrac{\frac{P(Y \le j | X_A)}{1 - P(Y \le j | X_A)}}{\frac{P(Y \le j | X_B)}{1 - P(Y \le j | X_B)}} = \dfrac{\exp(\text{logit}(P(Y \le j | X_A)))}{\exp(\text{logit}(P(Y \le j | X_B)))}$

$$= \frac{\exp(\theta_j - X_A^T \beta)}{\exp(\theta_j - X_B^T \beta)} = \exp(-(X_A - X_B)^T \beta)$$

so: $\exp(-(X_A - X_B)^T \beta) = \exp(0.02 \times (-6.563)) = 0.87699$

2.(a)

Here is the algorithm description:



Here is the code to build this function:

```
#2.(a)
#f(x)
beta_dis <- function(n, x) {
  b = factorial(2*n+1)/(factorial(n)^2)
  b*x^n*(1-x)^n
}
curve(beta_dis(3, x), 0, 1)
#function to generate samples
generate_point <- function(n){
  c <- beta_dis(3, 1/2)
  while(TRUE)
  {
    x <- runif(1, 0, 1)
    y <- runif(1, 0, c)
    if (y < beta_dis(n, x)) return(x)
  }
}
```

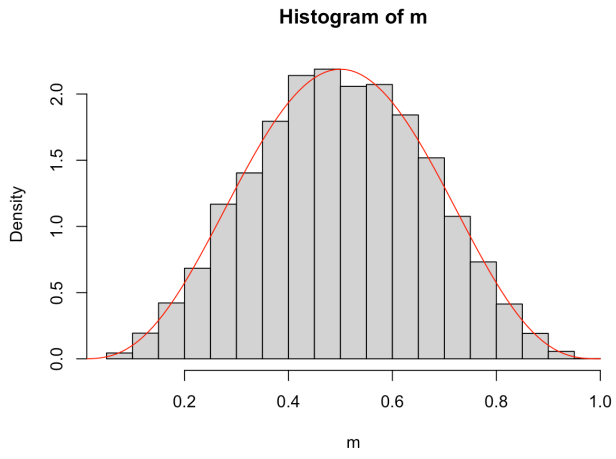
2.(b)

Here is the code, both algorithm and generating points, I use seed(7).

```r
#2.(a)
#f(x)
beta_dis <- function(n, x) {
  b = factorial(2*n+1)/(factorial(n)^2)
  b*x^n*(1-x)^n
}
curve(beta_dis(3, x), 0, 1)
#function to generate samples
generate_point <- function(n){
  c <- beta_dis(3, 1/2)
  while(TRUE)
  {
    x <- runif(1, 0, 1)
    y <- runif(1, 0, c)
    if (y < beta_dis(n, x)) return(x)
  }
}

#2.(b)
num <- 10000
m <- rep(0, num)
set.seed(7)
for (i in 1:num) {m[i] <- generate_point(3)}
hist(m, breaks=20, freq=FALSE)
curve(beta_dis(3, x), from=0, to=1, col="red", add=T)
```

Here is the graph:



4.

The code for PCA method:

```
#4
#(PCA method)
data(iris)
PCA_value <- prcomp(iris[1:4], scale=TRUE)
summary(PCA_value)
```

Here is the output:

```
> summary(PCA_value)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

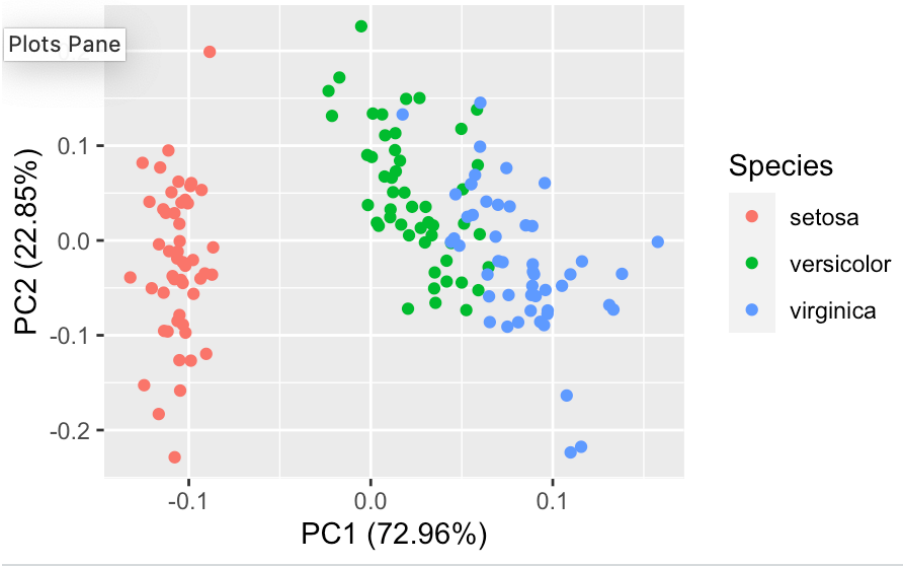
Here is the rotation value:

```
> PCA_value$rotation
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

Here is the code to plot when using PCA method:

```
library(ggplot2)
library(ggfortify)
autoplot(PCA_value)
autoplot(PCA_value, data=iris, colour="Species")
```

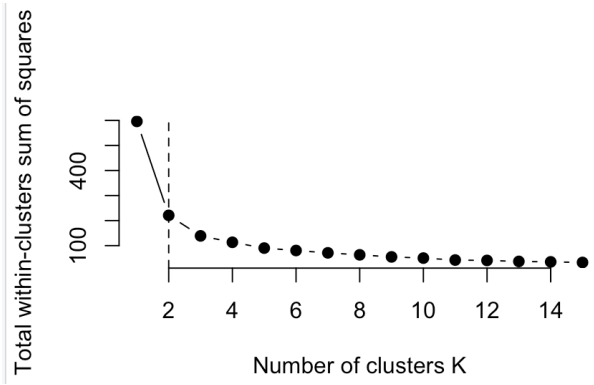
Here is the output plot:



Here is the code of the first method to determine K:

```
#K-means
#method(1) to define k
library(cluster)
k.max <- 15
iris_scale <- scale(iris[, 1:4], scale=TRUE)
wss <- sapply(1:k.max, function(k){kmeans(iris_scale, k, nstart=10)$tot.withinss})
plot(1:k.max, wss, type="b", pch=19, frame=FALSE,xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
abline(v=2, lty=2)
```

Here is the output graph:

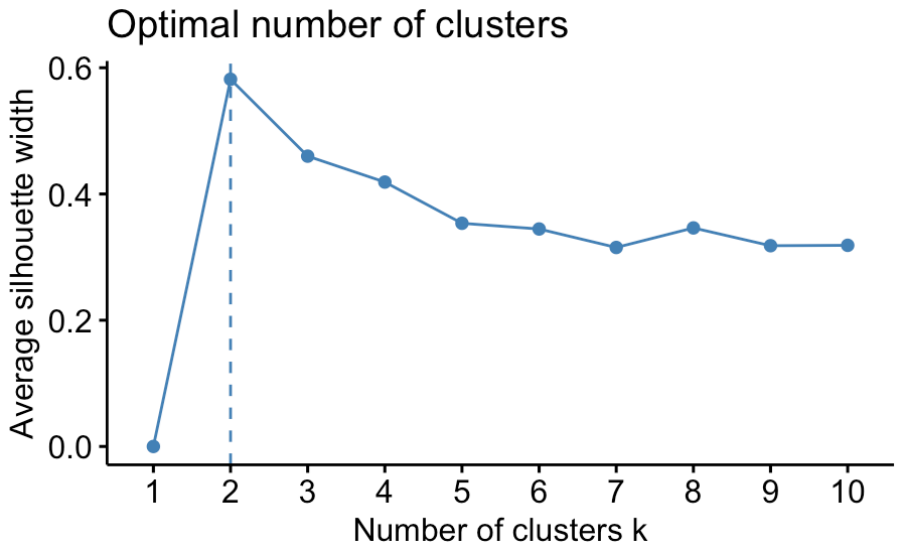


We choose K=2 by using this graph, because the decreased rate is much smaller after K=2.

Here is the code of the second method to determine K:

```
#method(2) to define k
library(factoextra)
library(NbClust)
library(fpc)
fviz_nbclust(iris_scale, kmeans, method=c("silhouette"))
```

Here is the output graph:

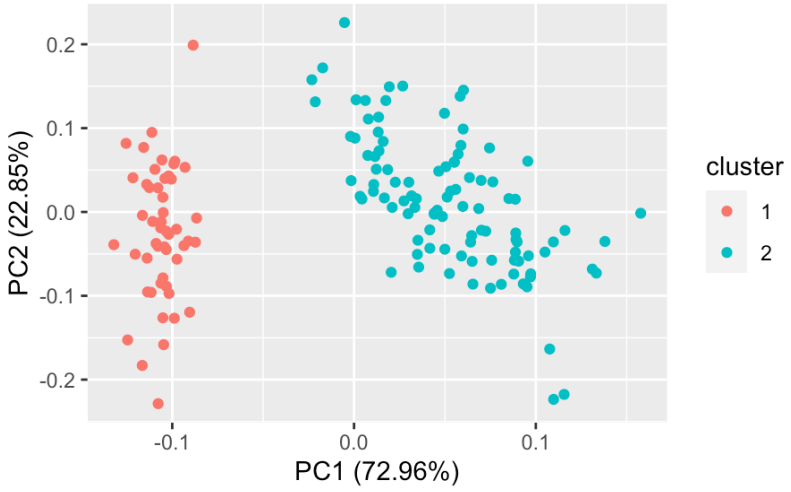


From this plot, we choose K=2, because it reaches the largest value.

Here is the plot code when using K-means method (K=2):

```
#use K-means
K.species <- kmeans(iris_scale, 2)
autoplot(K.species, data=iris_scale)
```

Here is the output plot:

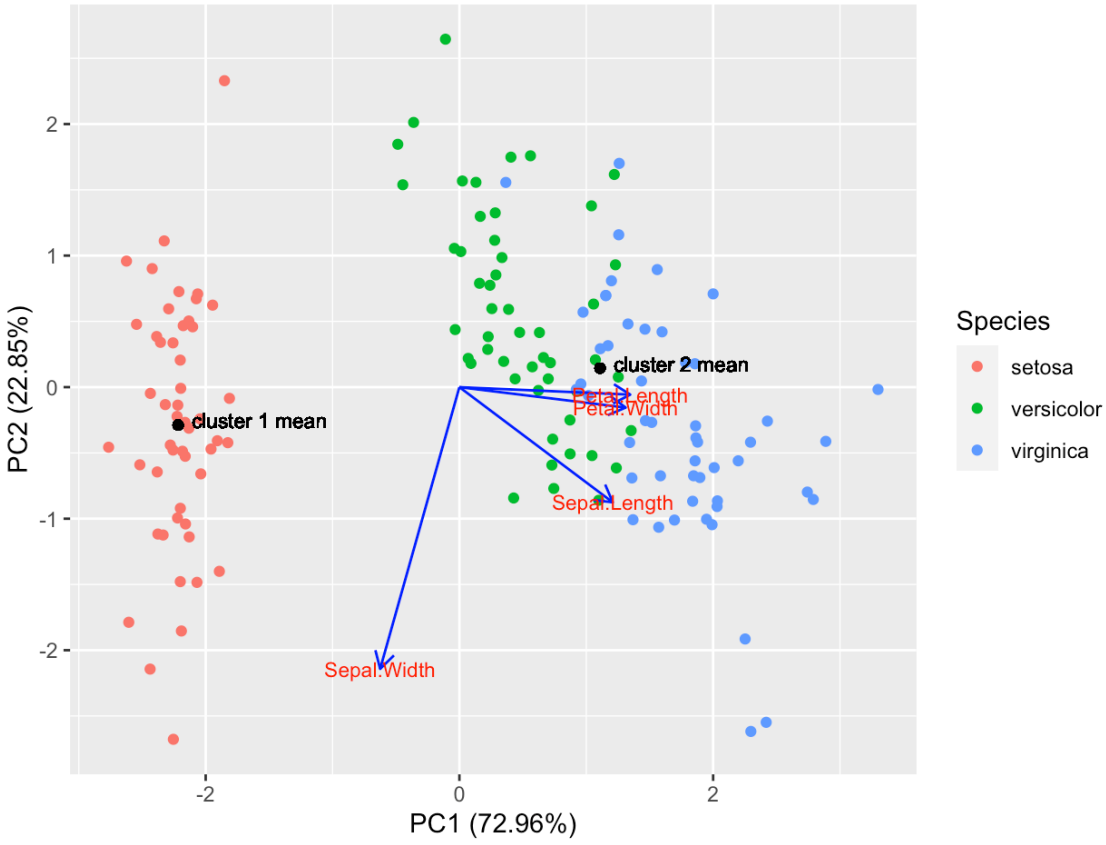


Here is the code to draw direction plot:

```
#direction graph
PCA_rotation <- prcomp(iris[,1:4], scale=TRUE, retx=TRUE)
x1 <- mean(PCA_rotation$x[,1][K.species$cluster==1])
y1 <- mean(PCA_rotation$x[,2][K.species$cluster==1])
x2 <- mean(PCA_rotation$x[,1][K.species$cluster==2])
y2 <- mean(PCA_rotation$x[,2][K.species$cluster==2])

autoplot(PCA_rotation, data = iris, colour = 'Species', loadings = TRUE,
         loadings.colour = 'blue', loadings.label = TRUE, loadings.label.size = 3,
         scale=0) +
  geom_point(x = x1,y = y1,colour = "black") +
  geom_text(label="cluster 1 mean",x = x1,y = y1,
            colour = "black",size=3,hjust=-0.1,vjust=0.2) +
  geom_point(x = x2,y = y2,colour = "black") +
  geom_text(label="cluster 2 mean",x = x2,y = y2,
           colour = "black",size=3,hjust=-0.1,vjust=0.2)
```

Here is the output plot:



Comment: The clusters created by K-means method shows that the length and width figures can be used to determine the species. We can also see that there is a large distance, which means large difference from setosa and other species. But the distance between versicolor and virginica is much smaller, which means it should be much harder to distinguish versicolor and virginica by using the K-means method.

3.(a)

3. (a)

for y_i we have $y_i \sim N(\beta_0+\beta_1 x_i, \sigma^2)$

$$p(y|\beta_0,\beta_1,\sigma^2)=\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{[y_i-(\beta_0+\beta_1x_i)]^2}{2\sigma^2}\right)$$

$$p(\beta_0)=\frac{1}{\sqrt{2\pi\sigma_\beta^2}}\exp\left(-\frac{\beta_0^2}{2\sigma_\beta^2}\right),\quad p(\beta_1)=\frac{1}{\sqrt{2\pi\sigma_\beta^2}}\exp\left(-\frac{\beta_1^2}{2\sigma_\beta^2}\right)$$

$$p(\sigma^2)=\frac{b^a}{\Gamma(a)}x^{-a-1}\exp\left(-\frac{b}{x}\right)$$

the posterior distribution:

$$p(\beta_0|y,\beta_1,\sigma^2)\propto p(y|\beta_0,\beta_1,\sigma^2)\cdot p(\beta_0)$$

$$=\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{[y_i-(\beta_0+\beta_1x_i)]^2}{2\sigma^2}\right)\cdot\frac{1}{\sqrt{2\pi\sigma_\beta^2}}\exp\left(-\frac{\beta_0^2}{2\sigma_\beta^2}\right)$$

$$\propto\exp\left(-\frac{1}{2\sigma^2}\sum(y_i^2-2y_i\beta_0-2y_i\beta_1x_i+\beta_0^2+\beta_1^2x_i^2+2\beta_0\beta_1x_i)-\frac{\beta_0^2}{2\sigma_\beta^2}\right)$$

$$\propto\exp\left(-\frac{1}{2\sigma^2}\sum(-2y_i\beta_0+2\beta_0\beta_1x_i+\beta_0^2)-\frac{\beta_0^2}{2\sigma_\beta^2}\right)$$

$$=\exp\left(\frac{\beta_0}{\sigma^2}\sum(y_i-\beta_1x_i)-\frac{1}{2}\left(\frac{n\beta_0^2}{\sigma^2}-\frac{\beta_0^2}{\sigma_\beta^2}\right)\right)$$

$$=\exp\left(-\frac{1}{2}\beta_0^2\left(\frac{n}{\sigma^2}-\frac{1}{\sigma_\beta^2}\right)+\frac{\beta_0}{\sigma^2}\sum(y_i-\beta_1x_i)\right)$$

so $-\frac{1}{2\sigma_m^2}=-\frac{1}{2}\left(\frac{n}{\sigma^2}-\frac{1}{\sigma_\beta^2}\right)\Rightarrow\sigma_m^2=\left(\frac{n}{\sigma^2}-\frac{1}{\sigma_\beta^2}\right)^{-1}$

$$\frac{\mu_m}{\sigma_m^2}=\frac{1}{\sigma^2}\sum(y_i-\beta_1x_i)\Rightarrow\mu_m=\frac{\sum(y_i-\beta_1x_i)}{\sigma^2}\cdot\left(\frac{n}{\sigma^2}-\frac{1}{\sigma_\beta^2}\right)^{-1}$$

so $\beta_0\sim N\left(\frac{\sum(y_i-\beta_1x_i)}{\sigma^2}\cdot\left(\frac{n}{\sigma^2}-\frac{1}{\sigma_\beta^2}\right)^{-1},\left(\frac{n}{\sigma^2}-\frac{1}{\sigma_\beta^2}\right)^{-1}\right)$

$$P(\beta_1 | y, \beta_0, \sigma^2) \propto P(y | \beta_0, \beta_1, \sigma^2) \cdot P(\beta_1)$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{[y_i - (\beta_0 + \beta_1 x_i)]^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{2\pi\sigma_\beta^2}} \exp\left(-\frac{\beta_1^2}{2\sigma_\beta^2}\right)$$

$$\propto \exp\left(-\frac{1}{2\sigma^2}\sum(-2y_i\beta_1 x_i + \beta_1^2 x_i^2 + 2\beta_0\beta_1 x_i) - \frac{\beta_1^2}{2\sigma_\beta^2}\right)$$

$$= \exp\left[\left(-\frac{\sum x_i^2}{2\sigma^2} - \frac{1}{2\sigma_\beta^2}\right)\beta_1^2 + \frac{1}{\sigma^2}\left(\sum y_i x_i - \beta_0 x_i\right)\beta_1\right]$$

$$-\frac{1}{2\sigma_1^2} = -\left(\frac{\sum x_i^2}{2\sigma^2} + \frac{1}{2\sigma_\beta^2}\right) \Rightarrow \sigma_1^2 = \left(\frac{\sum x_i^2}{\sigma^2} + \frac{1}{\sigma_\beta^2}\right)^{-1},$$

$$\frac{\mu_1}{\sigma_1^2} = \frac{\sum(y_i - \beta_0)x_i}{\sigma^2} \Rightarrow \mu_1 = \frac{\sum(y_i - \beta_0)x_i}{\sigma^2}\left(\frac{\sum x_i^2}{\sigma^2} + \frac{1}{\sigma_\beta^2}\right)^{-1}, \text{ so } \beta_1 \sim N\left(\frac{\sum(y_i - \beta_0)x_i}{\sigma^2}\left(\frac{\sum x_i^2}{\sigma^2} + \frac{1}{\sigma_\beta^2}\right)^{-1}, \left(\frac{\sum x_i^2}{\sigma^2} + \frac{1}{\sigma_\beta^2}\right)^{-1}\right)$$

$$P(\sigma^2 | \beta_0, \beta_1, y_i) \propto P(y_i | \beta_0, \beta_1, \sigma^2) \cdot P(\sigma^2)$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{[y_i - (\beta_0 + \beta_1 x_i)]^2}{2\sigma^2}\right) \cdot \frac{b^a}{\Gamma(a)} (\sigma^2)^{-a-1} \exp\left(-\frac{b}{\sigma^2}\right)$$

$$\propto (\sigma^2)^{-\frac{1}{2}n} \cdot (\sigma^2)^{-a-1} \cdot \exp\left(-\frac{\sum[y_i - \beta_0 - \beta_1 x_i]^2}{2\sigma^2} - \frac{b}{\sigma^2}\right)$$

$$= (\sigma^2)^{-(\frac{1}{2}n + a + 1)} \exp\left[-\frac{1}{\sigma^2}\left[\frac{\sum(y_i - \beta_0 - \beta_1 x_i)^2}{2} + b\right]\right]$$

$$\text{so } \sigma^2 \sim IG\left(\frac{1}{2}n + a, \frac{\sum(y_i - \beta_0 - \beta_1 x_i)^2}{2} + b\right)$$