Required R packages and Directories

Problem 1: Customer Segmentation with RFM (Recency, Frequency, and Monetary Value)

Problem 2: Poisson Mixture Model

Homework #6: Clustering

Hyunsuk Ko

Due: Wed Oct 19 | 11:45am

DS 6030 | Fall 2021 | University of Virginia

This is an **independent assignment**. Do not discuss or work with classmates.

Required R packages and Directories

```
data.dir = 'https://mdporter.github.io/DS6030/data/' # data directory
library(R6030)     # functions for DS-6030
library(tidyverse) # functions for data manipulation
library(mclust)    # functions for mixture models
library(mixtools)  # poisregmixEM() function
library(broom)
```

Problem 1: Customer Segmentation with RFM (Recency, Frequency, and Monetary Value)

RFM analysis is an approach that some businesses use to understand their customers' activities. At any point in time, a company can measure how recently a customer purchased a product (Recency), how many times they purchased a product (Frequency), and how much they have spent (Monetary Value). There are many ad-hoc attempts to segment/cluster customers based on the RFM scores (e.g., here is one based on using the customers' rank of each dimension independently: https://joaocorreia.io/blog/rfm-analysis-increase-sales-by-segmenting-your-customers.html (https://joaocorreia.io/blog/rfm-analysis-increase-sales-by-segmenting-your-customers.html)). In this problem you will use the clustering methods we covered in class to segment the customers.

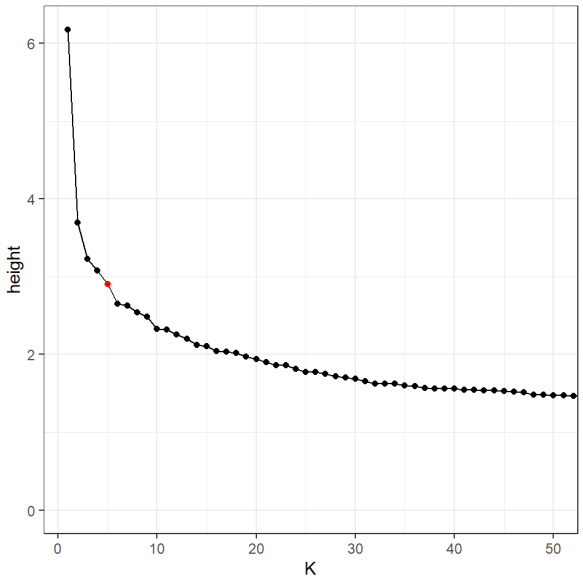
The data for this problem can be found here: https://mdporter.github.io/DS6030/data//RFM.csv (https://mdporter.github.io/DS6030/data//RFM.csv). Cluster based on the Recency, Frequency, and Monetary value columns.

```
rfm = read.csv("RFM.csv")
```

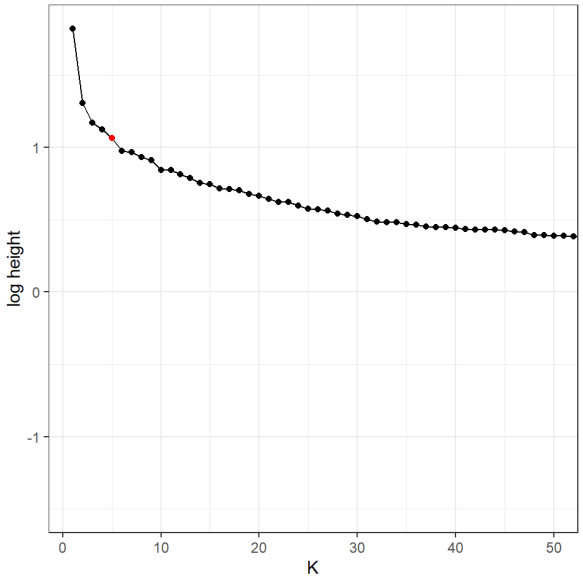
a. Implement hierarchical clustering.

- Describe any pre-processing steps you took (e.g., scaling, distance metric)
- State the linkage method you used with justification.
- Show the resulting dendrogram
- State the number of segments/clusters you used with justification.
- Using your segmentation, are customers 1 and 100 in the same cluster?

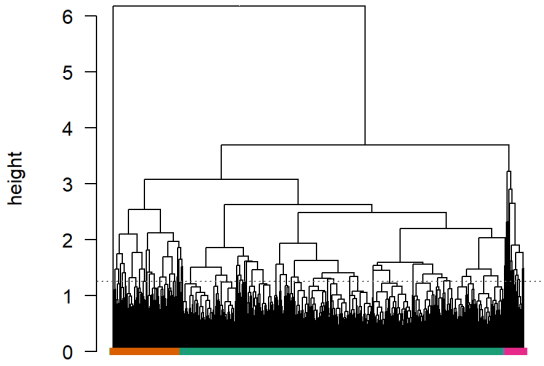
```
X = rfm %>% scale() %>% as_tibble()

dX = dist(X, method = "euclidean")
hc = hclust(dX, method = "average") # use average linkage

tibble(height = hc$height, K = row_number(-height)) %>%
  ggplot(aes(K, height)) +
  geom_line() +
  geom_point(aes(color = ifelse(K == 5, "red", "black"))) +
  scale_color_identity() +
  coord_cartesian(xlim = c(1, 50))
```




```
tibble(height = log(hc$height), K = row_number(-height)) %>%
  ggplot(aes(K, height)) +
  geom_line() +
  geom_point(aes(color = ifelse(K == 5, "red", "black"))) +
  scale_color_identity() +
  coord_cartesian(xlim = c(1, 50), ylim = c(-1.5, NA)) +
  labs(y = "log height")
```



```
colPalette = c("#1b9e77", "#d95f02", "#7570b3", "#e7298a", "#66a61e")
clusters = cutree(hc, k = 5)
plot(as.dendrogram(hc), las = 1, leaflab ="none", ylab = "height")
ord = hc$order
labels = clusters[ord]
colors = colPalette[labels]
shapes = 15
n = length(labels)
points(1:n, rep(0,n), col = colors, pch = shapes, cex=.8)
abline(h = 1.25, lty = 3, col = "grey40")
```




```
clusters[1]
```

```
#> [1] 1
```

```
clusters[100]
```

```
#> [1] 1
```

Customer 1 and 100 belong to the same cluster.

b. Implement k-means.

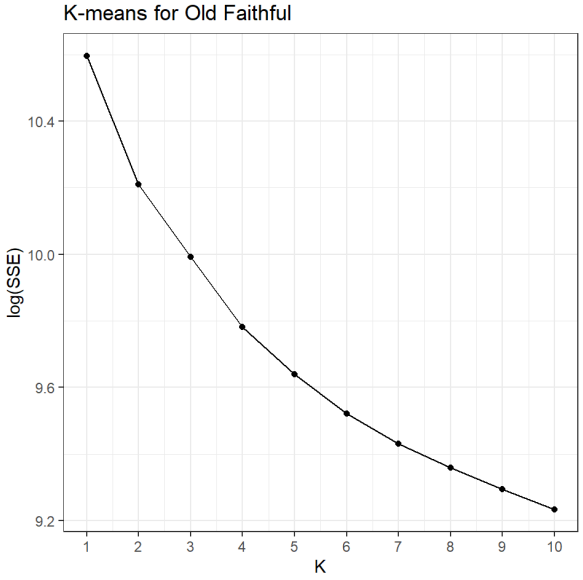
- Describe any pre-processing steps you took (e.g., scaling)
- State the number of segments/clusters you used with justification.
- Using your segmentation, are customers 1 and 100 in the same cluster?

```
#-- Run kmeans for multiple K
Kmax = 10  # maximum K
SSE = numeric(Kmax) # initiate SSE vector
set.seed(2022) # set seed for reproducibility
for(k in 1:Kmax){
  km = kmeans(X, centers=k, nstart=25) # use 25 initializations
  SSE[k] = km$tot.withinss # get SSE
  }
```

```
#> Warning: did not converge in 10 iterations
```

```
#> Warning: Quick-TRANSfer stage steps exceeded maximum (= 500000)
```

```
#-- Plot results
tibble(K = 1:Kmax, SSE) %>%
  ggplot(aes(K, log(SSE))) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = 1:Kmax) +
  labs(title = "K-means for Old Faithful")
```




```
k_result = X %>%
  kmeans(centers = 4, nstart = 100)
```

```
#> Warning: Quick-TRANSfer stage steps exceeded maximum (= 500000)

#> Warning: Quick-TRANSfer stage steps exceeded maximum (= 500000)

#> Warning: Quick-TRANSfer stage steps exceeded maximum (= 500000)
```

```
k_result$cluster[1]
```

```
#> [1] 4
```

```
k_result$cluster[100]
```

```
#> [1] 2
```

Customer 1 and 100 belong to the different cluster.

c. Implement model-based clustering

- Describe any pre-processing steps you took (e.g., scaling)
- State the number of segments/clusters you used with justification.
- Describe the best model. What restrictions are on the shape of the components?
- Using your segmentation, are customers 1 and 100 in the same cluster?

```
mix = Mclust(X, verbose = FALSE) # fit series of models

result = augment(mix, X)
result
```

id <dbl>	Recency <dbl>	Frequency <dbl>	Monetary <dbl>	.class <fct>	.uncertainty <dbl>
-1.7317910	-0.323069	1.01602	0.6100067	1	1.192e-01
-1.7314446	2.006845	0.05935	0.0095462	2	1.572e-03
-1.7310982	-0.133953	-0.89733	-0.7092743	1	1.467e-01
-1.7307518	-0.466798	0.37824	0.7370271	1	1.301e-01
-1.7304054	0.236716	1.01602	0.1740954	1	1.233e-01
-1.7300591	-0.534880	-0.25955	0.2000769	1	1.415e-01
-1.7297127	0.788935	1.01602	1.0228232	1	3.286e-01
-1.7293663	0.009776	-0.57844	-0.0799455	1	1.323e-01
-1.7290199	0.622513	-1.53511	-1.2029221	1	3.893e-01
-1.7286735	-0.126388	-0.89733	0.0326408	1	1.243e-01

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

```
#tidy(mix)

glance(mix) # best model
```

model <chr>	G <int>	BIC <dbl>	logLik <dbl>	df <dbl>	hypvol <dbl>	nobs <int>
VVE	8	-93486	-46388	77	*NA*	10000

1 row

```
result$.class[1]
```

```
#> [1] 1
#> Levels: 1 2 3 4 5 6 7 8
```

```
result$.class[100]
```

```
#> [1] 1
#> Levels: 1 2 3 4 5 6 7 8
```

Customer 1 and 100 belong to the same cluster.

d. Discuss how you would cluster the customers if you had to do this for your job. Do you think one model would do better than the others?

Since clustering is unsupervised learning, it has no one definitive best model. That means, we cannot directly compare the performance of one model to that of the others.

Problem 2: Poisson Mixture Model

The pmf of a Poisson random variable is:

$$f_k(x;\lambda_k) = \frac{\lambda_k^x e^{-\lambda_k}}{x!}$$

A two-component Poisson mixture model can be written:

$$f(x;\theta) = \pi\frac{\lambda_1^x e^{-\lambda_1}}{x!} + (1-\pi)\frac{\lambda_2^x e^{-\lambda_2}}{x!}$$

a. What are the parameters of the model?

$\theta = (\lambda_1, \lambda_2, \pi)$

b. Write down the log-likelihood for n independent observations $(x_1, x_2, \ldots, x_n)$.

$\Delta \in 0, 1, Pr(\Delta = 1) = \pi$

$\lambda = (1-\Delta) * \lambda_1 + \Delta * \lambda_2$

$logL = \sum_{k=1}^{n}[(1-\Delta_k) * log(\pi\frac{\lambda_k^x e^{-\lambda_k}}{x!}\lambda_k) + \Delta_k * log(\pi\frac{\lambda_k^x e^{-\lambda_k}}{x!}\lambda_k)] + \sum_{k=1}^{n}[(1-\Delta_k) * log(1-\pi) + \Delta_k * log\pi]$

c. Suppose we have initial values of the parameters. Write down the equation for updating the *responsibilities*.

Update r_{ik}, using θ

$$r_{ik} = Pr(g_i = k|D, \theta) = \frac{P(D|g_i = k, \theta_k)\pi_k}{\sum_{j=1}^{k} P(D|g_i = j, \theta_j)\pi_j}$$

d. Suppose we have responsibilities, r_{ik} for all $i = 1, 2, \ldots, n$ and $k = 1, 2$. Write down the equations for updating the parameters.

$\hat{\lambda_1} = \frac{\sum_{i=1}^{n}(1-r_{i1}\lambda_1)}{\sum_{i=1}^{n}(1-r_{i1})}$

$\hat{\lambda_2} = \frac{\sum_{i=1}^{n}r_{i2}\lambda_1}{\sum_{i=1}^{n}r_{i2}}$

$\pi = \frac{\sum_{k=1}^{2}\sum_{i=1}^{n}r_{ik}}{N}$

e. Fit a two-component Poisson mixture model, report the estimated parameter values, and show a plot of the estimated mixture pmf for the following data:

```
#-- Run this code to generate the data
set.seed(123)             # set seed for reproducibility
n = 200                   # sample size
z = sample(1:2, size=n, replace=TRUE, prob=c(.25, .75)) # sample the latent class
theta = c(8, 16)          # true parameters: lambda_1, lambda_2
y = ifelse(z==1, rpois(n, lambda=theta[1]), rpois(n, lambda=theta[2]))
x = rep(1, length(y))
```

- Note: The function `poisregmixEM()` in the R package `mixtools` is designed to estimate a mixture of *Poisson regression* models. We can still use this function for our problem of pmf estimation if it is recast as an intercept-only regression. To do so, set the x argument (predictors) to `x = rep(1, length(y))` and `addintercept = FALSE`.
  - Look carefully at the output from this model. The `beta` values (regression coefficients) are on the log scale.

```
pois_result = poisregmixEM(y, x, lambda = NULL, beta = NULL, k = 2,
             addintercept = FALSE, epsilon = 1e-08,
             maxit = 10000, verb = FALSE)
```

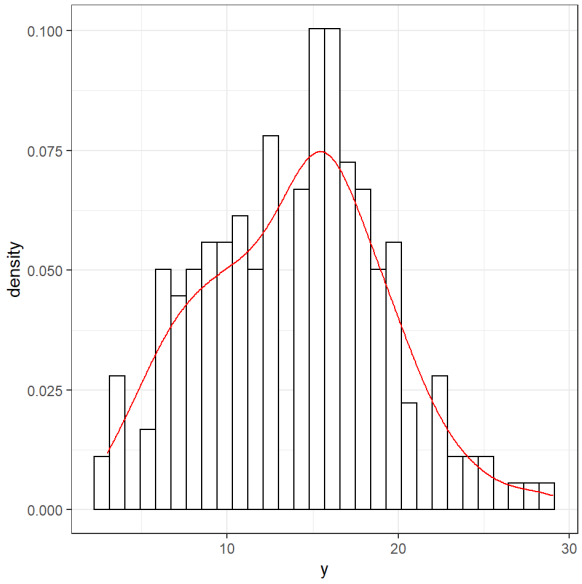
```
#> number of iterations= 46
```

```
pois_result$lambda # parameters
```

```
#> [1] 0.7283 0.2717
```

```
df = tibble(y = y)

ggplot(df, aes(x = y)) +
  geom_histogram(aes(y = ..density..), colour = 1, fill = "white") +
  geom_density(colour = "red")
```

```
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



f. **2 pts Extra Credit**: Write a function that estimates this two-component Poisson mixture model using the EM approach. Show that it gives the same result as part *e*.

- Note: you are not permitted to copy code. Write everything from scratch and use comments to indicate how the code works (e.g., the E-step, M-step, initialization strategy, and convergence should be clear).
- Cite any resources you consulted to help with the coding.

ADD SOLUTION HERE