Statistical Computing Midterm

作者: 109024701 林承慶

Introduction

這個資料集是英國的網路零售商店從2010/12/1到2011/12/9的每一筆交易資料。這家網路商店主要是賣禮品，而買方大多是批發商。這篇報告的目標是要利用每個顧客的交易資料，來建立RFM模型，並且分出屬於此網路商店的目標客群。

所謂RFM模型是指，利用三種變數，去分類客人的方法。其中的三種變數為:

1. Recency: 最近一次的消費。數字越大，則越有可能流失客人。
2. Frequency: 消費頻率。數字越大，則越有可能是常客。
3. Monetary: 消費金額。數字越大，則這家店的收入是源自於這個客人的比例越大。

傳統實務上，會利用這三個指標，依照經驗，切分客群。但這個報告就是要利用cluster的方式把客人分群，我所使用的cluster方法為GMM以及K-means。

資料集

我所用的資料集是有9個變數和541909資料筆數。其中所包含的變數有:

變數名稱	變數內容
InvoiceNo	每一筆交易編號
StockCode	商品編號
Description	商品名稱
Quantity	單一商品購買數量
InvoiceDate	交易日期
UnitPrice	單一商品價格
CustomerID	買家編號
Country	買家所屬國家

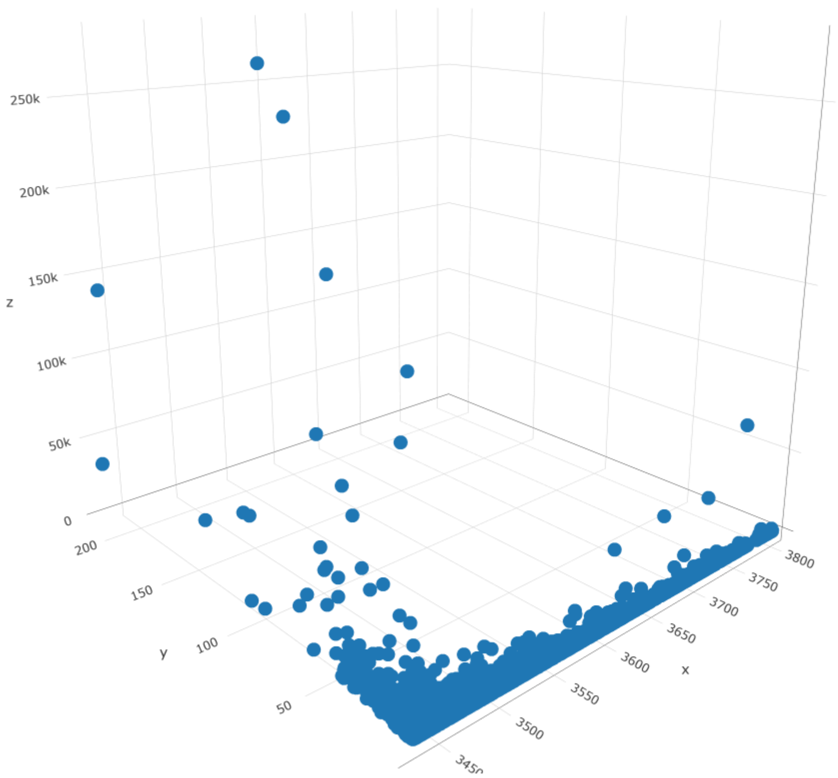
資料前處理

重新製造了我所要的變數，把資料改成單一買家為一個單位，並且去除了沒有買家編號的資料

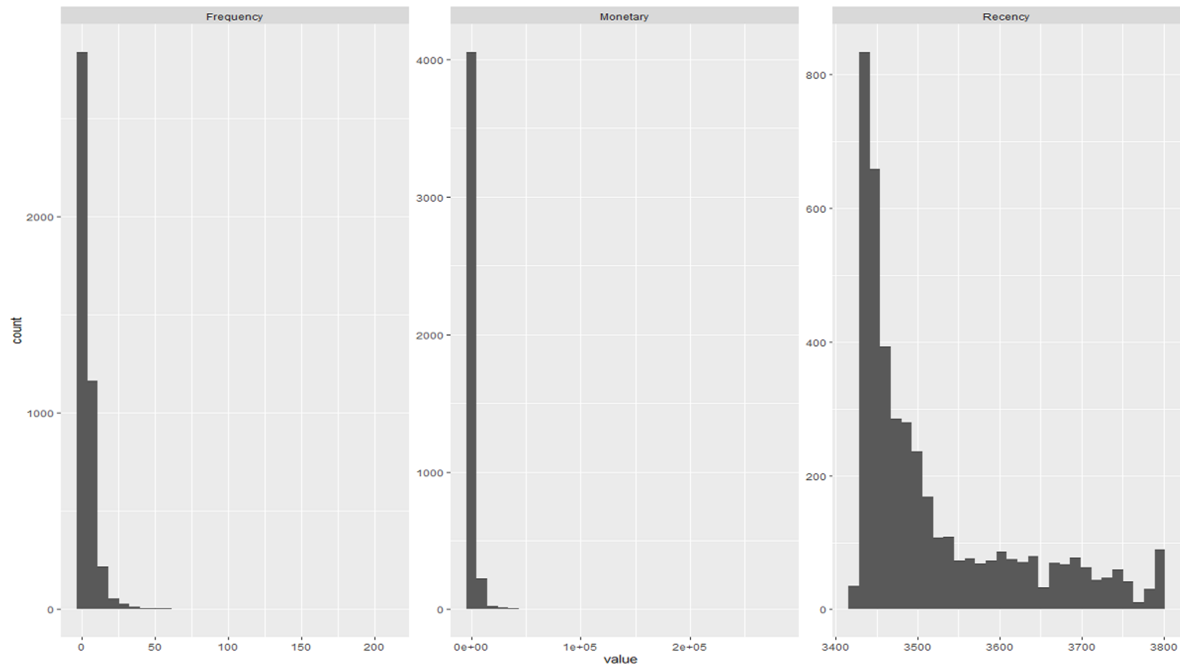
- Recency : 利用今天的日期減去交易日期(InvoiceDate)，並找到最短的間隔。
- Frequency : 在相同的買家編號(CustomerID)下，計算有多少不同的交易編號(InvoiceNo)。
- Monetary : 把商品數量乘上商品單價(Quantity×UnitPrice)，在單一買家下加總起來，並只保留了大於0的資料 。

EDA

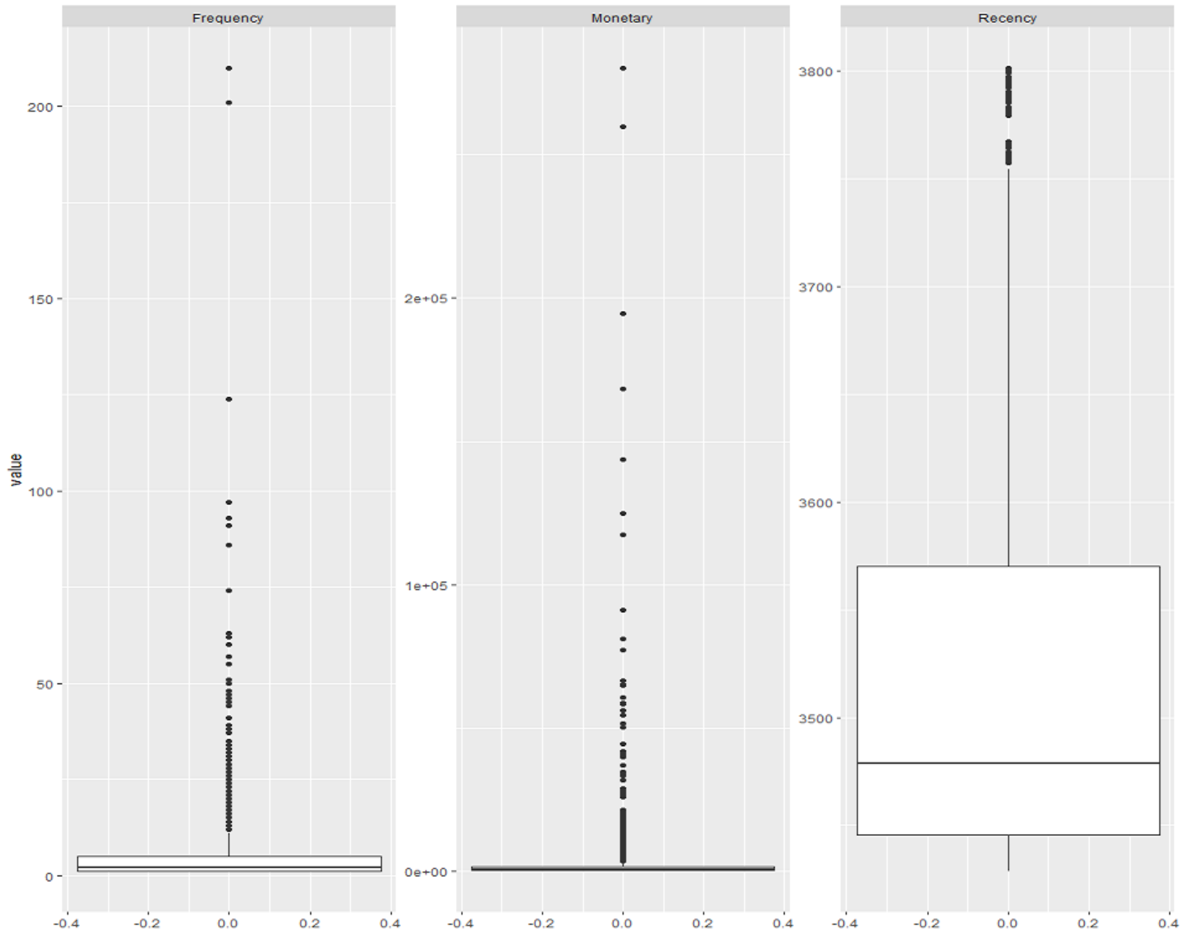
首先，我們來看3D散佈圖:



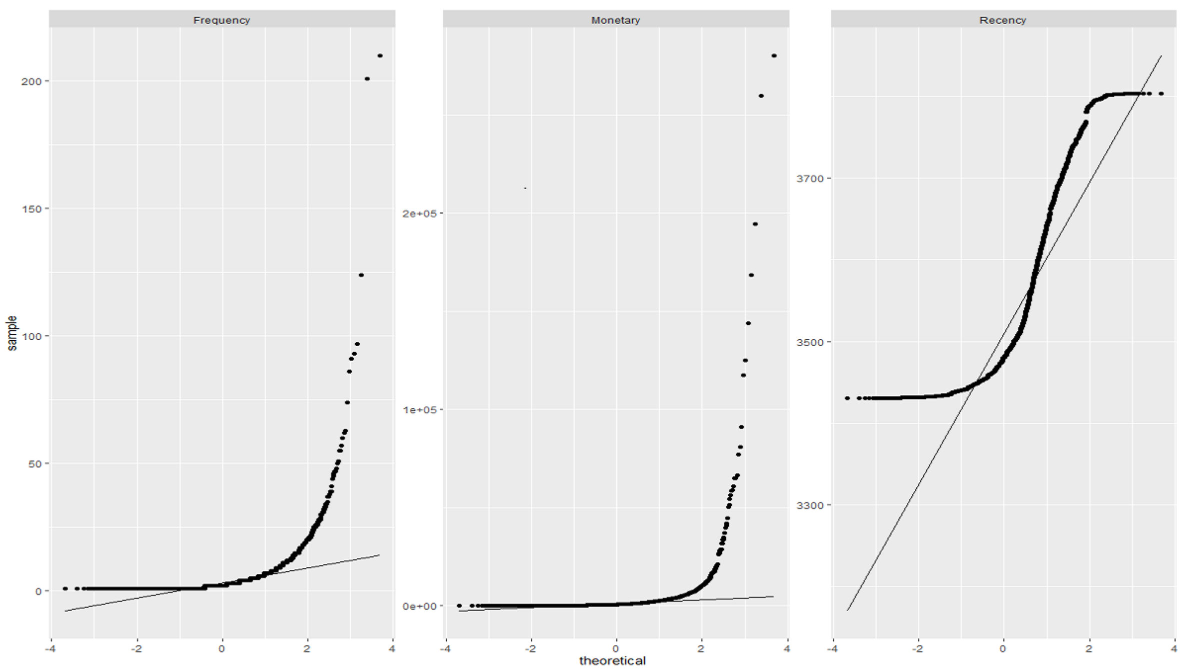
可以看到，資料大部分集中在Recency(x軸附近)，也就是對於Frequency(y軸)和Monetary(z軸)，是集中在0點附近。再來，我們可以看到原始資料的狀況的histogram



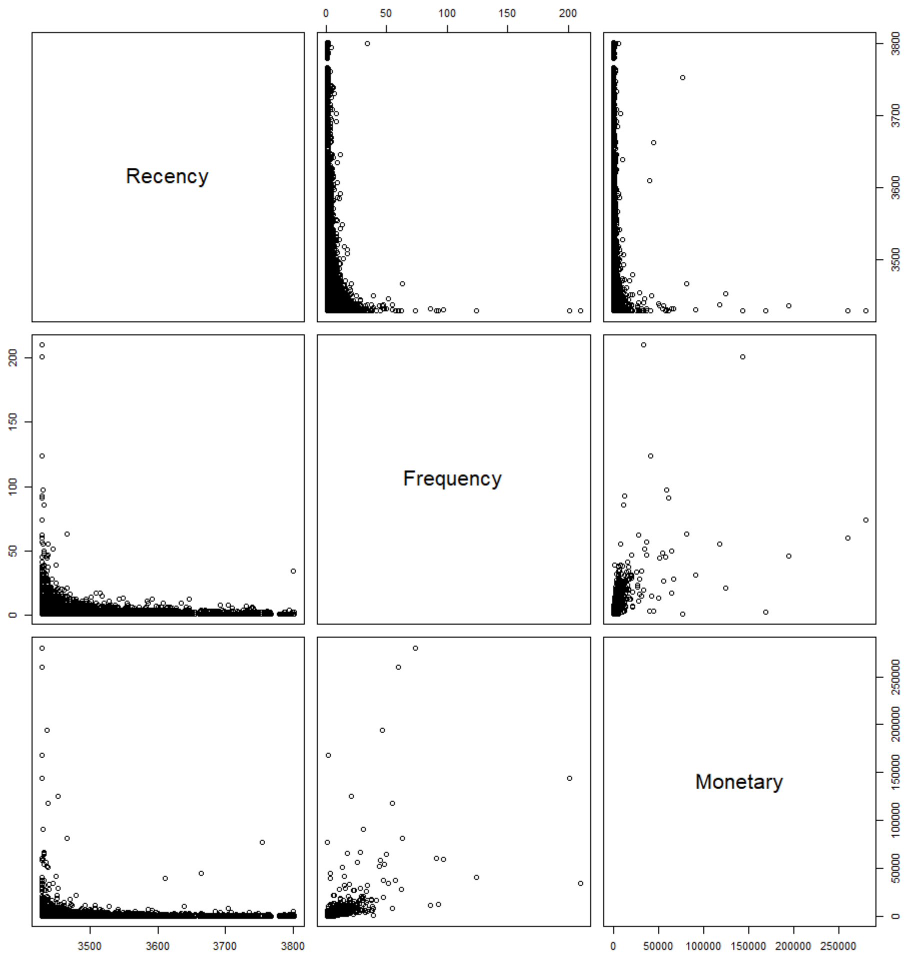
可以看到在histogram中，Frequency和Monetary這兩個變數大部分集中在左側，但是同時又有少量極大的值存在。而Recency則是類似於Exponential 的分布。再來使用boxplot觀察。



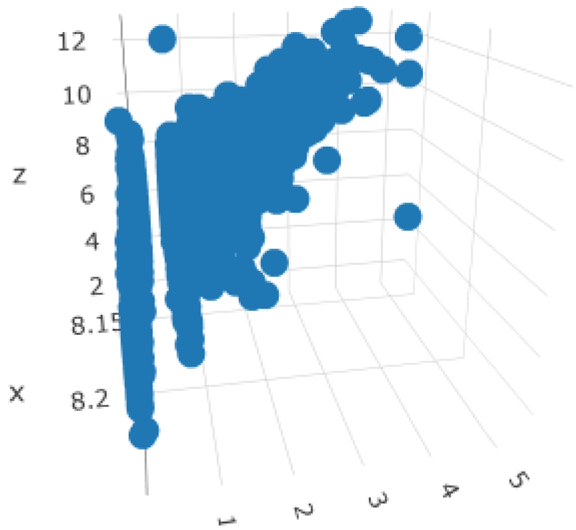
從boxplot中，能看出來Frequency和Monetary的outlier很多而且都很大，所以應該是要做資料轉換，使得資料分布不是大多集中在一塊。由於我們接著繼續觀察QQplot。



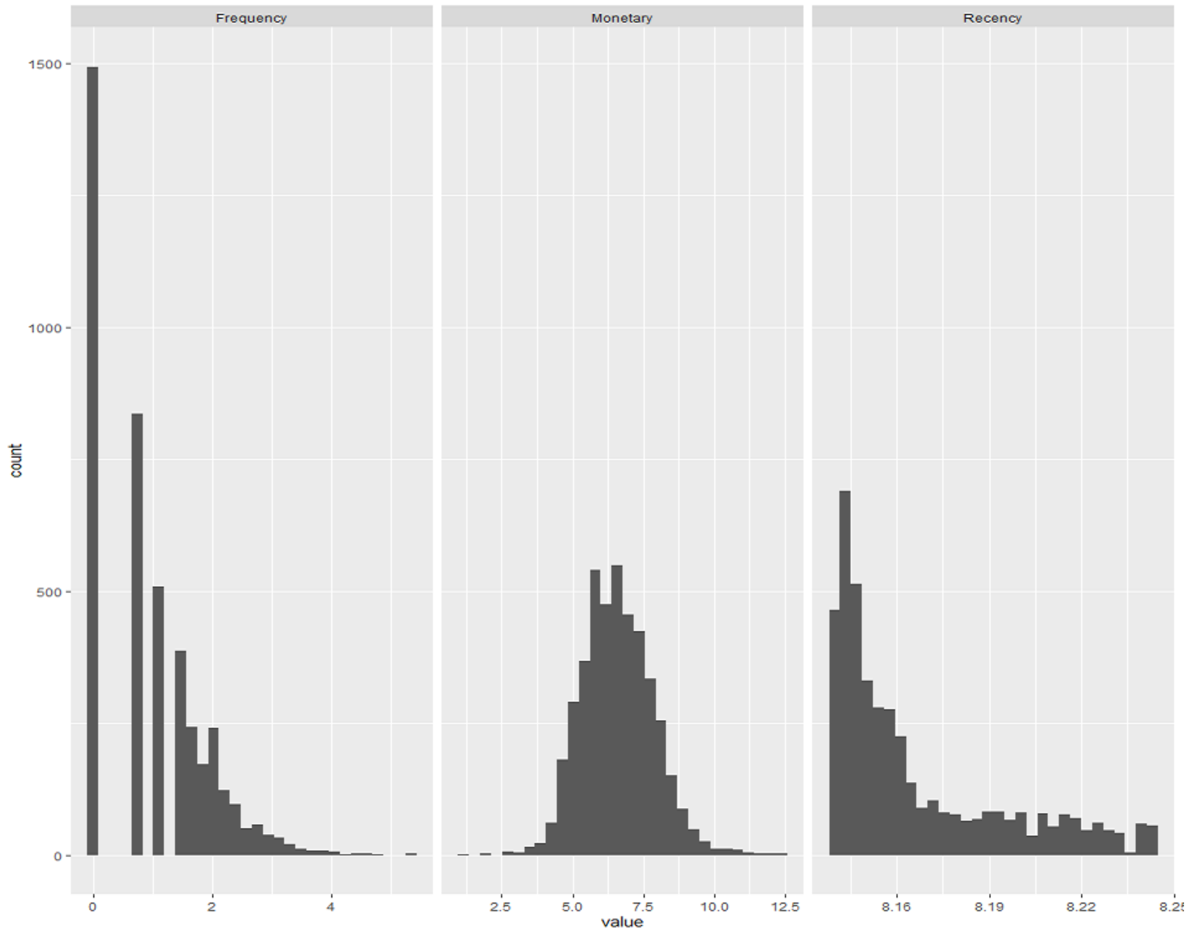
QQplot可以很明顯發現，所有的變數都並不符合常態分布，其中Frequency和Monetary為嚴重左偏，然而Recency則是過度集中。最後，我們觀察pairplot。



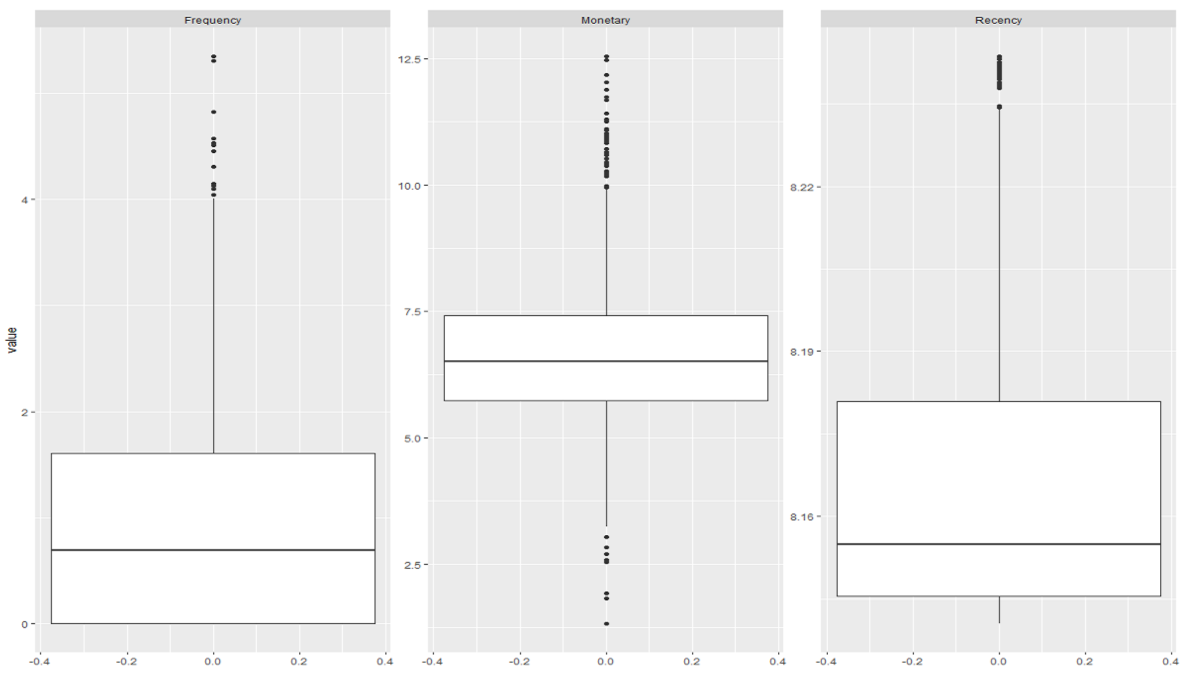
最後，從pairplot可以看到，大部分的資料都是集中在x軸、y軸、或是原點的附近，而其他的就是廣布在各處。因此，這不能個別處理outlier，而是必須要做資料轉換，又因為我看到以上三張圖的狀況，我認為應該要對資料取log。以下是取完log之後的變化:



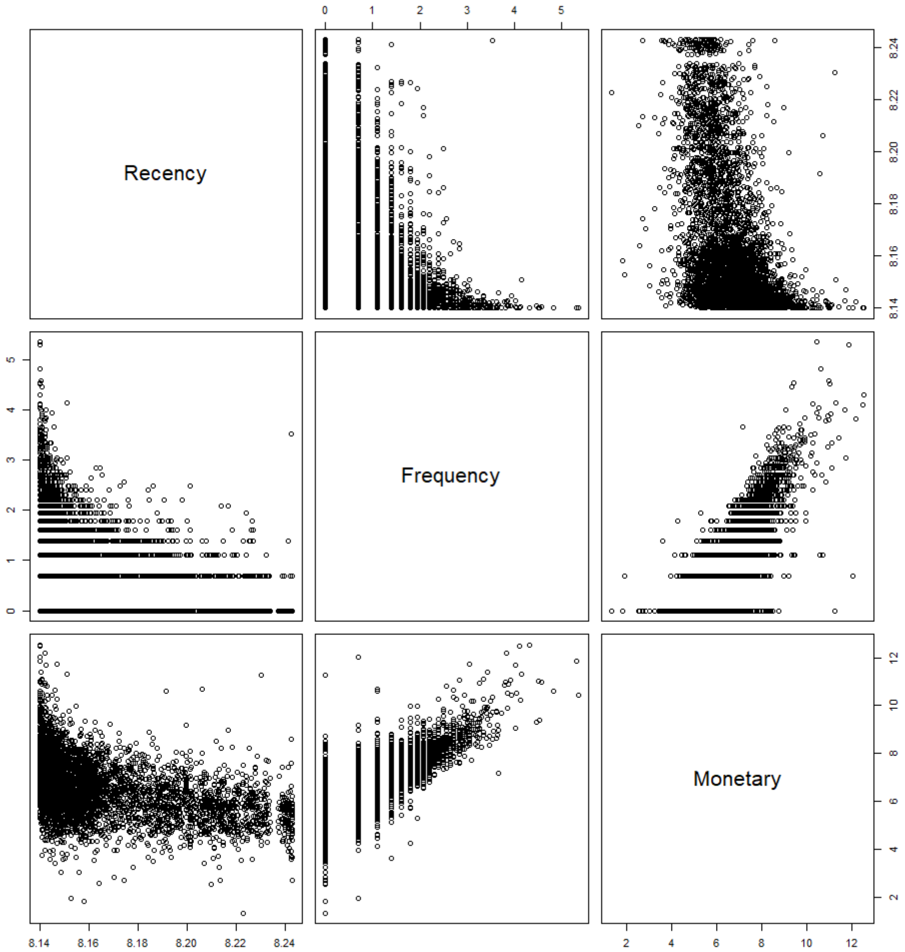
在做資料轉換之後，我們可以看到資料點在Frequency(y軸)方向上，分出了明顯的層次，而後面則有一段資料散佈。總之資料相較於原始資料，資料點不再那麼的集中。再來，我們來看histogram



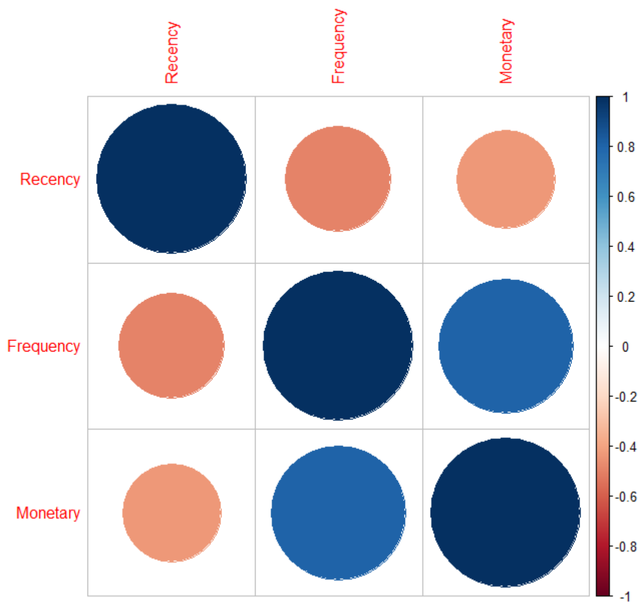
我們可以看到，Monetary變成很接近normal分布，Recency分布變化不大，而Frequency則是被拉開來看，並且接近左側的三個值分別就是來1次、2次及3次的量。



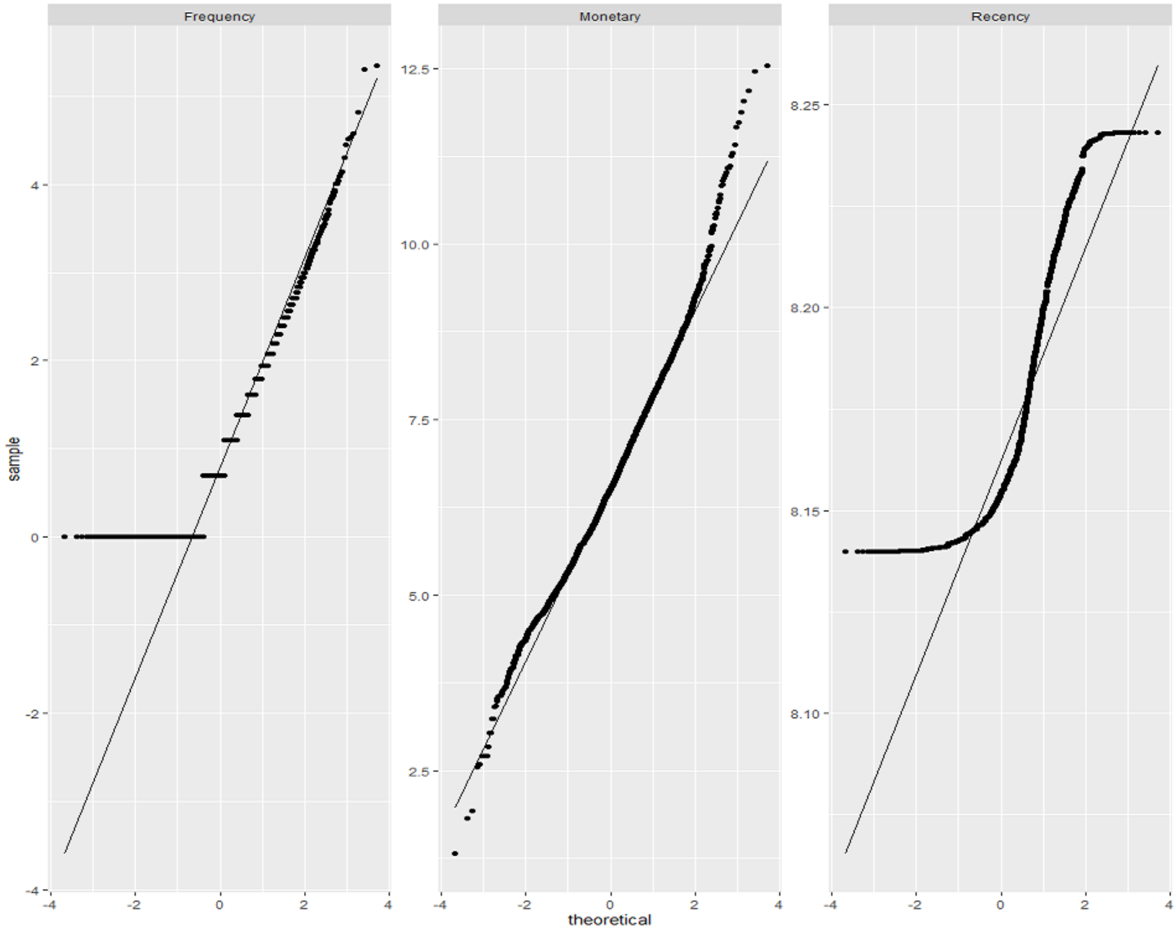
再來，從boxplot看得出來，outlier的狀況較小，同時也看到在Frequency和Recency有左偏的情況。



從pairplot，可以看到原本集中的資料都散開了，並且還可以看到Monetary和Frequency有類似線性的情形。因此接下來我們用correlation plot來做確認。



Monetary和Frequency確實有部分的正向線性狀況，自然的解釋是:越常來的客人，總共花的錢自然越多。除此之外，我們還看到Frequency和Recency有些微的負向線性關係，但並不影響分群結果。



最後，由於後面章節會用到GMM來做分群，因此想要看看QQplot的表現。Monetary除了尾端資料以外，其餘部分都十分貼近normal分配；Recency則是明顯的light-tailed的分布；Frequency除了0點以外，其他分布也都很接近normal，而0點則是看起來像被中斷的normal集中在0點這邊，但由於幾乎只有符合一半的常態，因此不能下定論是否與normal很接近。

Simulation

由於在上個章節，我利用log讓資料做轉換，所以在這個章節，我想要嘗試利用log multivariate normal mix model去呈現出資料前後轉換對於K-means和GMM分組的能力。因為我的資料是三個維度的，所以我需要產生 log-trivariate normal的資料。

Gibbs Sampling

Gibbs Sampling是利用貝式的方式，迭代出我們所需的資料。假設我們想要生成資料為 $X = (X_1, \ldots, X_n) \sim \pi(X)$，其中$\pi(X)$是指穩定分布。我們定義 $X_{-k} = (X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n)$，也就是說$X_{-k}$就是所有資料$X$但不包含$X_k$。對於$X_k$，我們定義Full Conditional Distribution是$\pi(X_k|X_{-k})$，而Full Conditional Distribution和Joint Distribution的關係可以由以下式子說明:

$$\pi(X_k|X_{-k}) = \frac{\pi(X_k, X_{-k})}{\int \pi(X_k, X_{-k}) dX_k} \propto \pi(X_k, X_{-k}) = \pi(X)$$

Gibbs Sampling把原本高維度的分布降到1維度來處理，因此在處理高緯度資料中，Gibbs Sampling常常是一個有用的工具。在這篇報告中，我將利用Gibbs Sampling 來生成 log-trivariate normal distribtuion。假設我想要生成資料為:

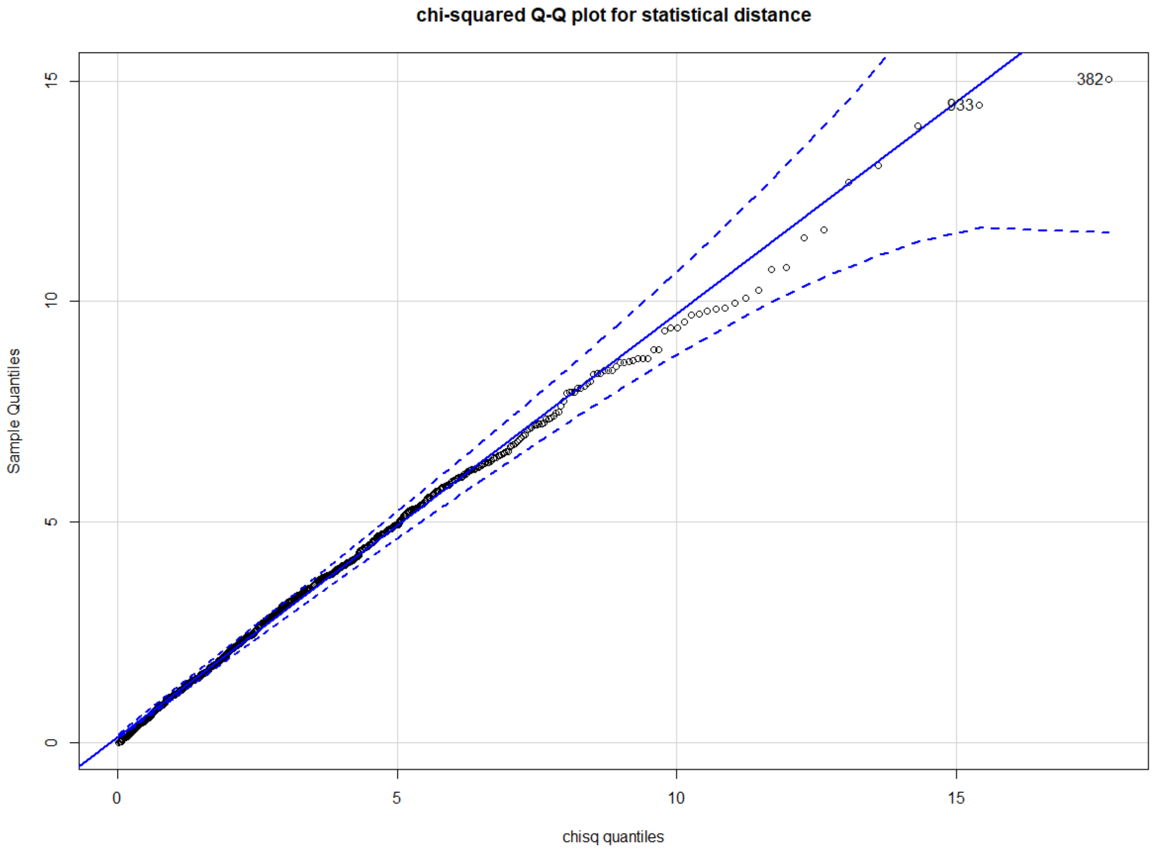
$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_{-1} \end{bmatrix} \sim N(\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_{-1} \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \Sigma_{13} \\ \Sigma_{21} & \Sigma_{22} & \Sigma_{23} \\ \Sigma_{31} & \Sigma_{32} & \Sigma_{33} \end{bmatrix} = \begin{bmatrix} \Sigma_{11} & \Sigma_{1(-1)} \\ \Sigma_{(-1)1} & \Sigma_{(-1)(-1)} \end{bmatrix})$$

那麼，log-trivariate normal distribtuion的生成方式是

Algorithm:

1. 放入初始值 $X^{(0)} = \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{bmatrix} = \mu$
2. 生成第(k+1)組數據
   $x_1^{(k+1)}|x_{(-1)}^{k} \sim N(\mu_1 + \Sigma_{1(-1)}\Sigma_{(-1)(-1)}^{-1}(x_{-1}^{(k)} - \mu_{-1}), \Sigma_{11} - \Sigma_{1(-1)}\Sigma_{(-1)(-1)}^{-1}\Sigma_{(-1)1})$, $x_2^{(k+1)}$和$x_3^{(k+1)}$也是相似作法。
3. 重複迭代第2個步驟
4. 回傳$Y = exp(X)$

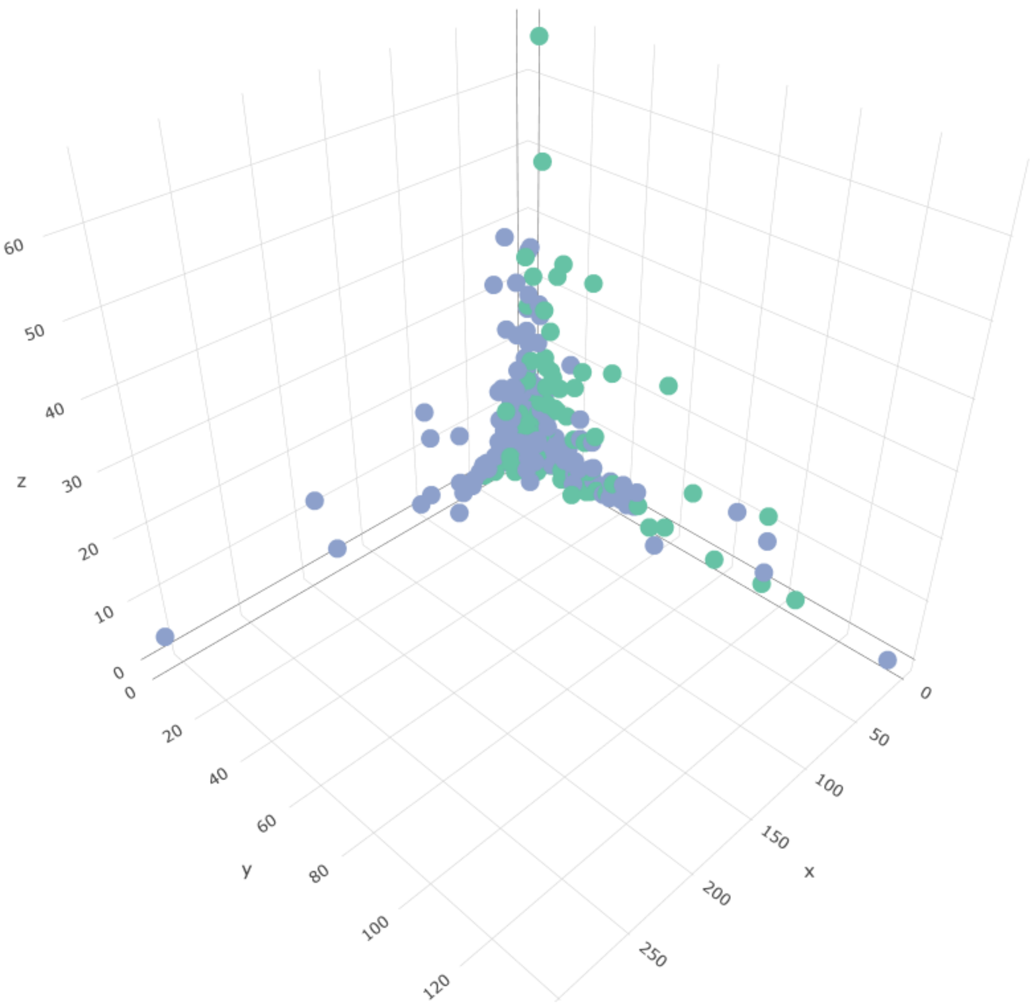
由於在生成的過程中，我們也會生成trivariate normal，因此我利用其數值去檢驗我所生成的資料是否為我所想要的，我利用χ^2的QQplot檢驗:



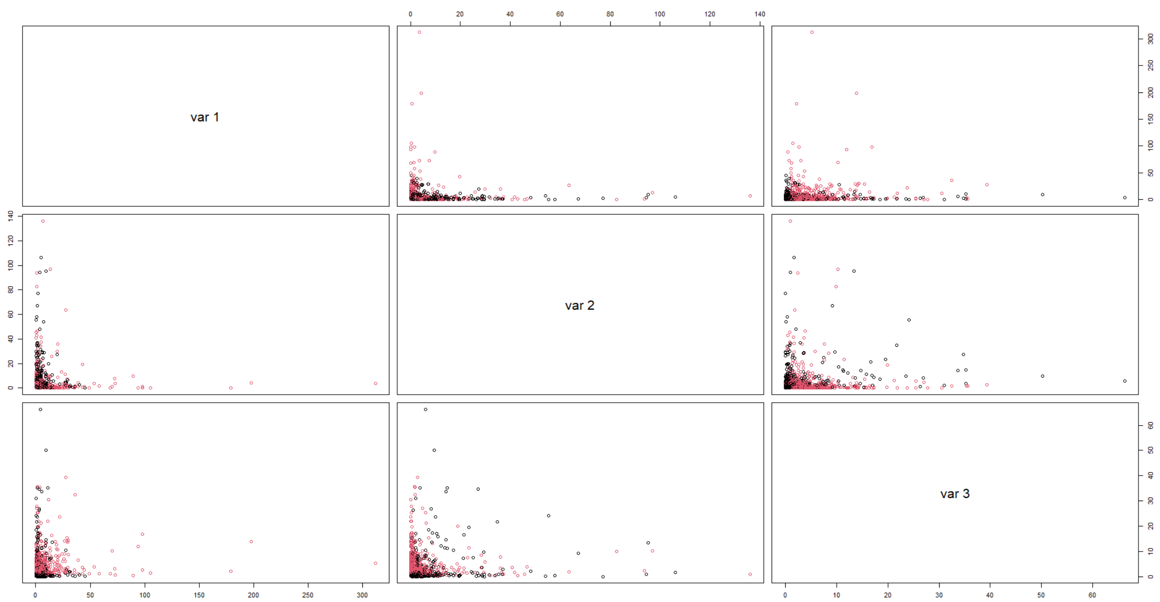
可以看到，所有數值都在顯得附近，幾乎沒有outlier，所以可以確定我們所生成的是trivariate normal distribution。

Simultation Results

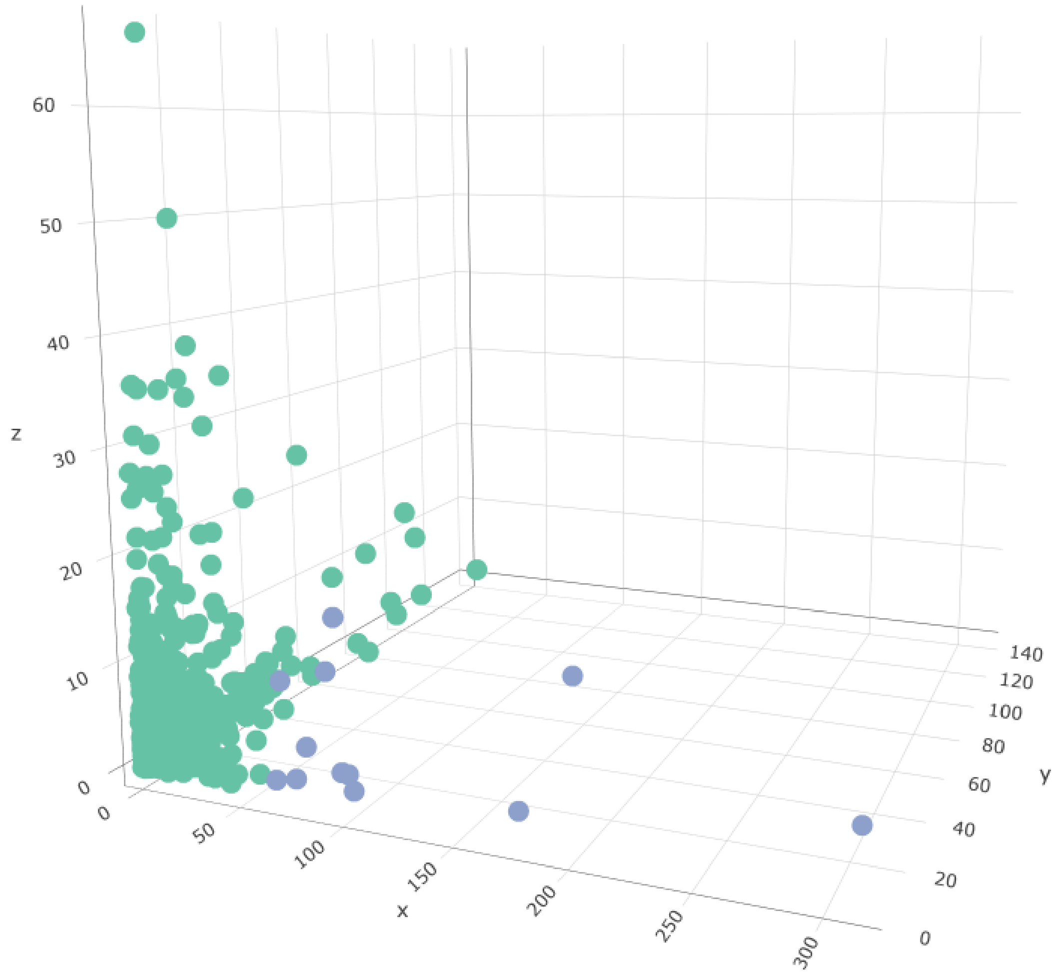
我利用Gibbs Sampling生成兩組1000筆的log-trivariate normal distribution，然後依照設定的機率從兩組抽出1000筆資料。其中一組的分布為$logN(\begin{bmatrix}1\\1\\0\end{bmatrix},\begin{bmatrix}1 & 0.3 & 0.4\\0.3 & 2 & 0.1\\0.4 & 0.1 & 3\end{bmatrix})$，抽取機率為0.6；另一組的分布為$logN(\begin{bmatrix}1\\0\\1\end{bmatrix},\begin{bmatrix}2 & 0.6 & 0.4\\0.6 & 3 & 0.6\\0.4 & 0.6 & 1\end{bmatrix})$，抽取機率為0.4。以下是生成後的3D 散佈圖:



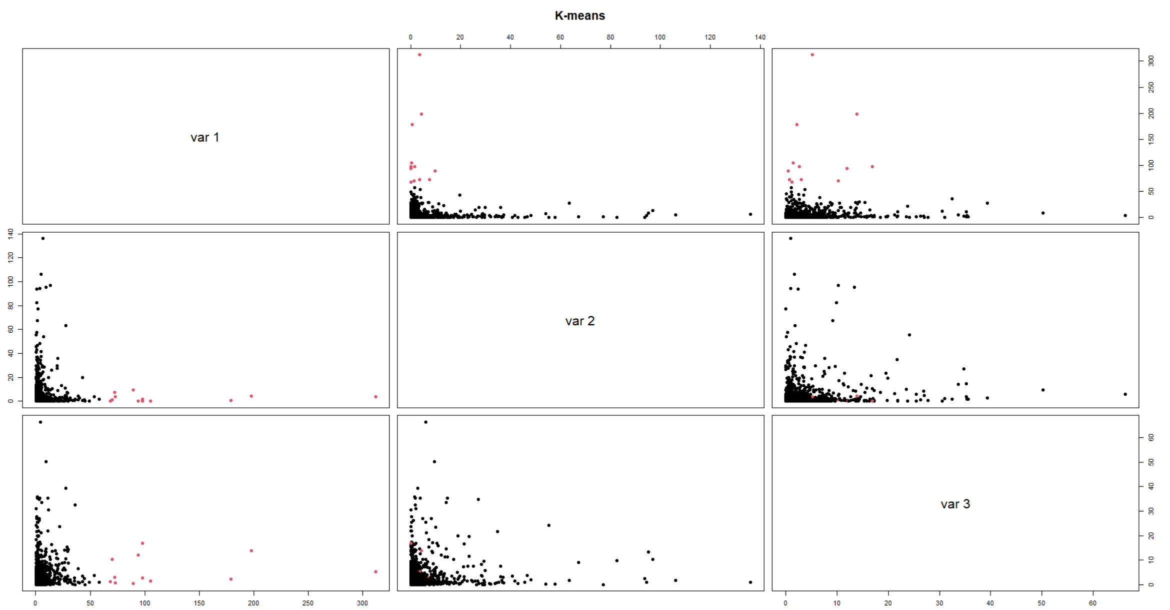
可以看到資料大多都集中在原點附近，並且兩群的分布極為交錯。我們可以用pair plot再看更加仔細



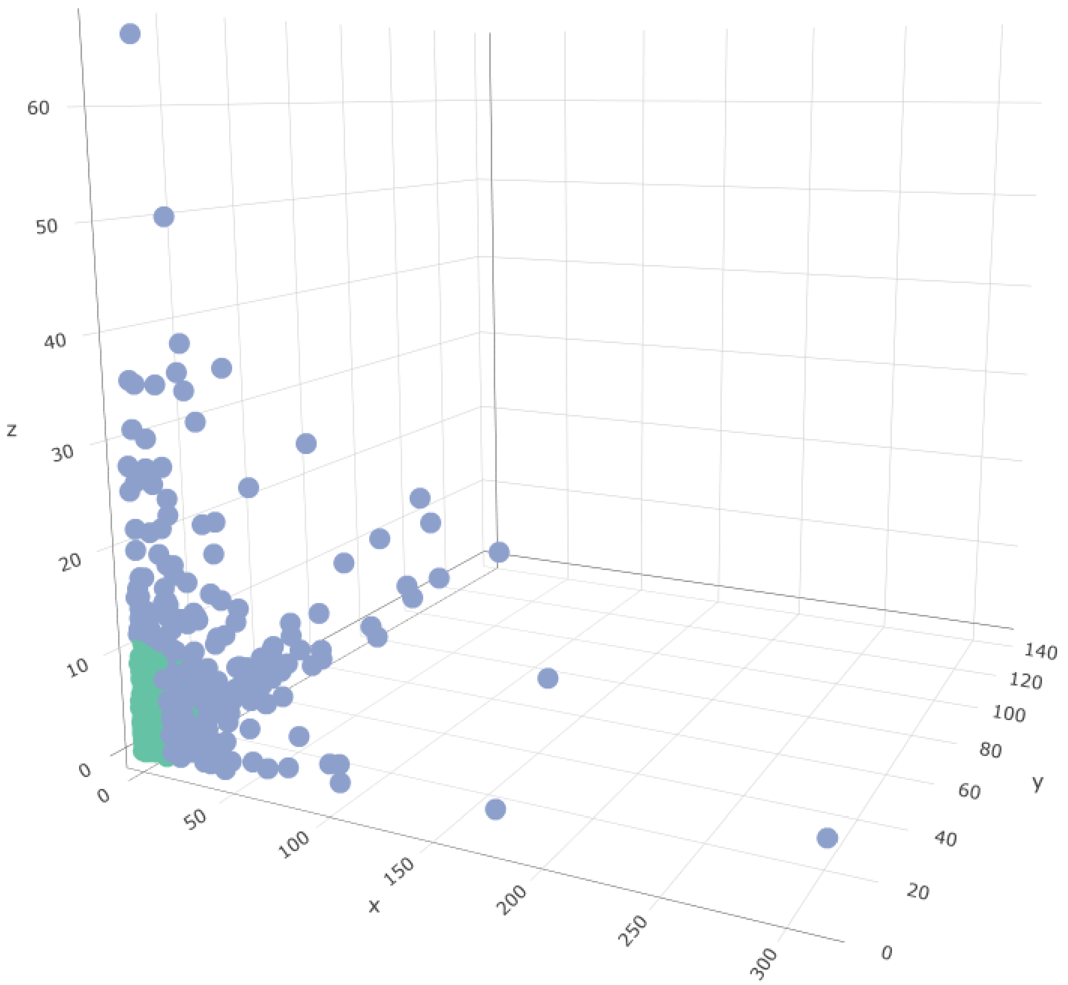
可以看得出來，就算投影在2維度，也是資料互相交錯的狀況。接下來，我將要利用K-means來分群:



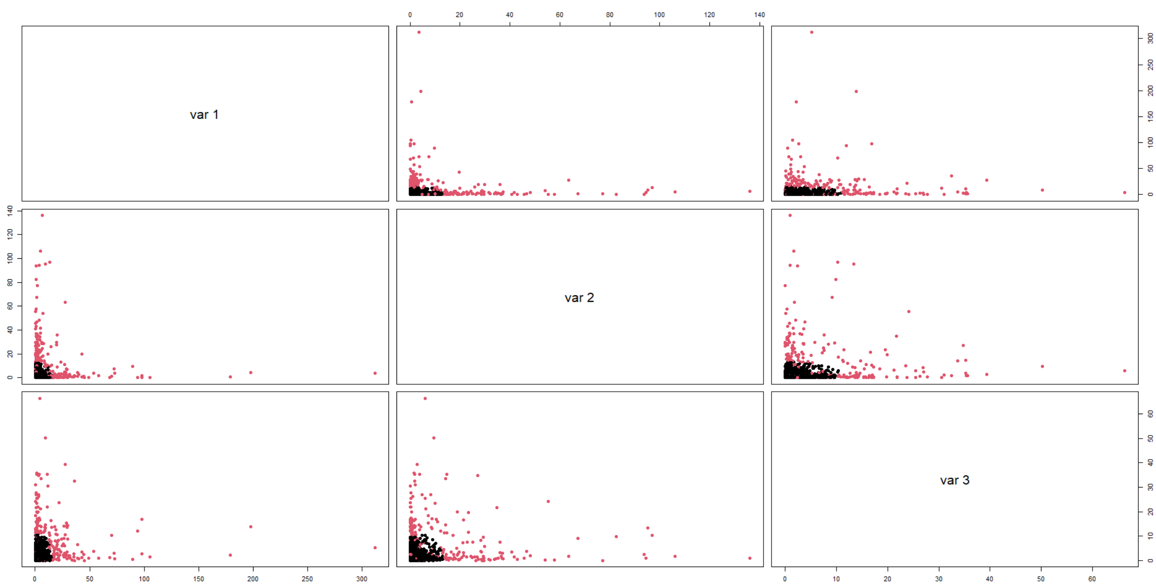
可以看到，K-means的效果非常不好，他就只有把某一側的outlier分出來而已，接下來來看pair plot更加明顯:



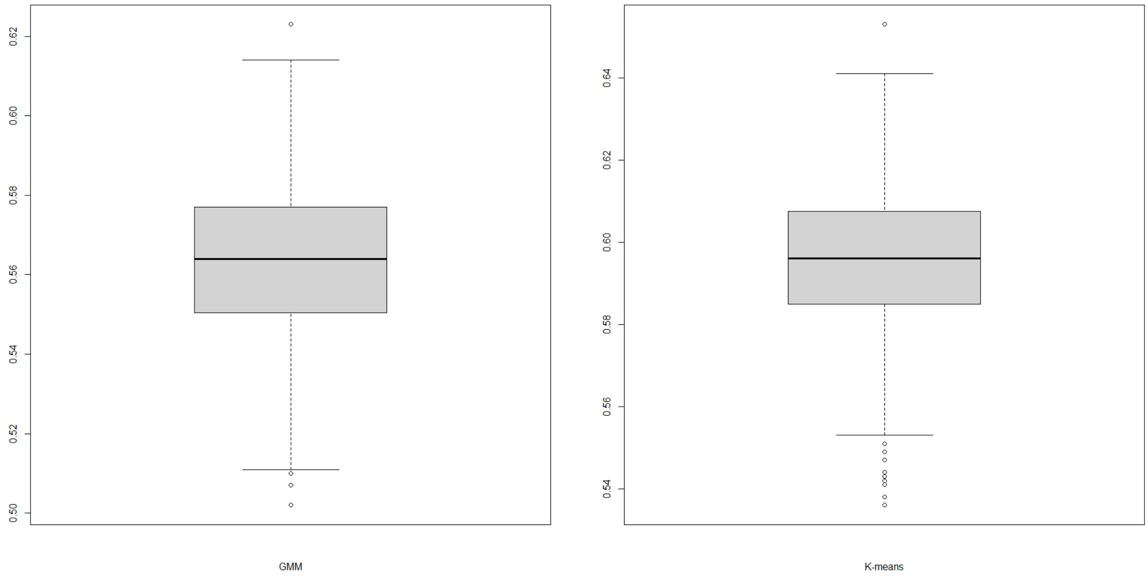
更加明顯的是，K-means只有把x軸(var1)的outlier抓出來而已。所以在log-normal的情況下，K-means無法分組的很好。再來我們看GMM的表現:



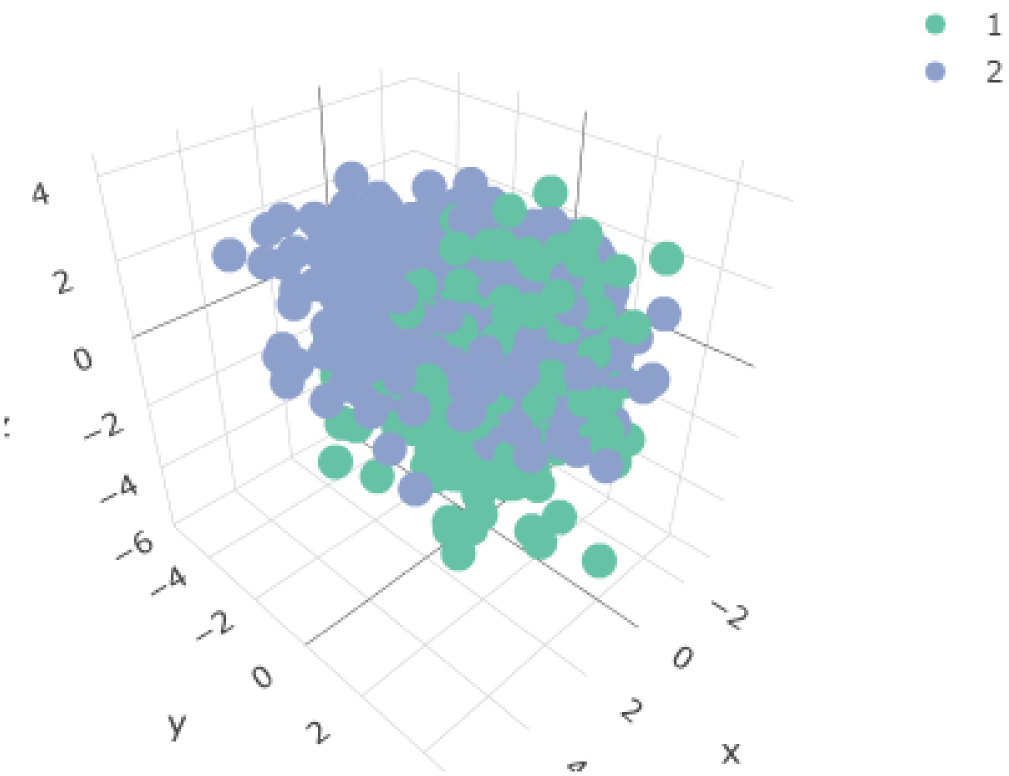
雖然GMM並沒有分得比較好，但是有趣的是，GMM分組分成集中在原點附近的為一群，其他的一群。我們再看pair plot:



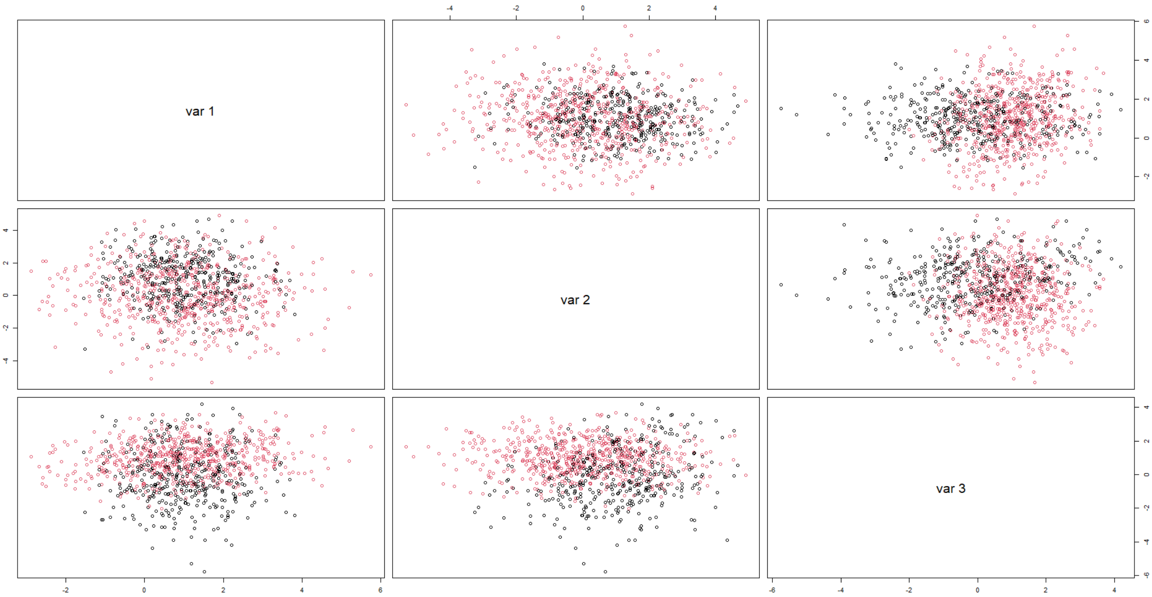
pair plot很明顯的顯示了GMM的分群狀況，主要是把離原點近的和遠的個別分成一群。但是不管是K-means或是GMM，在這裡的分群效果都很差，以下是模擬1000次的分群之後計算的accuracy rate:



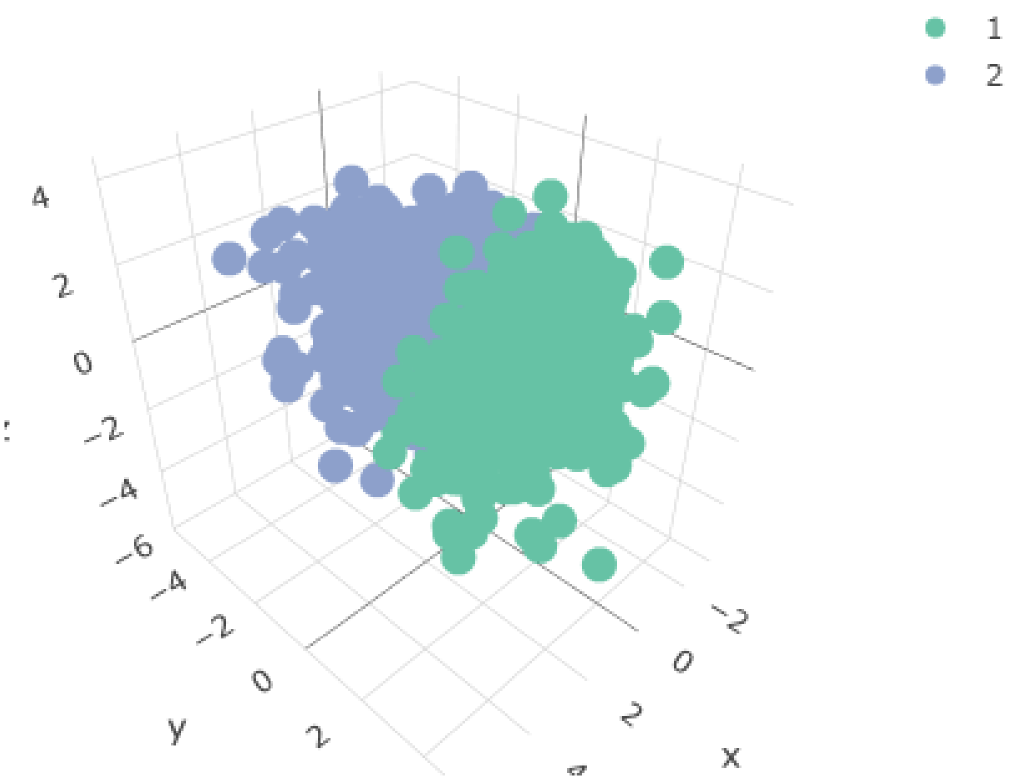
可以看到，GMM和K-means的準確率都在0.6附近，並沒有說特別的好。在log-normal的情況下，兩者的表現都很不好，如果做資料轉換，取log之後，是否會變得更好?接下來看到資料轉換後的模擬資料:



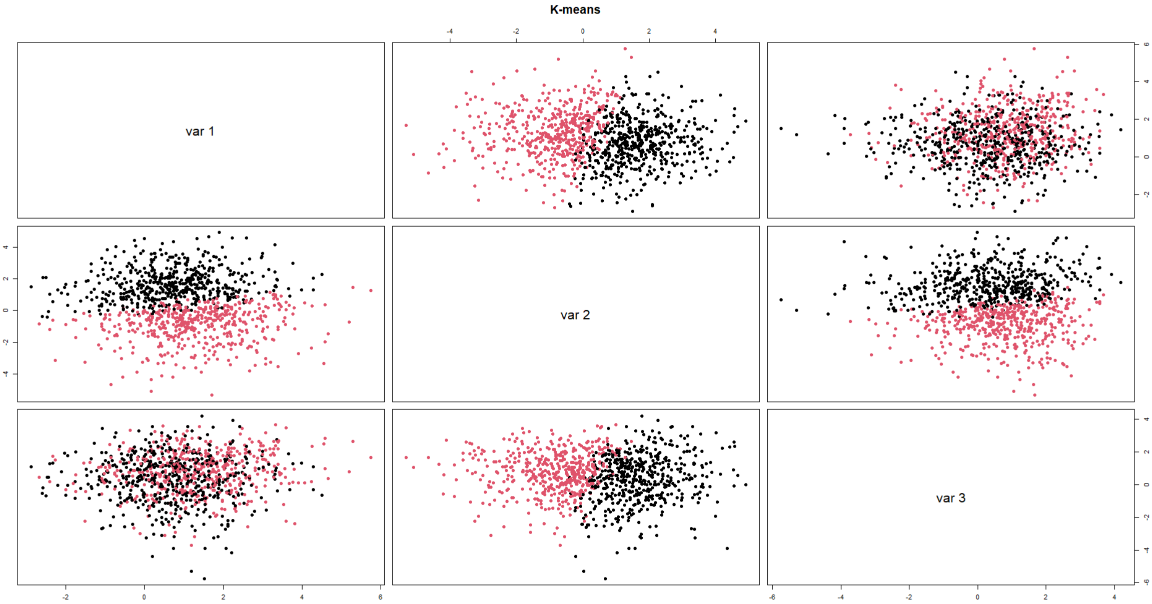
我們能看到，模擬資料在取log之後，就是單純的trivariate normal distribution。從這張圖，就看得出來很像2坨雲朵交織在一起的感覺，再來看到pair plot:



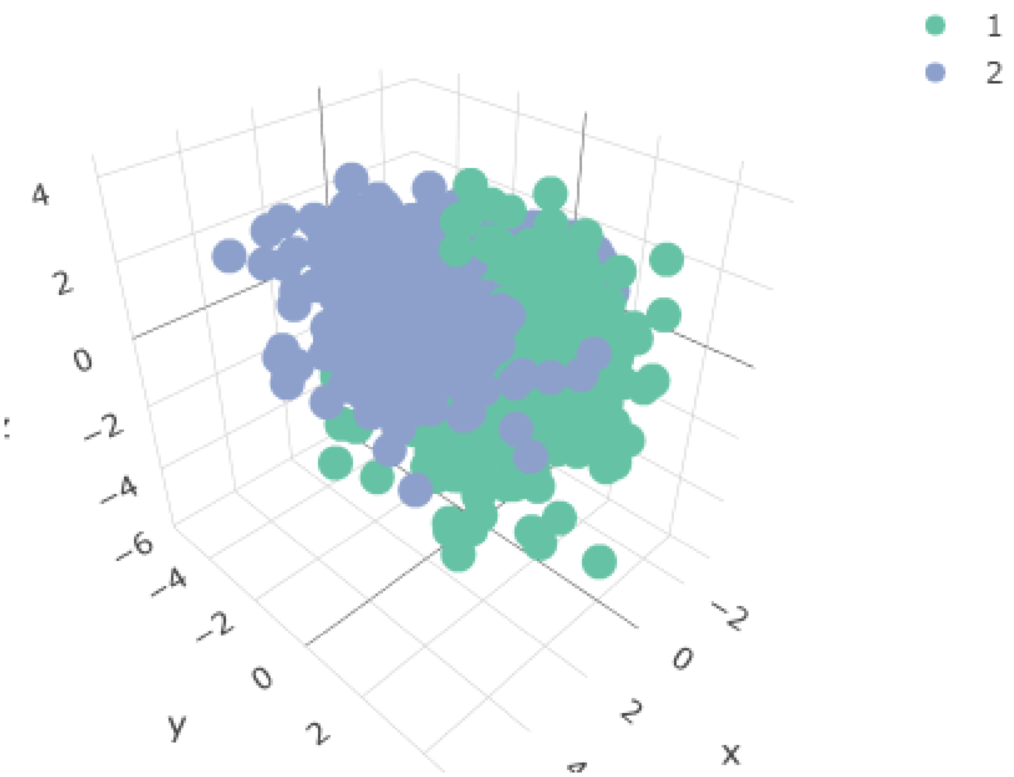
基本上就是兩個分佈交錯在一起的感覺。接下來，我們來看K-means在這情況下的分群表現:



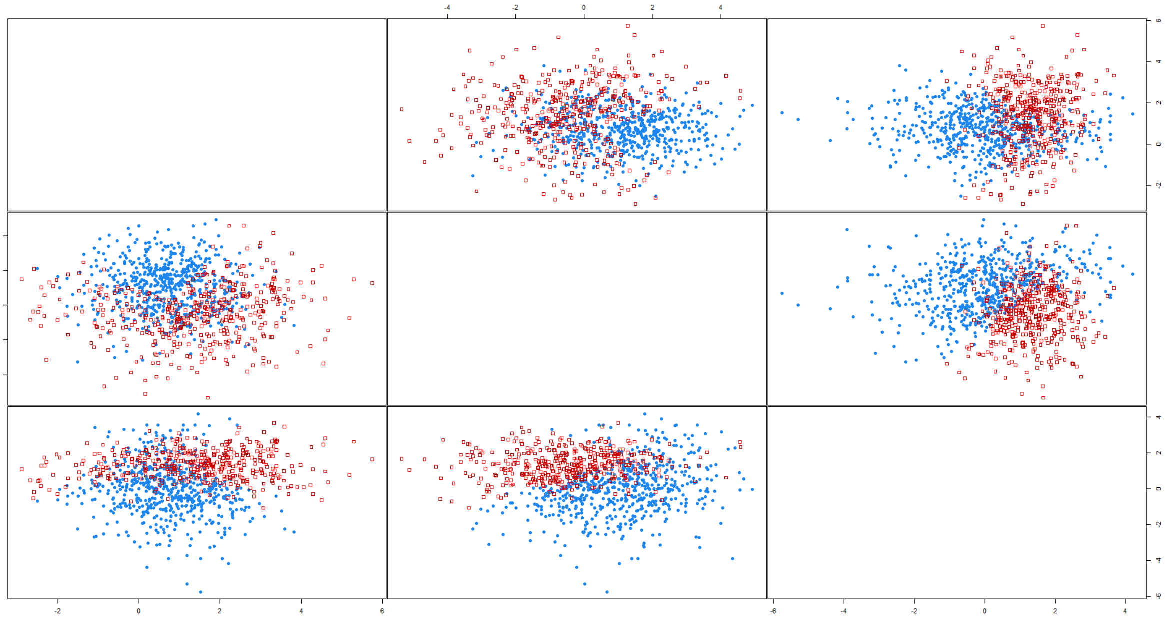
從圖中，我們能夠看出K-means的分群一定會有明確的界線，並不能分出交錯的資料。接下來看pair plot:



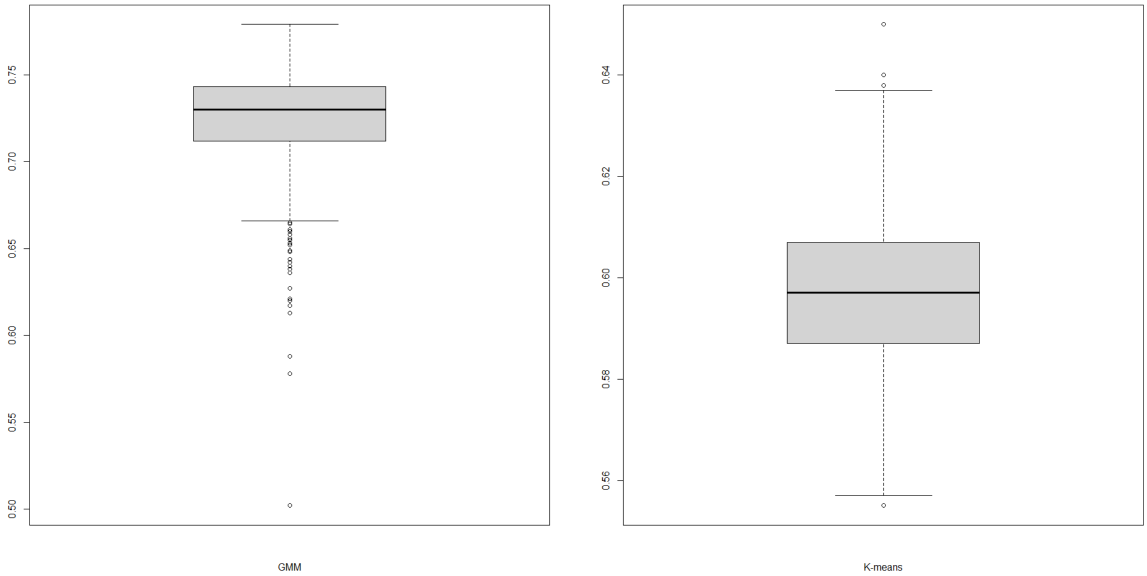
從pair plot，我們可以看到K-means主要是以y軸(var2)為主要分界的參考。雖然已經做完資料轉換，並且讓資料散開在各處了，但是K-means在處理資料會有交錯的情況下，並不能有很好的分群。再來我們來看GMM的表現:



我們可以看到，GMM在這種狀況下，可以分出交錯的組別，並不會像K-means一樣會有極為明顯的分界線。我們繼續來看pair plot的狀況:



可以看得出來，在資料交錯分布的情況下，GMM分組會比K-means好很多，尤其可以看到右上角的分群狀況，是分出兩組資料互相交錯。因此可以發現，在做完資料轉換之後，資料仍然會互相交錯，此時K-means的分群能力較GMM來說比較不好。以下是模擬1000次的分群之後計算的accuracy rate



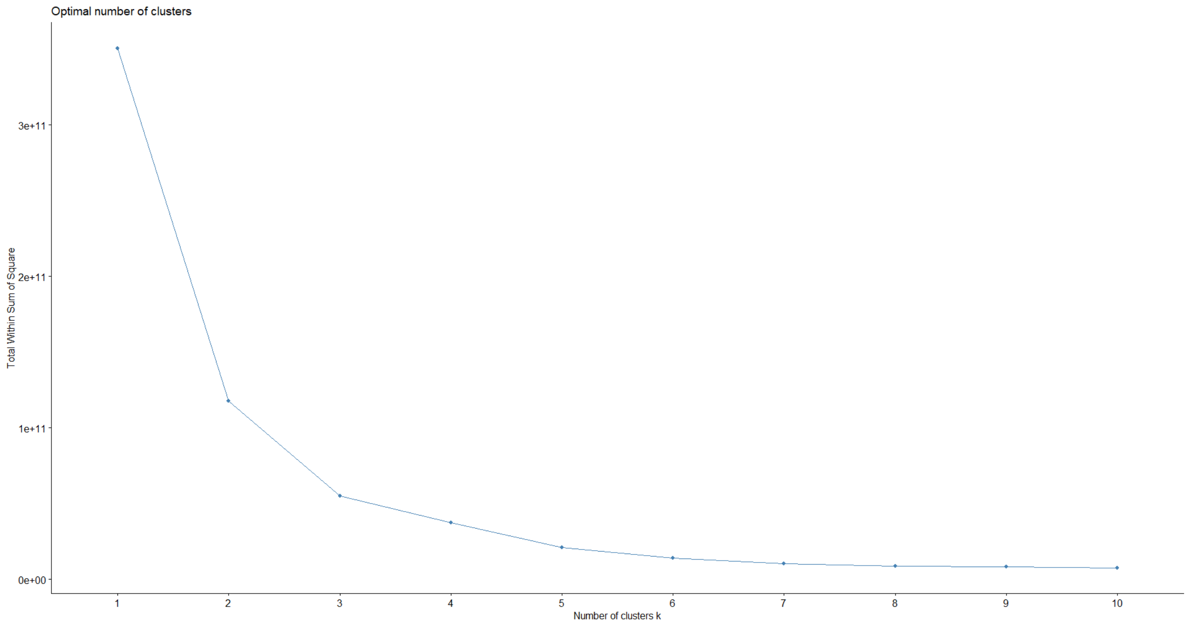
K-means的accuracy rate仍然在0.6附近，但是GMM則提升到了0.75左右。也就可以說明，資料轉換後，資料若本身還有交疊的部分，K-means就無法做好分類，但是GMM在資料轉換後，分組能力就很好。但是不能忽略的是，這份的模擬資料是有模型所支撐的，也有可能是讓GMM的accuracy rate提高的原因。

Case Study

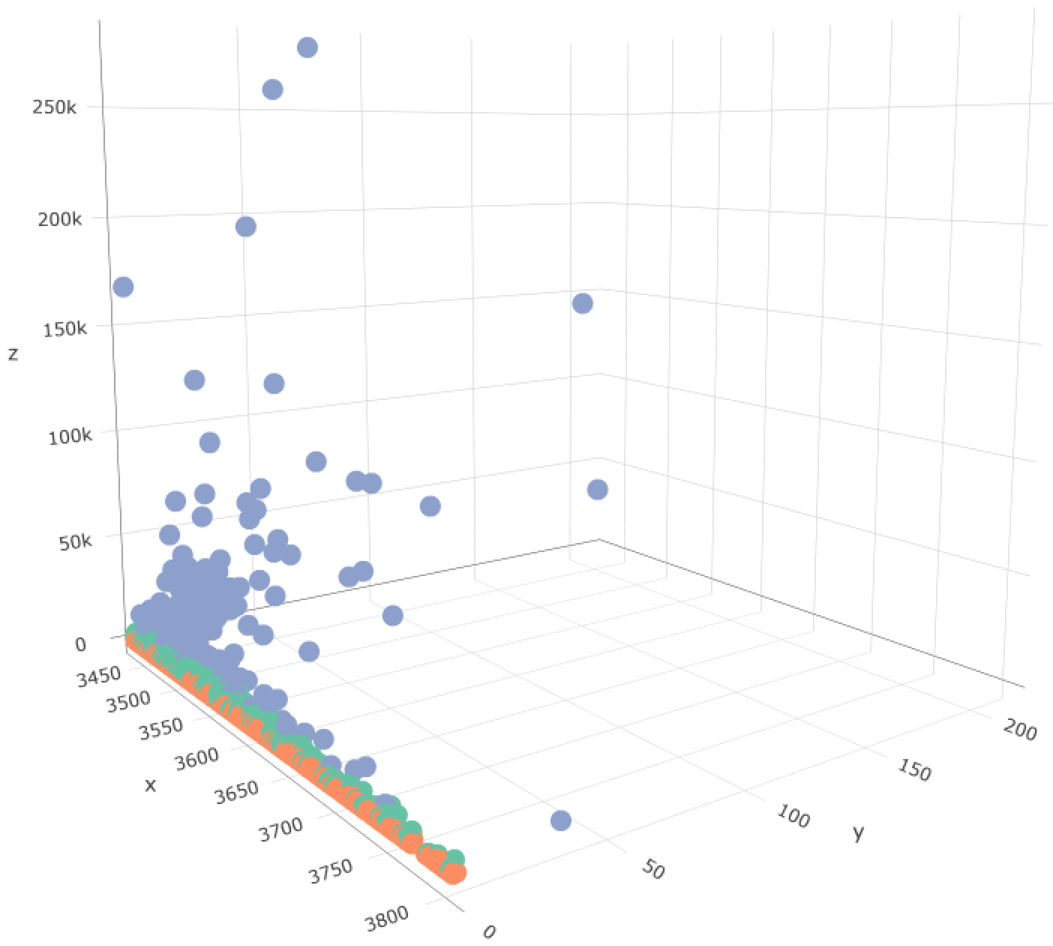
這邊開始來處理所拿到的資料。在Introduction中，我們提到要利用行銷學中的RFM模型，但是實際上，當使用RFM模型時，如何判斷該客人是否為目標客群，都是由使用者自己的經驗所判斷的，並沒有特定的數值、界線、甚至是分群群數。對於我所用的方法:GMM和K-means，恰好都是必須要提前輸入群數，所以必須要利用別的方式判斷分群的狀況。首先，我們來看K-means的狀況。

K-means

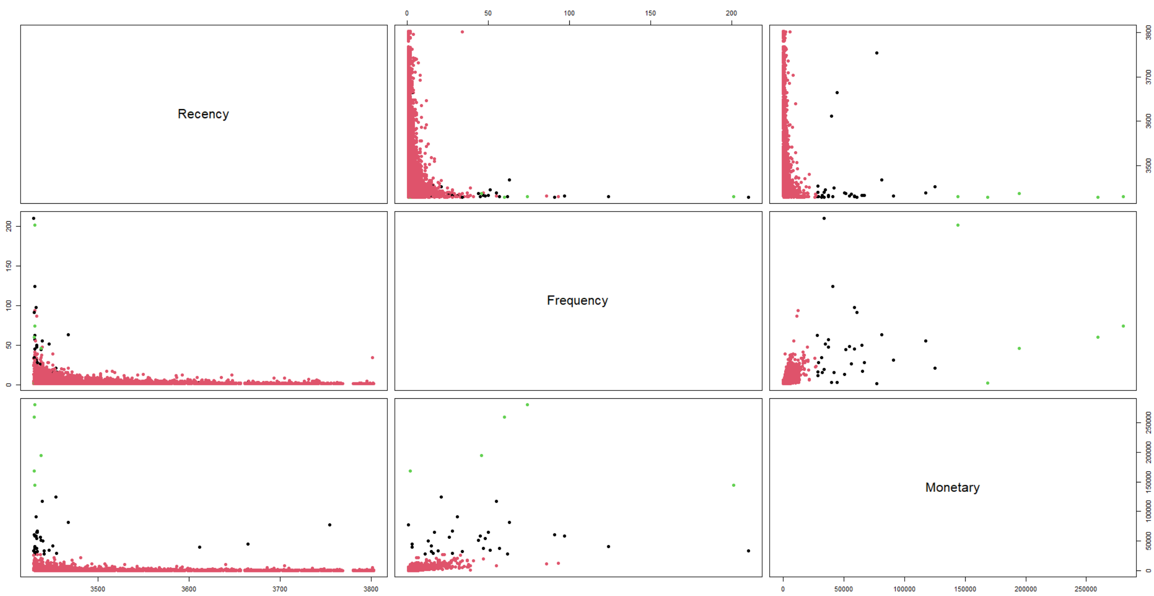
在K-means，我們通常會用Elbom method，來看如何選群數(K)。接下來我先使用原始資料來做分群，以下就是這筆資料的Scree plot，其中x軸為群數K，y軸為群內變異的加總(Total Within Sum of Square)



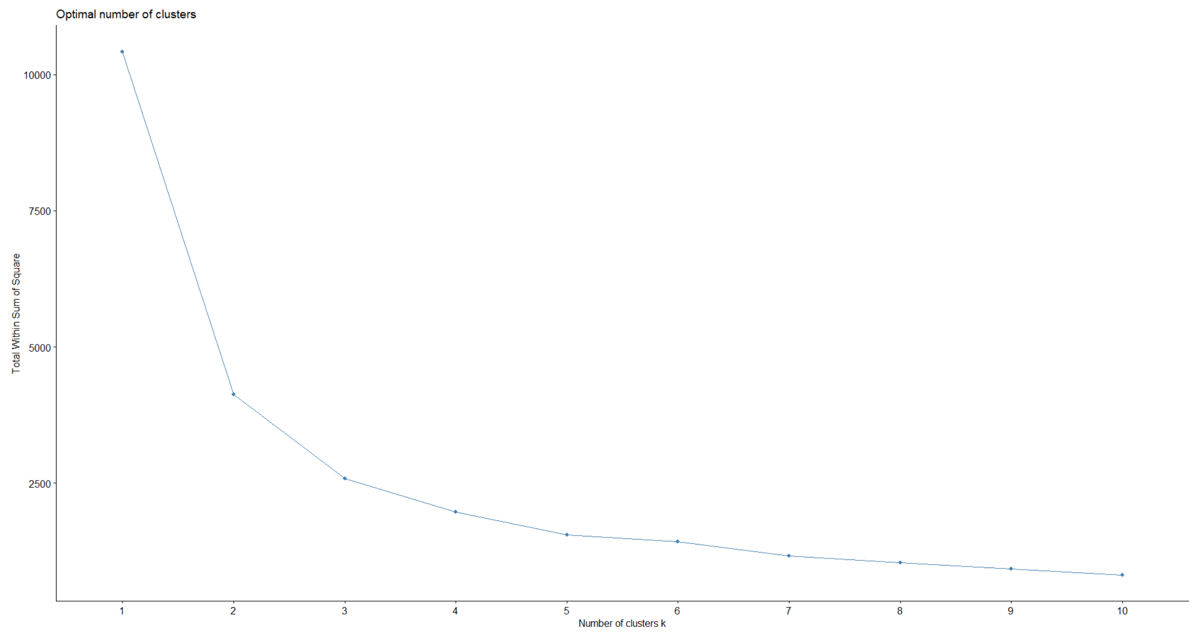
這張圖中，我們要找尋讓線的斜率變化最大的點。換句話說，要找一個點使得點右邊的線趨於垂直，且點左邊的線趨於水平。在這張圖，可以看得出來在$K=3$的點是最符合這些條件的。接著，把$K=3$帶入K-means模型，我們可以的以下結果:



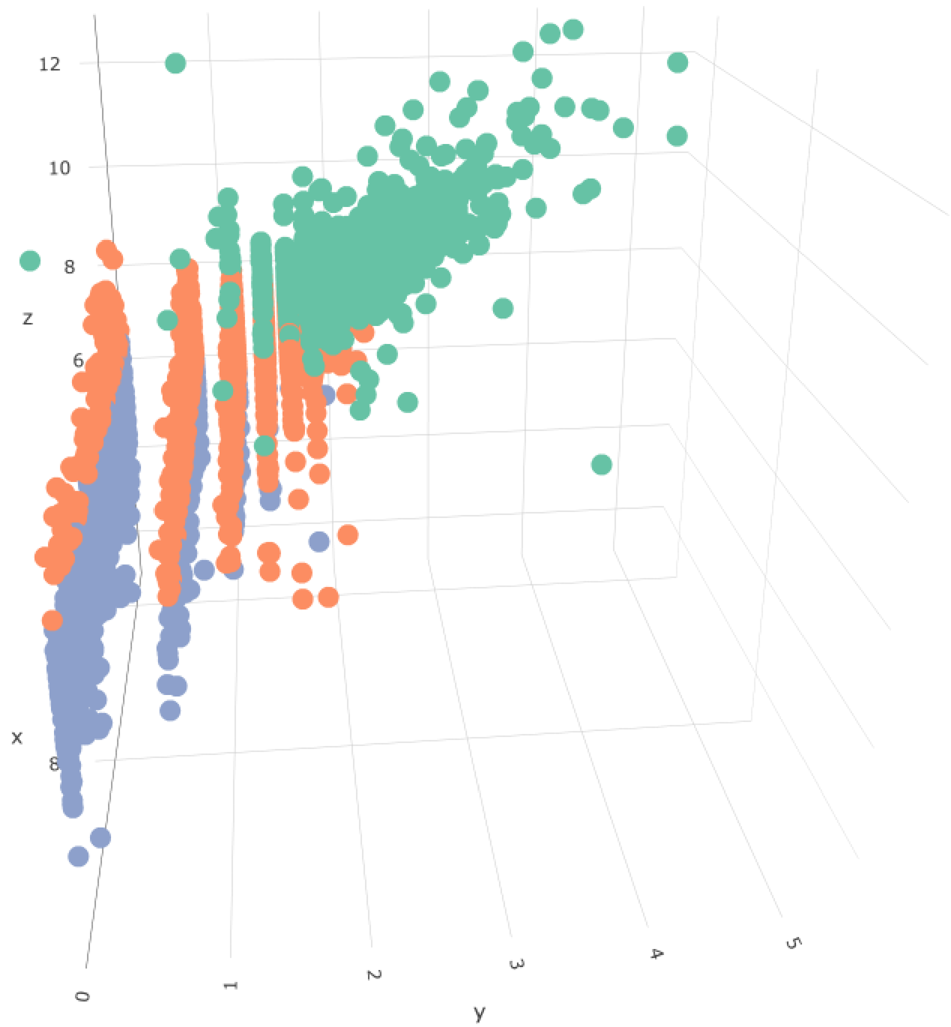
明顯看到，kmeans沿著Frequency(y軸)做分界，分出了outlier以及其他兩群。我們可以看pair plot更加明顯:



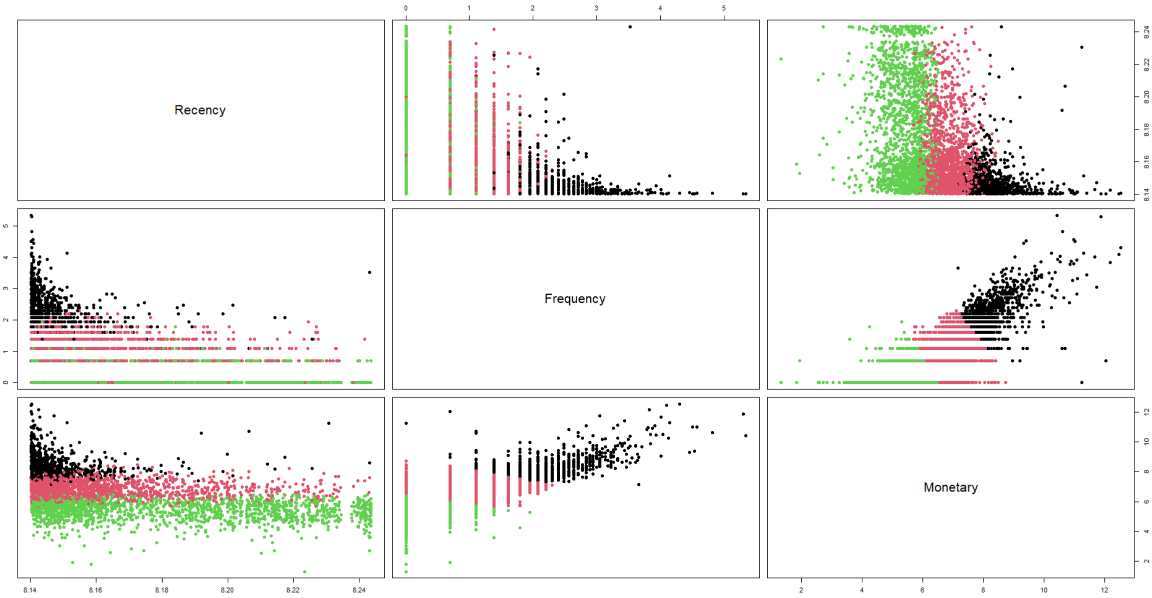
可以看得出來K-means基本上只是把outliers分出來而已，沒有更深入的客群的分析，對於分析並沒有幫助，因此我不採用此方法。再來，我要使用取log後的資料做分群:



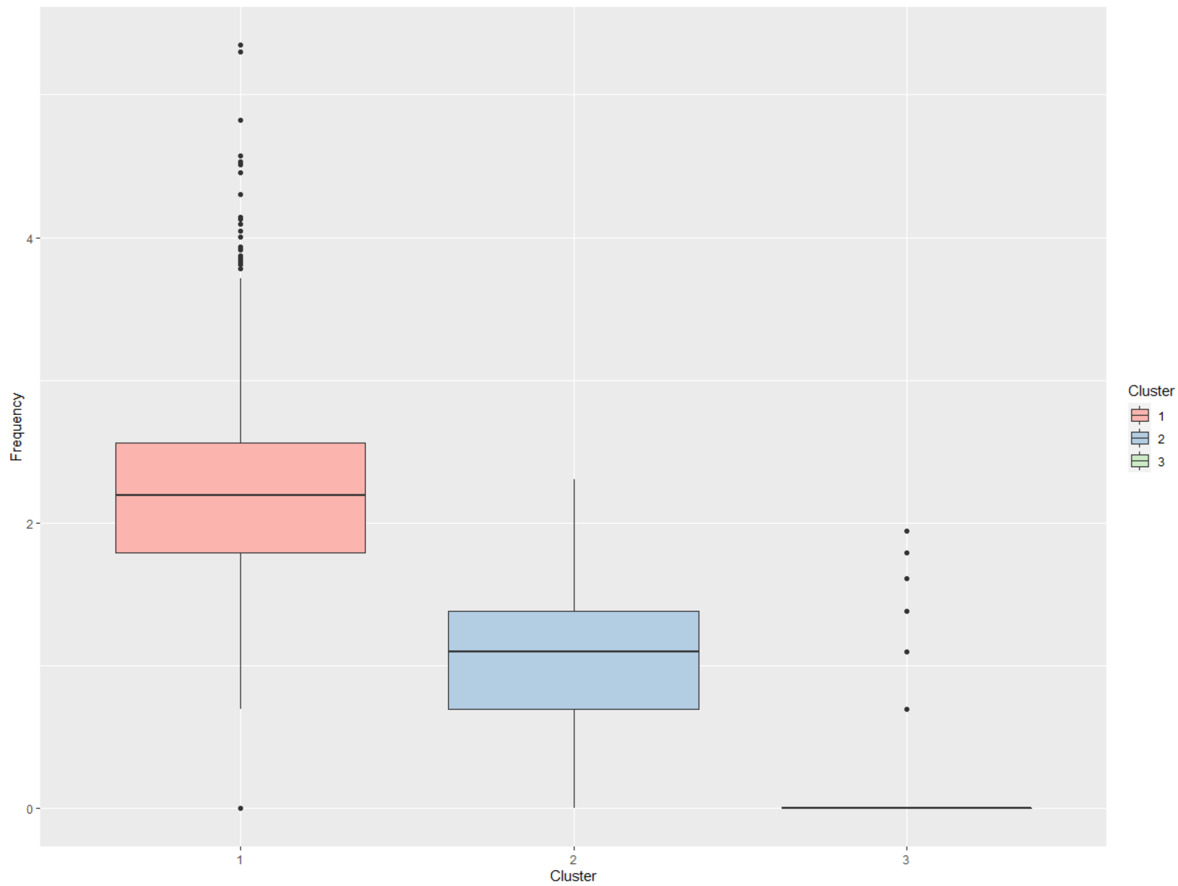
同樣的，我們也看得到在$K=3$的點是我們所要找的點，因此帶入分群:



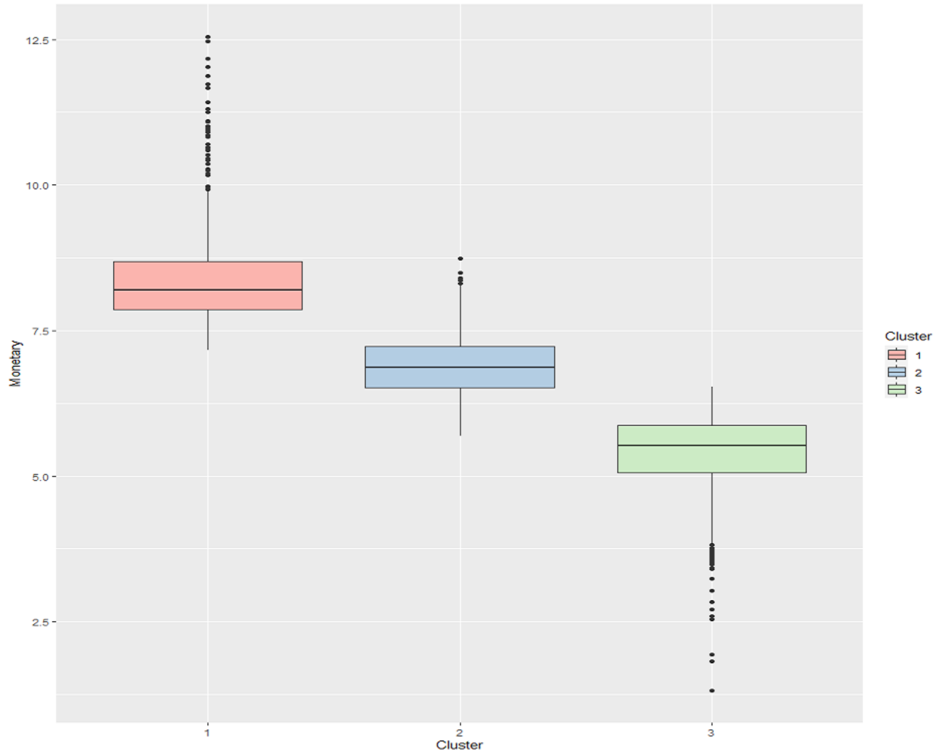
我們能夠看得出來，K-means主要是沿著Monetary(z軸)方向做分界。並且著三群有很明確的客群。詳細可以看pair plot:



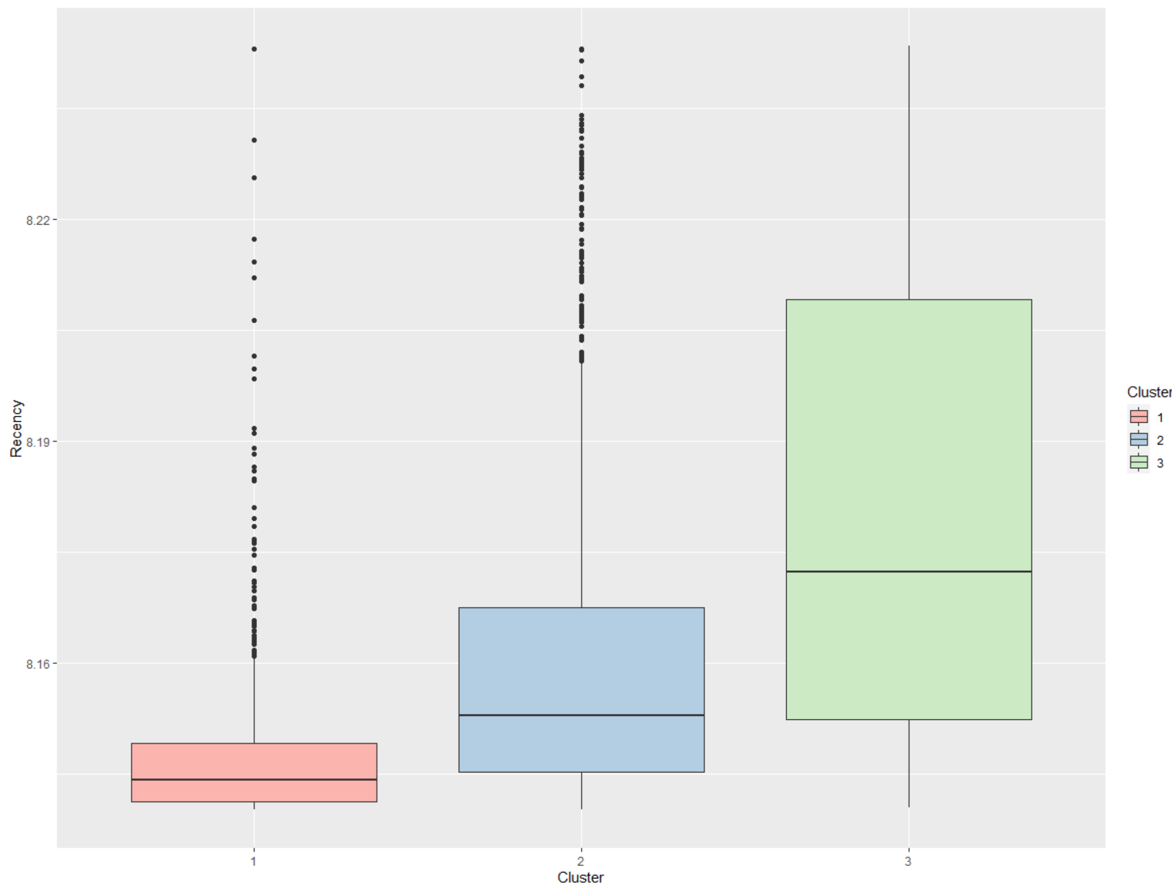
從Monetary v.s. Recency的圖，可以約略看得出來這3群的邊界是垂直於Monetary的方向，也就是以Monetary切開。從Monetary v.s. Frequency的圖，可以看到分群邊界是兩條水平的斜線。但從Recency v.s. Frquency就看不出來明顯的邊界。再來我們來看個群的特色



首先先看到Fequency，很明顯地看到第1群是數值最高的，也就是常客群，而第3群則是數值最低的，也就是購物的次數很低，其可能性是流失的客人或是新客人，因此還要繼續分析Monetary:



再來看到Monetary，這張圖有趣的是，存在的outlier也跟著分群被分了出來。所以看到第一群是貢獻最大的一群，而第三群是貢獻最低的一群。也同時驗證了pairplot中，對於Monetary界線明確的看法。最後看到Recency:



對於Recentcy這變數，K-means並沒有分得很好，第1.2群的outlier特別多，並且第三群大部分的客群佔了全部的範圍，因此無法區分3者差異。總結來說，K-means幫助我們找到三組客人，這三組客人的描述如下。

分群編號	Frequency	Monetary	Recency	客群推測
1	最高	最高	最低	VIP級的死忠客人
2	中間	中間	中間	普通客人
3	最低	最低	最高	流失的客人或是新客人

GMM

在GMM中，我們除了要選群組數量以外，還要選擇模型形狀，而這兩者的選擇都是用BIC來決定。首先，GMM因為是模型做分類的，因此對於模型的假設必須要確定，在我所使用的套件Mclust中，它分成了以下的模型:

Model	$\boldsymbol{\Sigma}_k$	Distribution	Volume	Shape	Orientation
EII	$\lambda \boldsymbol{I}$	Spherical	Equal	Equal	—
VII	$\lambda_k \boldsymbol{I}$	Spherical	Variable	Equal	—
EEI	$\lambda \boldsymbol{A}$	Diagonal	Equal	Equal	Coordinate axes
VEI	$\lambda_k \boldsymbol{A}$	Diagonal	Variable	Equal	Coordinate axes
EVI	$\lambda \boldsymbol{A}_k$	Diagonal	Equal	Variable	Coordinate axes
VVI	$\lambda_k \boldsymbol{A}_k$	Diagonal	Variable	Variable	Coordinate axes
EEE	$\lambda \boldsymbol{D}\boldsymbol{A}\boldsymbol{D}^\top$	Ellipsoidal	Equal	Equal	Equal
EVE	$\lambda \boldsymbol{D}\boldsymbol{A}_k\boldsymbol{D}^\top$	Ellipsoidal	Equal	Variable	Equal
VEE	$\lambda_k \boldsymbol{D}\boldsymbol{A}\boldsymbol{D}^\top$	Ellipsoidal	Variable	Equal	Equal
VVE	$\lambda_k \boldsymbol{D}\boldsymbol{A}_k\boldsymbol{D}^\top$	Ellipsoidal	Variable	Variable	Equal
EEV	$\lambda \boldsymbol{D}_k\boldsymbol{A}\boldsymbol{D}_k^\top$	Ellipsoidal	Equal	Equal	Variable
VEV	$\lambda_k \boldsymbol{D}_k\boldsymbol{A}\boldsymbol{D}_k^\top$	Ellipsoidal	Variable	Equal	Variable
EVV	$\lambda \boldsymbol{D}_k\boldsymbol{A}_k\boldsymbol{D}_k^\top$	Ellipsoidal	Equal	Variable	Variable
VVV	$\lambda_k \boldsymbol{D}_k\boldsymbol{A}_k\boldsymbol{D}_k^\top$	Ellipsoidal	Variable	Variable	Variable

Table 3: Parameterisations of the within-group covariance matrix $\boldsymbol{\Sigma}_k$ for multidimensional data available in the **mclust** package, and the corresponding geometric characteristics.

從上面的表，我們看出來每一個模型的假設，並且對於組內的變異數矩陣Σ_k做eigen-decomposition。舉例來說，如果我們是使用VVV模型，假設有G個模型，每個模型是$f_k(x;\theta_k) \sim N(\mu_k, \Sigma_k), k = 1, \dots, G$，則$\Sigma_k = \lambda_k D_k A_k D'_k$，其中$\lambda_k$是控制模型體積的常數、$A_k$則是決定模型機率密度的形狀的對角矩陣，並且$det(A_k) = 1$、$D_k$是決定橢球方向的正交矩陣。下圖可以看到GMM的2為模型形狀:

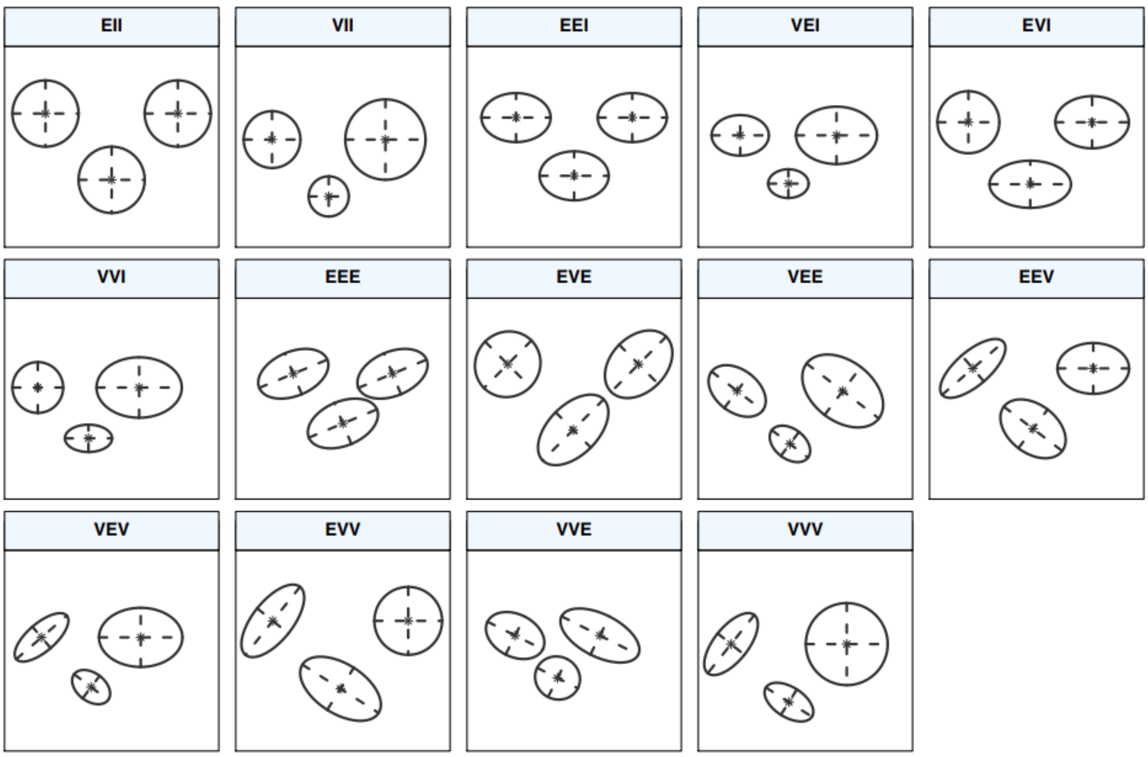
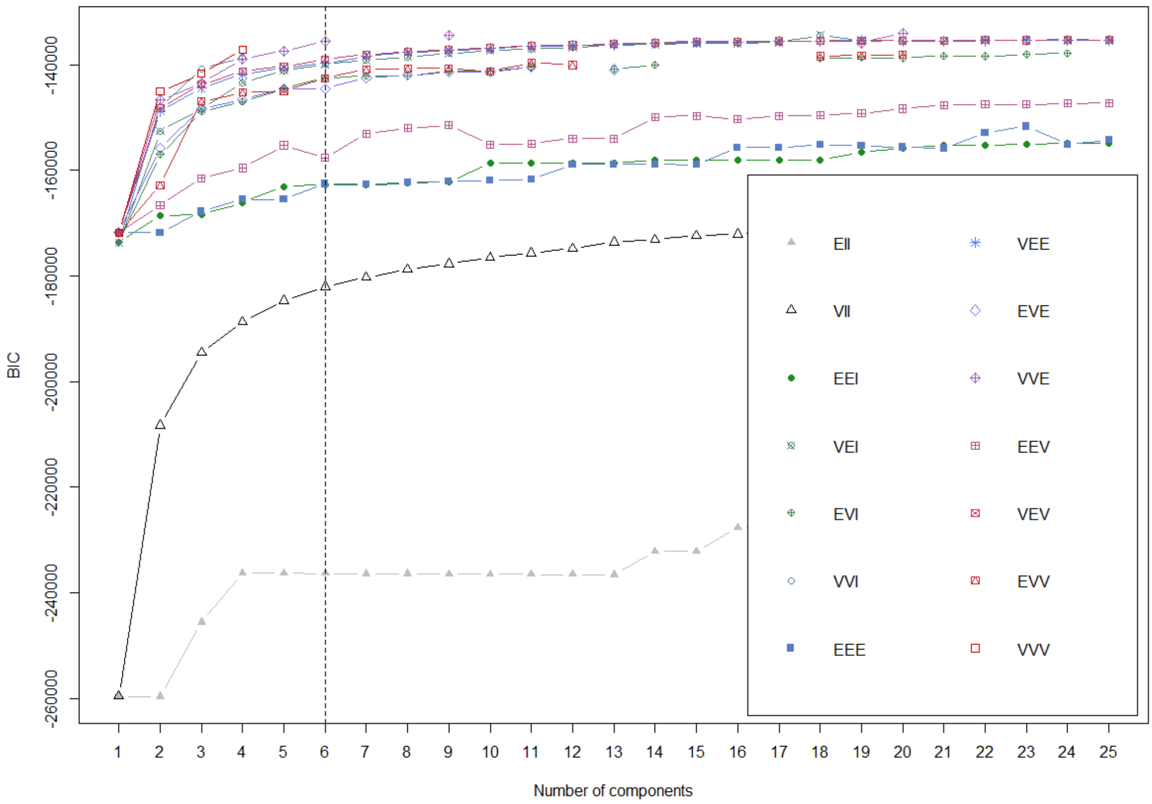


Figure 2: Ellipses of isodensity for each of the 14 Gaussian models obtained by eigen-decomposition in case of three groups in two dimensions.

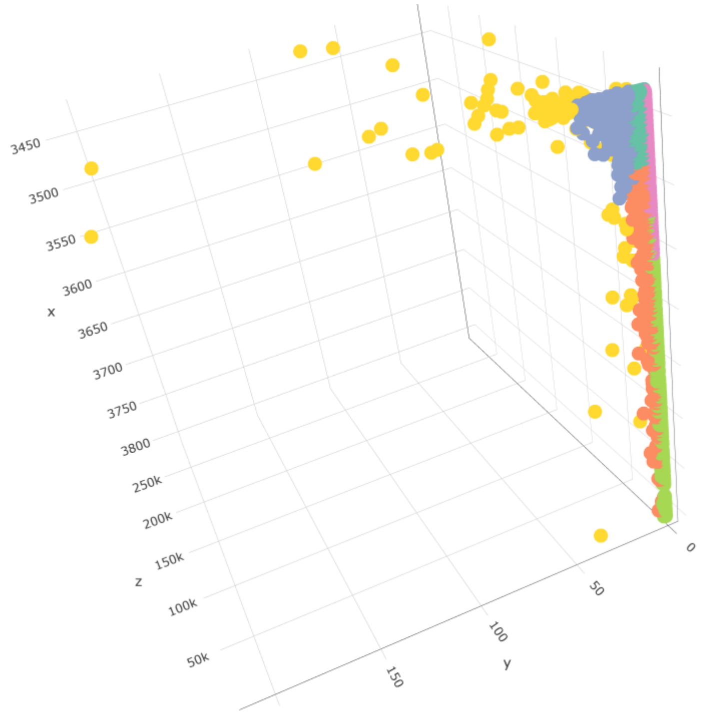
兩者做比較，我們會發現若分解出來的模型越複雜，可以使用的分群方式越靈活，也就可以看出GMM較K-means更加靈活的地方。其次，在這個套件中，判定的標準BIC並不是我們常見的BIC，而是由作者自己定義的，定義如下:

$$2logp(x|\mathcal{M}) + constant \approx 2l_{\mathcal{M}}(x,\hat{\theta}) - m_{\mathcal{M}}log(n) \equiv BIC$$

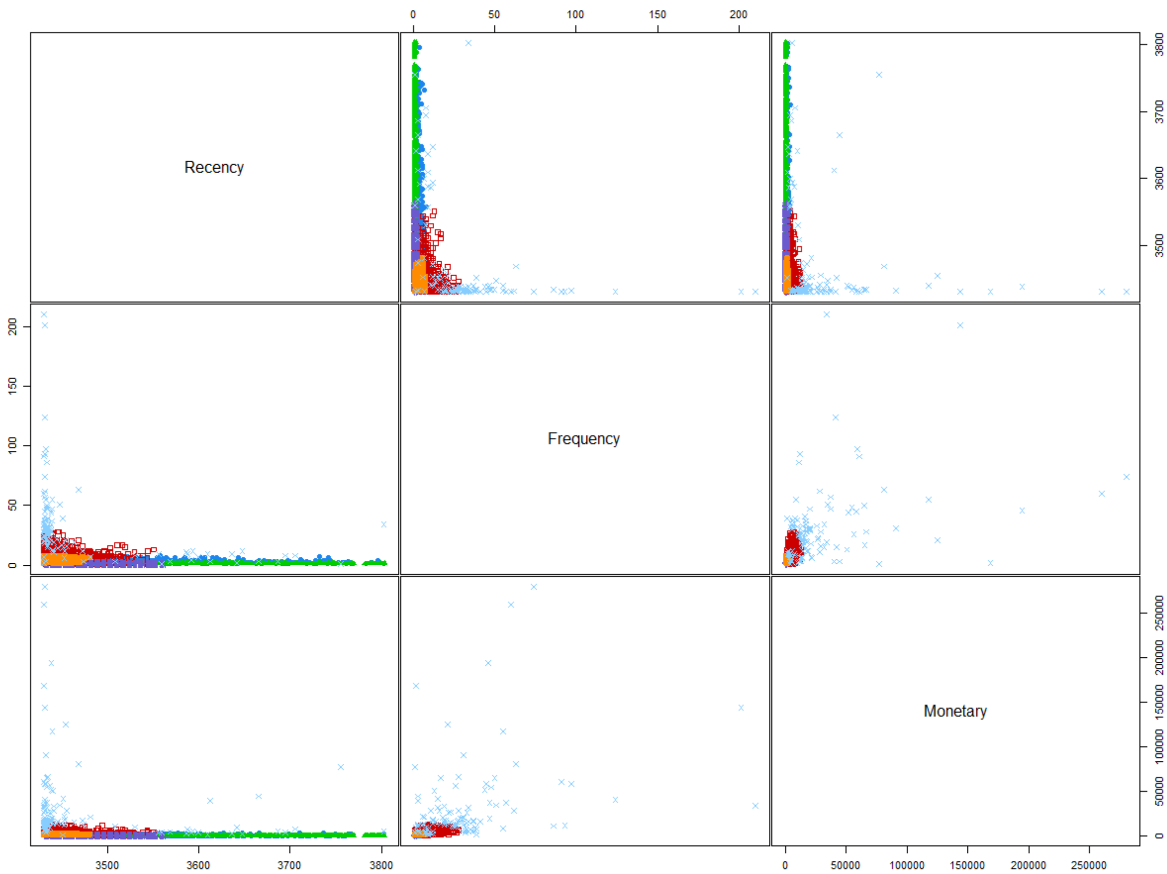
其中 $p(x|\mathcal{M})$ 為該資料在模型下的likelihood，$l_{\mathcal{M}}(x,\hat{\theta})$ 是最大混合的 log likelihood，$m_{\mathcal{M}}$ 是需要估計的獨立參數數量，而 n 位資料總數。因此，我們應該要找的是最大的BIC，而不是最小的。利用套件內建的程式，幫我們尋找最佳的模型以及群數:



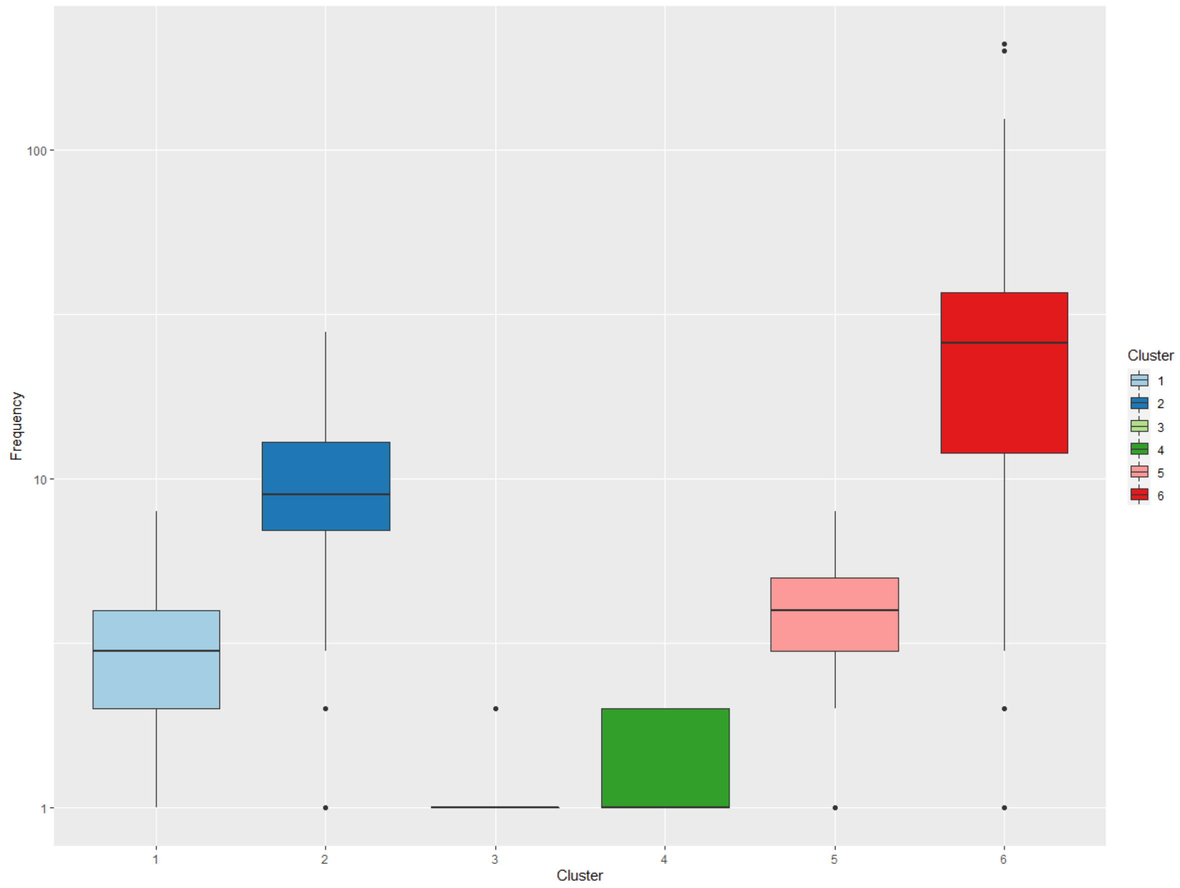
從這張圖，我們可以看到應該要選擇的模型為VVE。雖然我們看到最大的是，但是我選擇群數為6，因為從6之後，BIC都趨於穩定，所以我認為之後的BIC大小只是浮動而已，並且比較好解釋。當我們套入這兩個值進去模型分群，會得到以下結論:



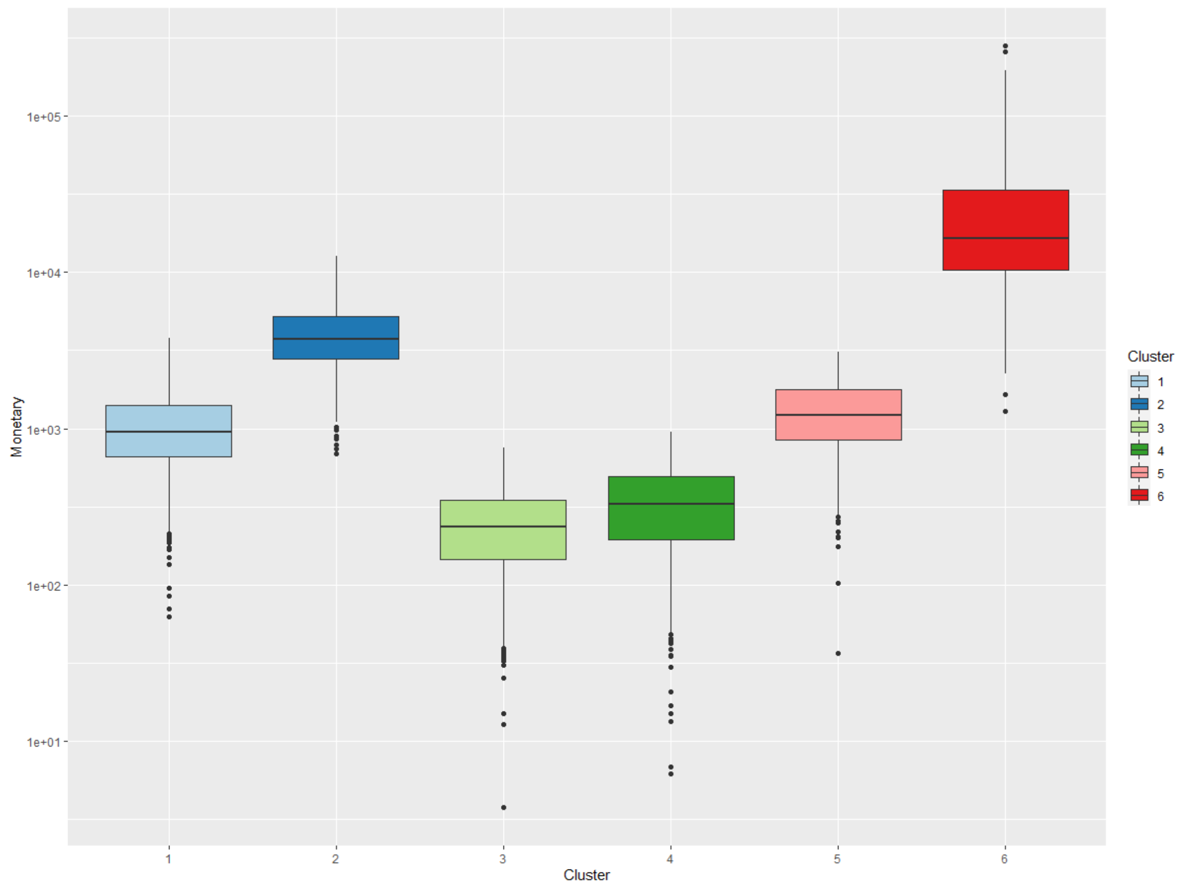
特別可以看到說，除了一組是收集outliers以外，特別的是GMM把集中的資料層層分群，這是K-means無法做到的狀況。再來，我們來看pair plot:



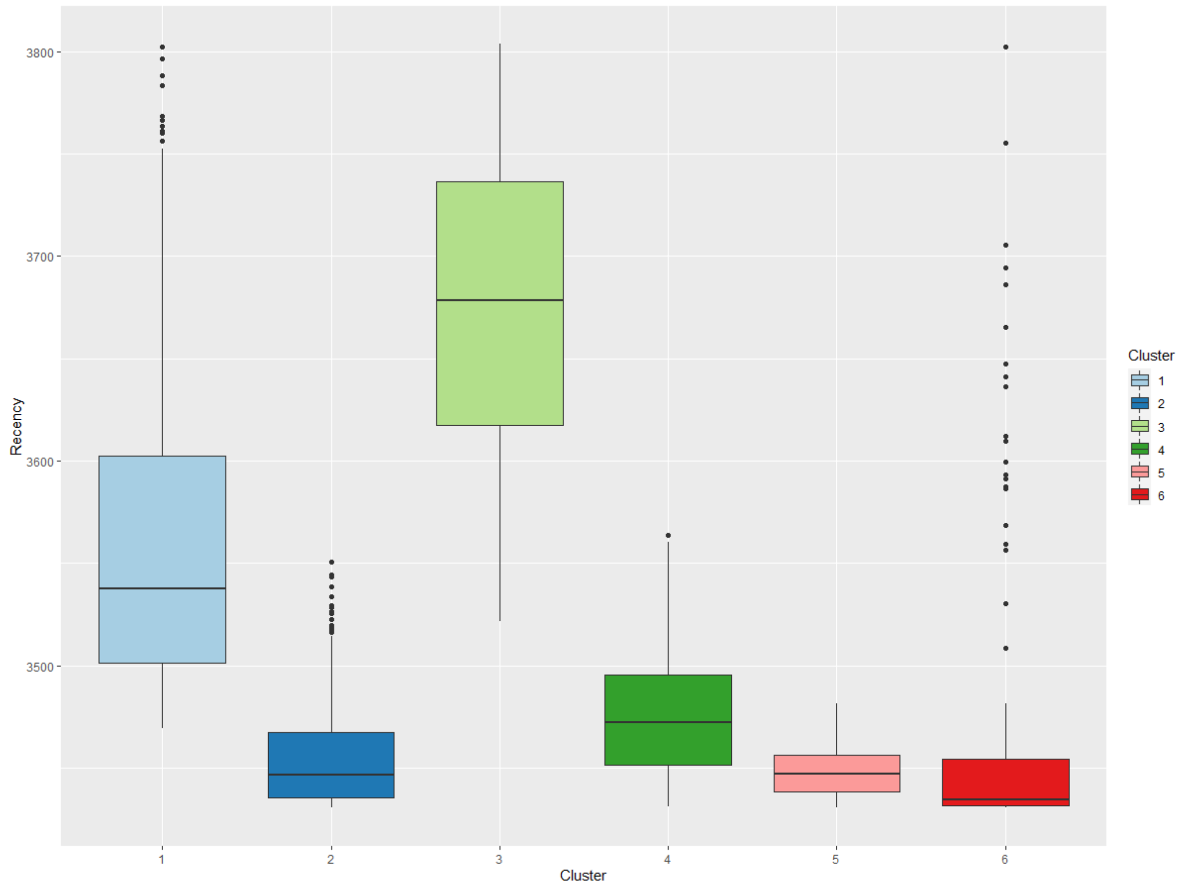
圖中點太集中了，所以無法判讀，就直接看分群後的各項變數狀況。首先，我們來看Frequency:



GMM的分群狀況比K-means更加詳細，但是由於y軸分散太廣了，因此我用log scale的y。其中，第6群是最常來的客人，第2群則是次常來的客人，而第3.4群就是最不常來的客人。再來，我們來看Monetary:



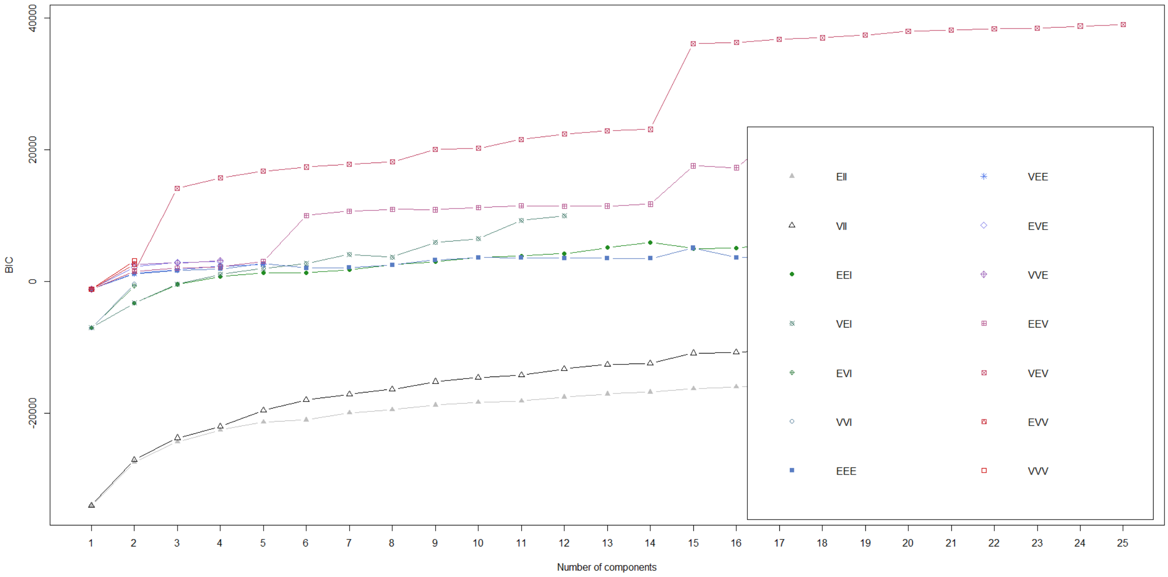
相同的，Monetary的y軸也是log scale的。其中第6群是貢獻最大的，其次是第2群，而第3.4群是貢獻最少的。最後來看Recency:



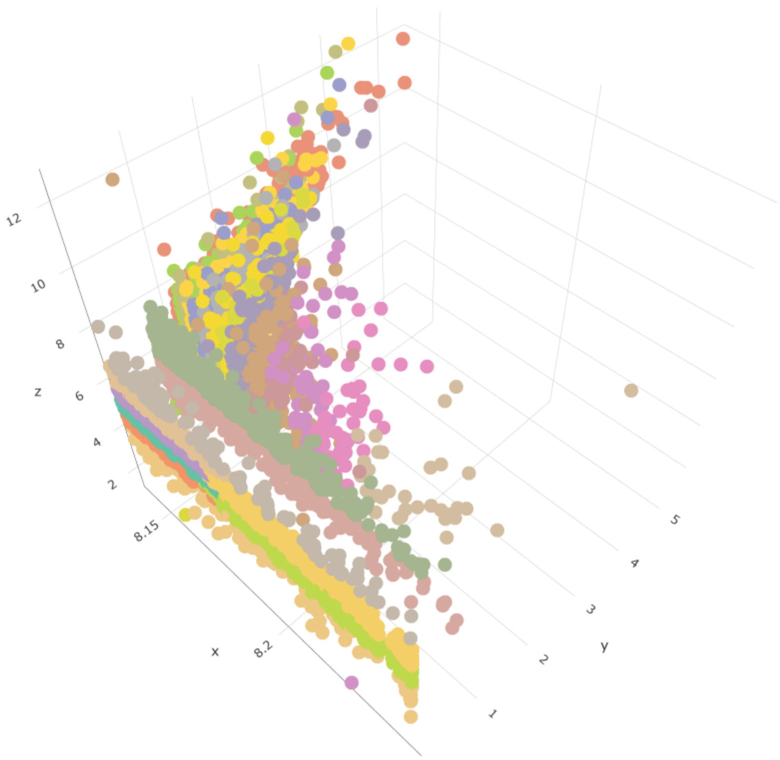
Recency的y軸就是原始的y軸。如同K-means一樣，Recency的分群相較於另外兩個變數，分群的效果並不好。從圖中，可以觀察第1.3群是近期沒來的一群，而最近來的則是2.5.6群。由於GMM分群更加細緻，因此比K-means可以分析更加深入，我們可以做以下表格:

分群編號	Frequency	Monetary	Recency	客群推測
1	中間	中間	次高	學生族群或是小資本客人
2	次高	次高	次低	極有潛力的客人
3	最低	最少	最高	已喪失的客人
4	次低	次少	中間	即將喪失的客人
5	中間	中間	次低	嘗新的新客人
6	最高	最高	最低	VIP級的死忠客人

相同的，GMM也要試一試資料經log轉換後的分群效果。以下是BIC圖:



由此圖中，我可以看到應該取得的模型是VEV，群組數量是25。以下是分群狀況:



雖然GMM分出來層層分群，但是由於群組數量過多，已經超過可以解釋範圍了，我認為就算會分群的好，也不會是我們所需的模型，因此選擇捨棄。

Conclusion

這次的報告主題是要利用GMM和K-means，建立RFM模型，並且幫線上購物網站分出不同類別的客人，進而找到目標客群。由於原始資料極度的不平衡，因此我們中間研究了資料轉換對於分群的效果如何，並且中間穿插了利用GMM和K-means對於log bivariate normal distribution以及bivariate normal distribution分群的狀況。經由此次的報告，我發現:

1. K-means在資料極度集中、或是資料群有交錯的情況下，無法分群分好的。此時就必須要利用資料轉換，協助K-means把資料散開。
2. 相反的，對於資料不平衡或是資料交錯複雜的情況下，GMM相較於K-means靈活很多，同時也會比K-means的分群更為細緻。
3. 雖然在simulation，我們使用了log normal的模型做分組資料的結論是，GMM在資料轉換後會較轉換前分群效果好，但這是資料本身是有模型的條件下成立的。在真實資料中，我們無法去檢驗是否有許多的模型交疊在其中。因此會導致GMM在資料轉換前的狀況下，分群較好。
4. 使用套件的時候，需要詳讀套件使用說明，不然就會產生作者的定義與自己不同的問題。

Appendix

```
#設定
library(knitr)
library(tidyverse)
library(lubridate)
library(mclust)
library(factoextra)
library(corrplot)
library(plotly)
setwd("C:\\Users\\stat-pc\\Desktop\\GraduateCourse\\Statistical
Computation\\midterm")
rawdata = read.csv("OnlineRetail.csv", header = T, stringsAsFactors = F)
data = rawdata %>% filter((!is.na(CustomerID)) & (Quantity >=0))

#資料前處理
mon = data %>% mutate(total = Quantity*UnitPrice) %>% group_by(CustomerID) %>%
summarize(Monetary = sum(total))
data$InvoiceDate = parse_date_time(data$InvoiceDate, order = "%d-%m-%Y %H:%M")
fre = data %>% group_by(CustomerID) %>% summarize(Frequency =
n_distinct(InvoiceNo))
maxdate = Sys.Date()
rec = data %>% mutate(date_diff = as.numeric(difftime(maxdate, InvoiceDate,
units = "day"))) %>% group_by(CustomerID) %>% summarise(Recency =
min(date_diff))
rfm = rec %>% inner_join(fre, by = "CustomerID") %>% inner_join(mon, by =
"CustomerID")
rfm_model_out = rfm %>% remove_rownames %>% column_to_rownames(var="CustomerID")
rfm_model_out = rfm_model_out[-which(rfm_model_out$Monetary==0),]

# EDA
plot_ly(x=rfm_model_out$Recency , y=rfm_model_out$Frequency,
z=rfm_model_out$Monetary, type="scatter3d", mode="markers")
rfm_gather = rfm_model_out %>% gather()
ggplot(rfm_gather, aes(x = value)) + geom_histogram() + facet_wrap(.~key, scales
= "free")
ggplot(rfm_gather, aes(x = value)) + geom_boxplot() + facet_wrap(.~key, scales =
"free") + coord_flip()
pairs(rfm_model_out)
ggplot(rfm_gather, aes(sample = value)) + facet_wrap(.~key, scales = "free") +
stat_qq()+stat_qq_line()
rfm_model_out_log = rfm_model_out %>% mutate_all(log)
plot_ly(x=rfm_model_out_log$Recency , y=rfm_model_out_log$Frequency,
z=rfm_model_out_log$Monetary, type="scatter3d", mode="markers")
rfm_gather = rfm_model_out_log %>% gather()
ggplot(rfm_gather, aes(x = value)) + geom_histogram() + facet_wrap(.~key, scales
= "free")
ggplot(rfm_gather, aes(x = value)) + geom_boxplot() + facet_wrap(.~key, scales =
"free") + coord_flip()
pairs(rfm_model_out_log)
corrplot(cor(rfm_model_out_log))
ggplot(rfm_gather, aes(sample = value)) + facet_wrap(.~key, scales = "free") +
stat_qq()+stat_qq_line()

#Simulation
gibbs<-function (n, rho,mu)
```

```
{
  mat <- matrix(ncol = 3, nrow = n)
  mat[1, ] <- mu
  for (i in 2:n) {
    x <- rnorm(1, mu[1]+rho[2:3,1]%*%solve(rho[2:3,2:3])%*%(mat[i-1,2:3]-
mu[2:3]), sqrt(rho[1,1]-rho[1,2:3]%*%solve(rho[2:3,2:3])%*%rho[1,2:3]))
    y <- rnorm(1, mu[2]+rho[c(1,3),2]%*%solve(rho[c(1,3),c(1,3)])%*%(mat[i-
1,c(1,3)]-mu[c(1,3)]), sqrt(rho[2,2]-
rho[2,c(1,3)]%*%solve(rho[c(1,3),c(1,3)])%*%rho[2,c(1,3)]))
    z = rnorm(1, mu[3]+rho[1:2,3]%*%solve(rho[1:2,1:2])%*%(mat[i-1,1:2]-
mu[1:2]), sqrt(rho[3,3]-rho[3,1:2]%*%solve(rho[1:2,1:2])%*%rho[3,1:2]))
    mat[i, ] <- c(x, y, z)
  }
  mat
}
rho = runif(6)
sigma1 = matrix(c(1,rho[1],rho[2],rho[1],2,rho[3],rho[2],rho[3],3), nrow = 3,
ncol = 3)
mu1 = matrix(c(1,1,0),nrow = 3, ncol = 1)
sigma2 = matrix(c(2,rho[4],rho[5],rho[4],3,rho[6],rho[5],rho[4],1), nrow = 3,
ncol = 3)
mu2 = matrix(c(1,0,1),nrow = 3, ncol = 1)
num = 1000
norm1 = gibbs(num, sigma1, mu1)
norm2 = gibbs(num, sigma2, mu2)
statistical_distance = function(data){
  y = data.frame(data) %>% mutate_all(function(x){x-mean(x)}) %>% data.matrix()
  ans = diag(y%*%solve(cov(y))%*%t(y))
  return(ans)
}
QQplot_chi = function(data){
  df = ncol(data)
  ans = statistical_distance(data)
  car::qqPlot(ans, dist = "chisq", df = df, ylab = "Sample Quantiles", main =
'chi-squared Q-Q plot for statistical distance')
}
QQplot_chi(norm1)
QQplot_chi(norm2)

temp = matrix(ncol = 3, nrow = num)
group = c()
for(i in 1:num){
  x = runif(1)
  if(x<0.4){
    temp[i,] = norm1[i,]
    group = c(group, 1)
  }else{
    temp[i,] = norm2[i,]
    group = c(group,2)
  }
}
sim_exp = exp(temp)
plot_ly(x=sim_exp[,1] , y=sim_exp[,2], z=sim_exp[,3], type="scatter3d",
mode="markers", color = as.factor(group))
pairs(sim_exp,col = as.factor(group))
mod = Mclust(sim_exp, G = 2)
plot.Mclust(mod, what = "classification", main = F, addEllipses = F)
```

```
plot_ly(x=sim_exp[,1] , y=sim_exp[,2], z=sim_exp[,3], type="scatter3d",
mode="markers", color = as.factor(mod$classification))
pairs(sim_exp, col = mod$classification, pch = 19, main = "K-means")
title(main = "GMM")
mod = kmeans(sim_exp, centers = 2)
pairs(sim_exp, col = mod$cluster, pch = 19, main = "K-means")
plot_ly(x=sim_exp[,1] , y=sim_exp[,2], z=sim_exp[,3], type="scatter3d",
mode="markers", color = as.factor(mod$cluster))

gmm_accu = c()
k_accu = c()
for(i in 1:1000){
  num = 1000
  norm1 = gibbs(num, sigma1, mu1)
  norm2 = gibbs(num, sigma2, mu2)
  temp = matrix(ncol = 3, nrow = num)
  group = c()
  for(i in 1:num){
    x = runif(1)
    if(x<0.4){
      temp[i,] = norm1[i,]
      group = c(group, 1)
    }else {
      temp[i,] = norm2[i,]
      group = c(group, 2)
    }
  }
  sim_exp = exp(temp)
  gmm = Mclust(sim_exp, G = 2)
  temp = sum(group == gmm$classification)/1000
  if(temp <= 0.5){
    gmm_accu = c(gmm_accu, 1-temp)
  }else{
    gmm_accu = c(gmm_accu, temp)
  }
  kmod = kmeans(sim_exp, centers = 2)
  temp = sum(group==kmod$cluster)/1000
  if(temp <= 0.5){
    k_accu = c(k_accu, 1-temp)
  }else{
    k_accu = c(k_accu, temp)
  }
}
par(mfrow = c(1,2))
boxplot(gmm_accu, xlab = "GMM")
boxplot(k_accu, xlab = "K-means")

sim = log(sim_exp)
plot_ly(x=sim[,1] , y=sim[,2], z=sim[,3], type="scatter3d", mode="markers",
color = as.factor(group))
pairs(sim,col = as.factor(group))
mod = Mclust(sim, G = 2)
plot.Mclust(mod, what = "classification", main = F, addEllipses = F)
plot_ly(x=sim[,1] , y=sim[,2], z=sim[,3], type="scatter3d", mode="markers",
color = as.factor(mod$classification))
mod = kmeans(sim, centers = 2)
pairs(sim, col = mod$cluster, pch = 19, main = "K-means")
```

```r
plot_ly(x=sim[,1] , y=sim[,2], z=sim[,3], type="scatter3d", mode="markers",
color = as.factor(mod$cluster))

gmm_accu = c()
k_accu = c()
for(i in 1:1000){
  num = 1000
  norm1 = gibbs(num, sigma1, mu1)
  norm2 = gibbs(num, sigma2, mu2)
  temp = matrix(ncol = 3, nrow = num)
  group = c()
  for(i in 1:num){
    x = runif(1)
    if(x<0.4){
      temp[i,] = norm1[i,]
      group = c(group, 1)
    }else{
      temp[i,] = norm2[i,]
      group = c(group, 2)
    }
  }
  sim = temp
  gmm = Mclust(sim, G = 2)
  temp = sum(group == gmm$classification)/1000
  if(temp <= 0.5){
    gmm_accu = c(gmm_accu, 1-temp)
  }else{
    gmm_accu = c(gmm_accu, temp)
  }
  kmod = kmeans(sim_exp, centers = 2)
  temp = sum(group==kmod$cluster)/1000
  if(temp <= 0.5){
    k_accu = c(k_accu, 1-temp)
  }else{
    k_accu = c(k_accu, temp)
  }
}
par(mfrow = c(1,2))
boxplot(gmm_accu, xlab = "GMM")
boxplot(k_accu, xlab = "K-means")

# Case study
# k-means
fviz_nbclust(rfm_model_out, FUNcluster = kmeans, method = "wss")
mod = kmeans(rfm_model_out, centers = 3)
pairs(rfm_model_out, col = mod$cluster, pch = 19)
fviz_nbclust(rfm_model_out_log, FUNcluster = kmeans, method = "wss")
mod = kmeans(rfm_model_out_log, centers = 3)
pairs(rfm_model_out_log, col = mod$cluster, pch = 19)
rfm_box = cbind(rfm_model_out_log, mod$cluster) %>% mutate(Cluster =
as.factor(mod$cluster))
ggplot(rfm_box, aes(x = Cluster, y = Recency, group = Cluster, fill = Cluster))
+ geom_boxplot() + scale_fill_brewer(palette="Pastel1")
ggplot(rfm_box, aes(x = Cluster, y = Frequency, group = Cluster, fill =
Cluster)) + geom_boxplot() + scale_fill_brewer(palette="Pastel1")
ggplot(rfm_box, aes(x = Cluster, y = Monetary, group = Cluster, fill = Cluster))
+ geom_boxplot() + scale_fill_brewer(palette="Pastel1")
```

```r
plot_ly(x=rfm_model_out_log$Recency , y=rfm_model_out_log$Frequency,
z=rfm_model_out_log$Monetary, type="scatter3d", mode="markers", color =
as.factor(mod$cluster))
plot_ly(x=rfm_model_out$Recency , y=rfm_model_out$Frequency,
z=rfm_model_out$Monetary, type="scatter3d", mode="markers", color =
as.factor(mod$cluster))


# GMM
mod = Mclust(rfm_model_out, G = 1:25)
plot.Mclust(mod, what = "BIC")
abline(v = 6, lty = 8)
mod = Mclust(rfm_model_out, G = 6)
plot.Mclust(mod, what = "classification",addEllipses = F)
rfm_box = cbind(rfm_model_out, mod$classification) %>% mutate(Cluster =
as.factor(mod$classification))
ggplot(rfm_box, aes(x = Cluster, y = Recency, group = Cluster, fill = Cluster))
+ geom_boxplot() + scale_fill_brewer(palette="Paired")
ggplot(rfm_box, aes(x = Cluster, y = Frequency, group = Cluster, fill =
Cluster)) + geom_boxplot() + scale_fill_brewer(palette="Paired") +
scale_y_log10()
ggplot(rfm_box, aes(x = Cluster, y = Monetary, group = Cluster, fill = Cluster))
+ geom_boxplot() + scale_fill_brewer(palette="Paired") + scale_y_log10()
mod = Mclust(rfm_model_out_log, G = 1:25)
plot.Mclust(mod, what = "BIC")
plot.Mclust(mod, what = "classification",addEllipses = F)
plot_ly(x=rfm_model_out$Recency , y=rfm_model_out$Frequency,
z=rfm_model_out$Monetary, type="scatter3d", mode="markers", color =
as.factor(mod$classification))
plot_ly(x=rfm_model_out_log$Recency , y=rfm_model_out_log$Frequency,
z=rfm_model_out_log$Monetary, type="scatter3d", mode="markers", color =
as.factor(mod$classification))
```