BIOSTAT 285 Spring 2020 Project 3

K-means clustering on TCGA Gene Expression Data

Nan Liu

TCGA gene expression data TCGA contains the expression of a random sample of 2000 genes for 563 patients from three cancer subtypes: Basal (Basal), Luminal A (LumA), and Luminal B (LumB). In this report, we aim to distinguish Luminal A samples from Luminal B by K-means clustering.

The true subtypes of patients (stored in Subtypes) are masked (in Gene) during our unsupervised learning pipelines and will be accessed only when assessing the clustering relevance.

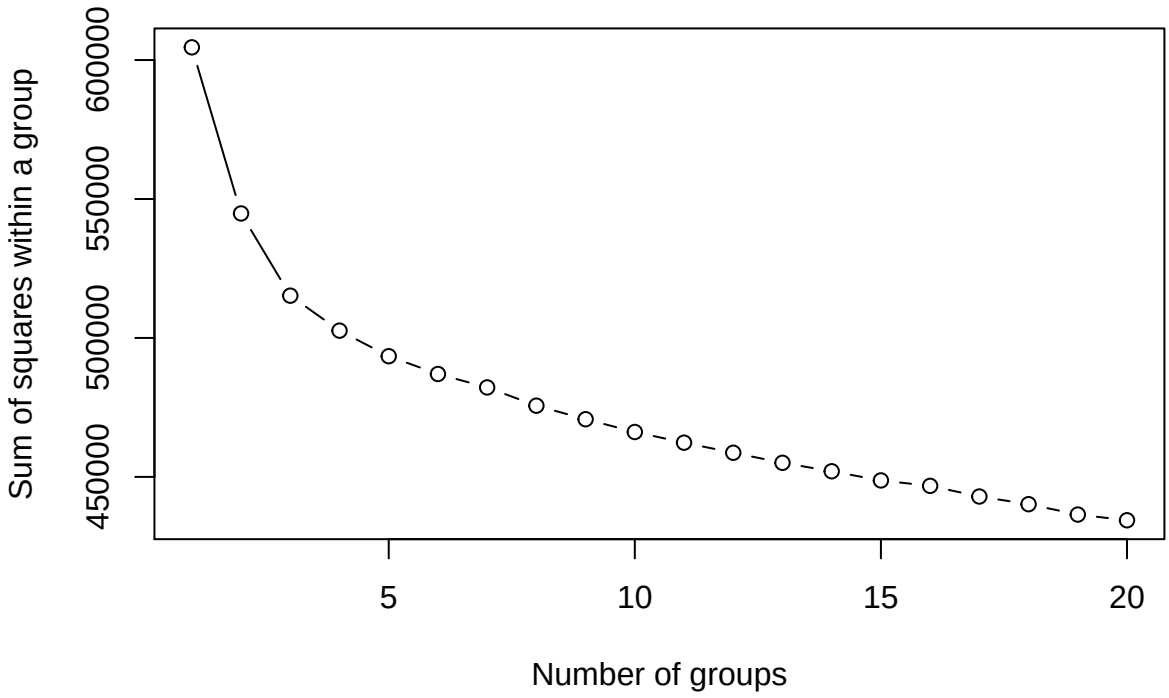
Since dimensionality might introduce noise masking the true clustering information, we also apply dimension reduction techniques (PCA and MDS) to explore whether our clustering results might be improved. We try being an oracle knowing in advance what would be the best two-dimensional representation of the expression data, and see whether the K-means clustering better identifies subtypes. We also apply prior knowledge that the first PC of the expression data is irrelevant to the subtypes, and explore the clustering results after denoising.

First, let's import the data and store the subtypes of tissue and the gene expression data

```
TCGA <- read.csv("TCGA_sample_2.txt", header = TRUE)

Subtypes <- TCGA[ ,1]
#Gene <- as.matrix(TCGA[,-1])
Gene <- TCGA[,-1]
```

Then we run K-means for K from 1 to 20 and plot the associated within cluster sum of squares (WSSs).

```
wssplot <- function(data, nc=20, seed=123){
               wss <- (nrow(data)-1)*sum(apply(data,2,var))
               for (i in 2:nc){
                    set.seed(seed)
                    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
                plot(1:nc, wss, type="b", xlab="Number of groups",
                     ylab="Sum of squares within a group")}

wssplot(Gene, nc = 20)
```




```
#can also use the fviz_nbclust function
library(factoextra)
```

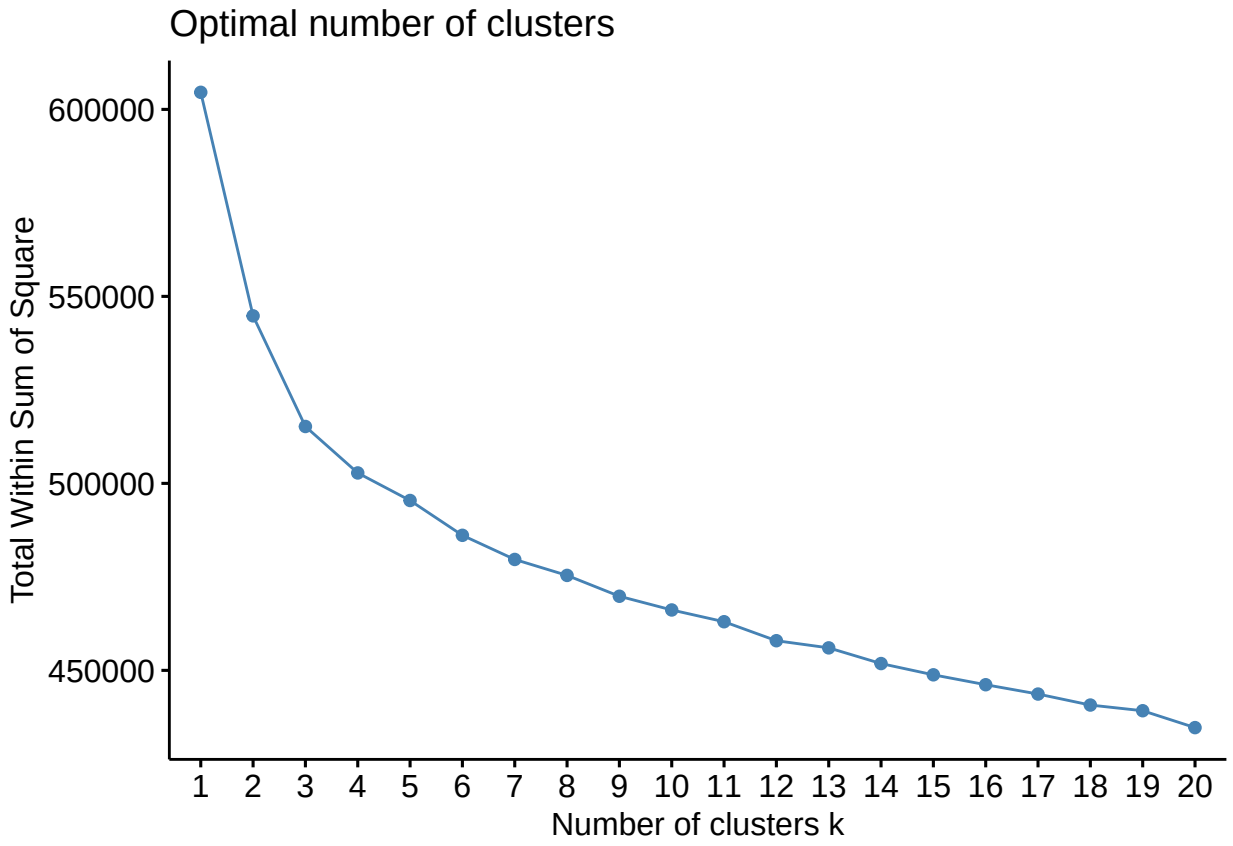
```
## Warning: package 'factoextra' was built under R version 3.6.2
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_nbclust(Gene, kmeans, k.max = 20, method = "wss")
```



From the plot we can identify K = 3 as the elbow in WSS reduction along with K. Hence, we conclude that the optimal number of cluster is 3 in this dataset.

K-means with K= 3

Apply K-means with $K = 3$ to the `Gene` dataset:

```
set.seed(1)
new.result <- kmeans(Gene, 3, nstart = 20)
#confusion matrix
table(true = Subtypes, pred = new.result$cluster)
```

```
##        pred
## true      1   2   3
##   Basal 101   1   0
##   LumA    0 192 117
##   LumB    0  27 125
```

The first cluster only includes `Basal` type samples. The second resulting cluster includes 0.4% `Basal` type, 12.3% `LumB` type and 87.3% `LumA` type. The third resulting type includes 48.3% `LumA` type and 51.7% `LumB` type.We might conclude that we did not do a good job distinguishing `LumA` from `LumB`

Denosie by PCA

In order to obtain better clustering result, we apply PCA to the `Gene` dataset.

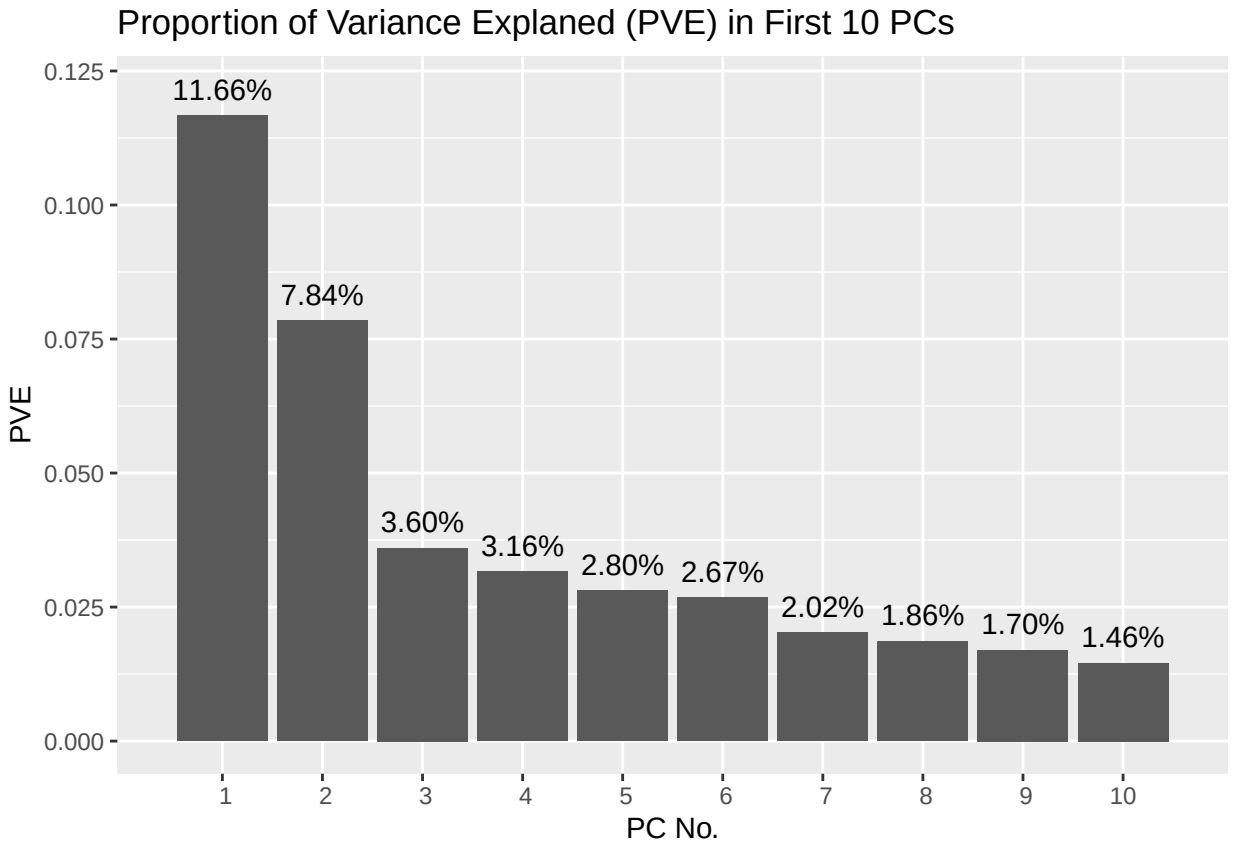
```
gene.pca <- prcomp(~., Gene)
```

Analyze the Proportion of Variance Explained (PVE) and Cumulative Proportion of Variance Explained (CPVE) for the first 10 PCs

```
gene.pca.pve <- summary(gene.pca)$importance[2,]
gene.pca.cpve <- summary(gene.pca)$importance[3,]
```

```
npc <- 10

ggplot(data = data.frame(m = 1:npc, pve = gene.pca.pve[1:npc]),
       aes(x = m, y = pve)) +
  geom_col() +
  geom_text(aes(label = sprintf("%.2f%%", 100 * gene.pca.pve[1:npc]), y = pve + 0.005)) +
  scale_x_discrete(name = "PC No.", limits = 1:npc) +
  ylab("PVE") +
  ggtitle("Proportion of Variance Explaned (PVE) in First 10 PCs")
```

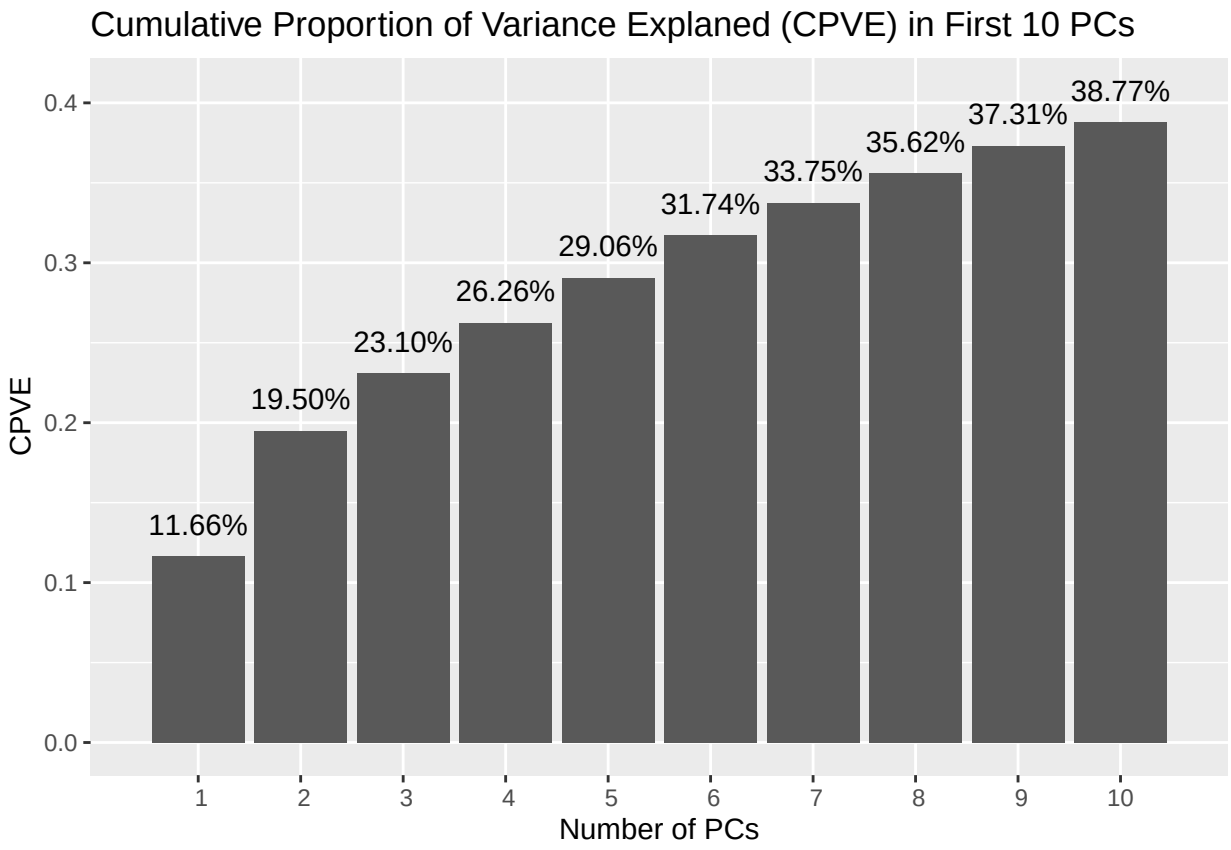
```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?
```




```
ggplot(data = data.frame(m = 1:npc, cpve = gene.pca.cpve[1:npc]),
       aes(x = m, y = cpve)) +
  geom_col() +
```

```
  geom_text(aes(label = sprintf("%.2f%%", 100 * gene.pca.cpve[1:npc]), y = cpve + 0.02)) +
  scale_x_discrete(name = "Number of PCs", limits = 1:npc) +
  ylab("CPVE") +
  ggtitle("Cumulative Proportion of Variance Explaned (CPVE) in First 10 PCs")
```

```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?
```



From the plot we see, the first 10 pricipal components account for 38.77% variance in total.

Try plotting some more first 5 PC combinations in order to find a pair of PCs that appear to separate all three subtypes well:

```
npc <- 5
plot.subcap <- paste("(", outer(paste("PC", 1:npc, sep = ""), paste("PC", 1:npc, sep = ""), paste,sep =
plot.sep <- character()
for(i in 1:npc)
  plot.sep <- c(plot.sep, rep(" ", i - 1), "\\newline")
plot.sep <- plot.sep[-length(plot.sep)]
```

```
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 3.6.2
```

```
## Loading required package: usethis
```

```
## Warning: package 'usethis' was built under R version 3.6.2
```

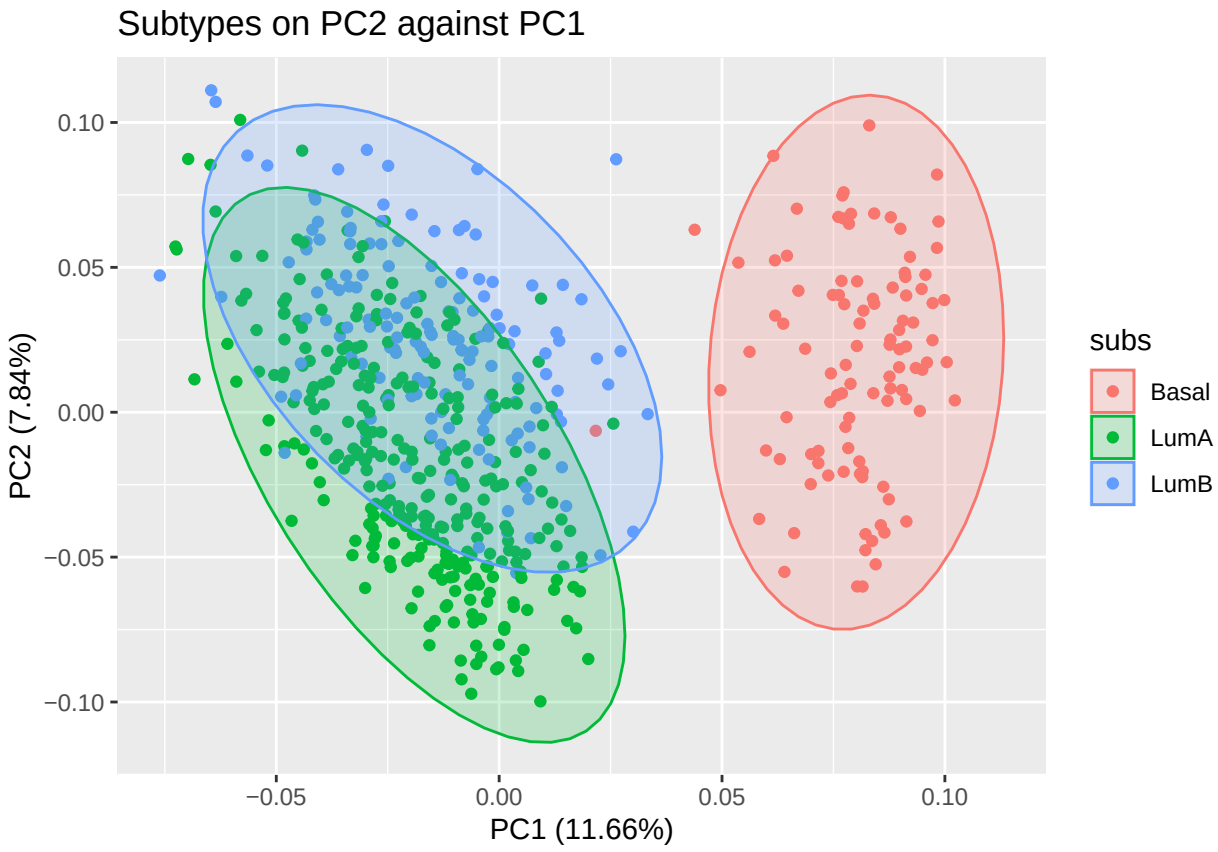
```
#install_github('sinhrks/ggfortify')
library(ggfortify)
```

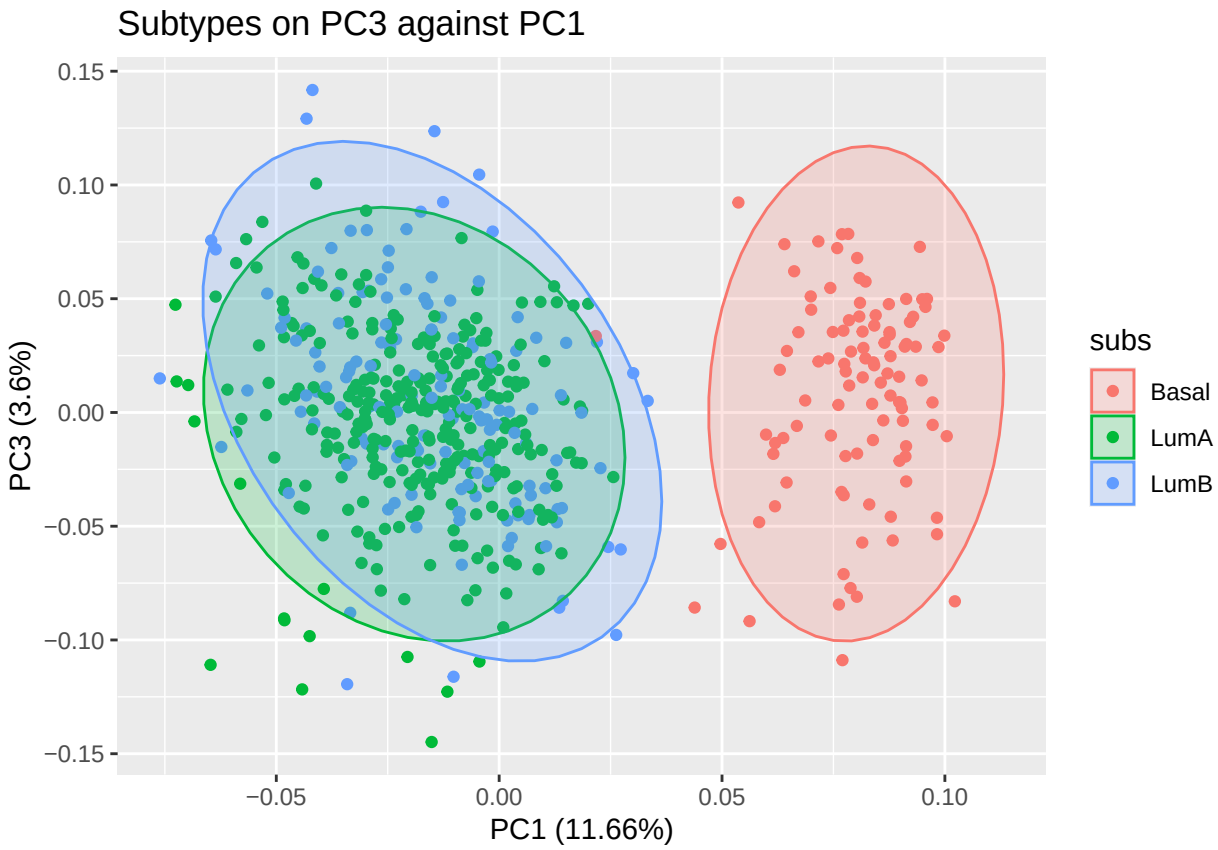
```
## Warning: package 'ggfortify' was built under R version 3.6.2
```

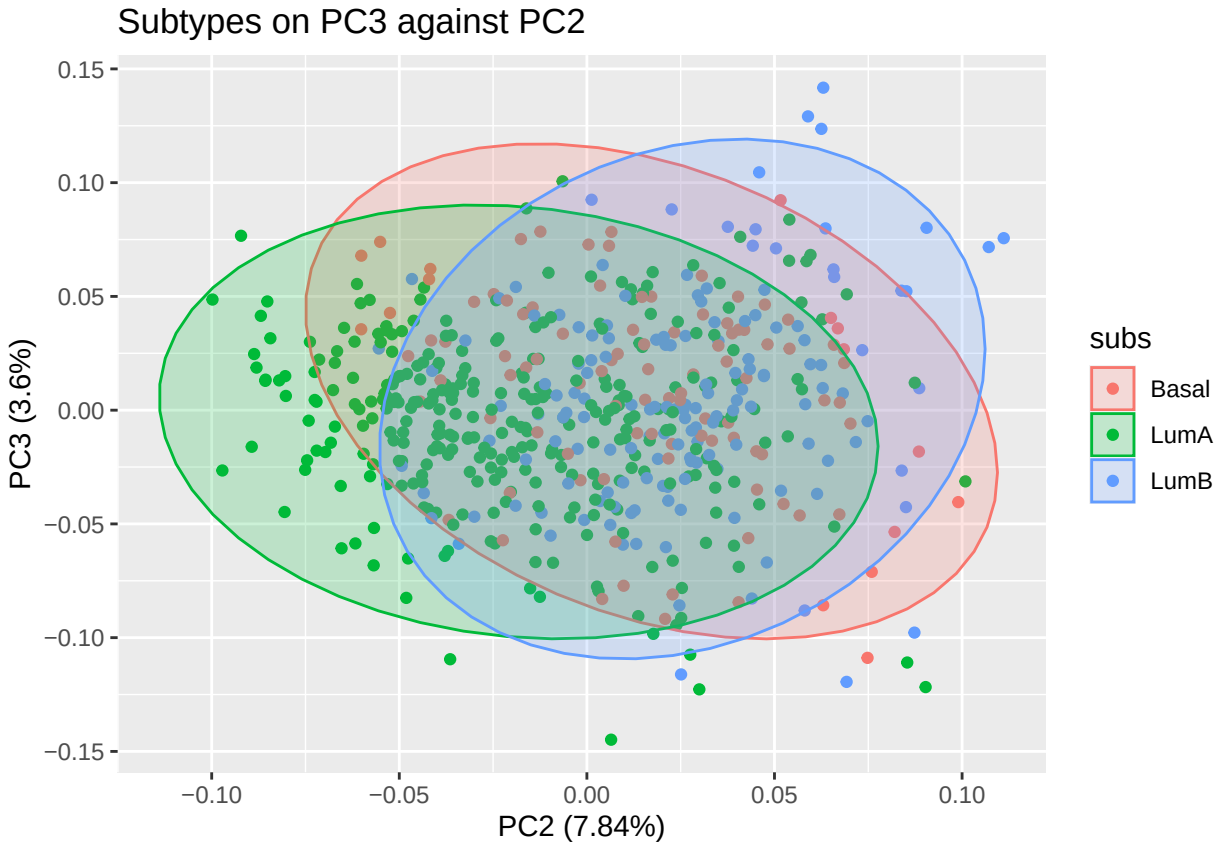
```
library(ggplot2)
```

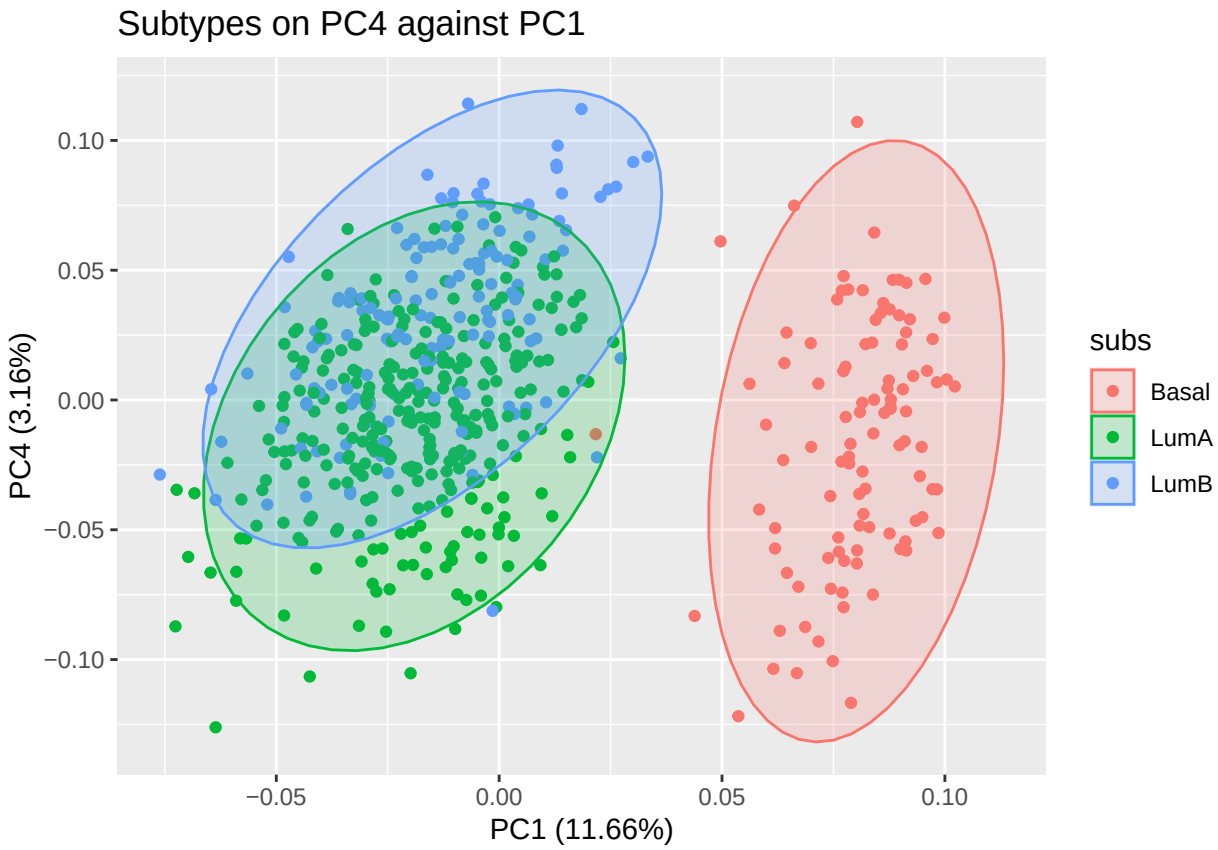
```
for(i in 2:npc)
  for(j in 1:(i-1))
      print(autoplot(gene.pca, data = TCGA,  x = j, y = i,
                     col = "subs", frame = T, frame.type = "norm") +
              ggtitle(paste("Subtypes on ",
                      "PC", i, " against ", "PC", j,
                      sep = "")))
```

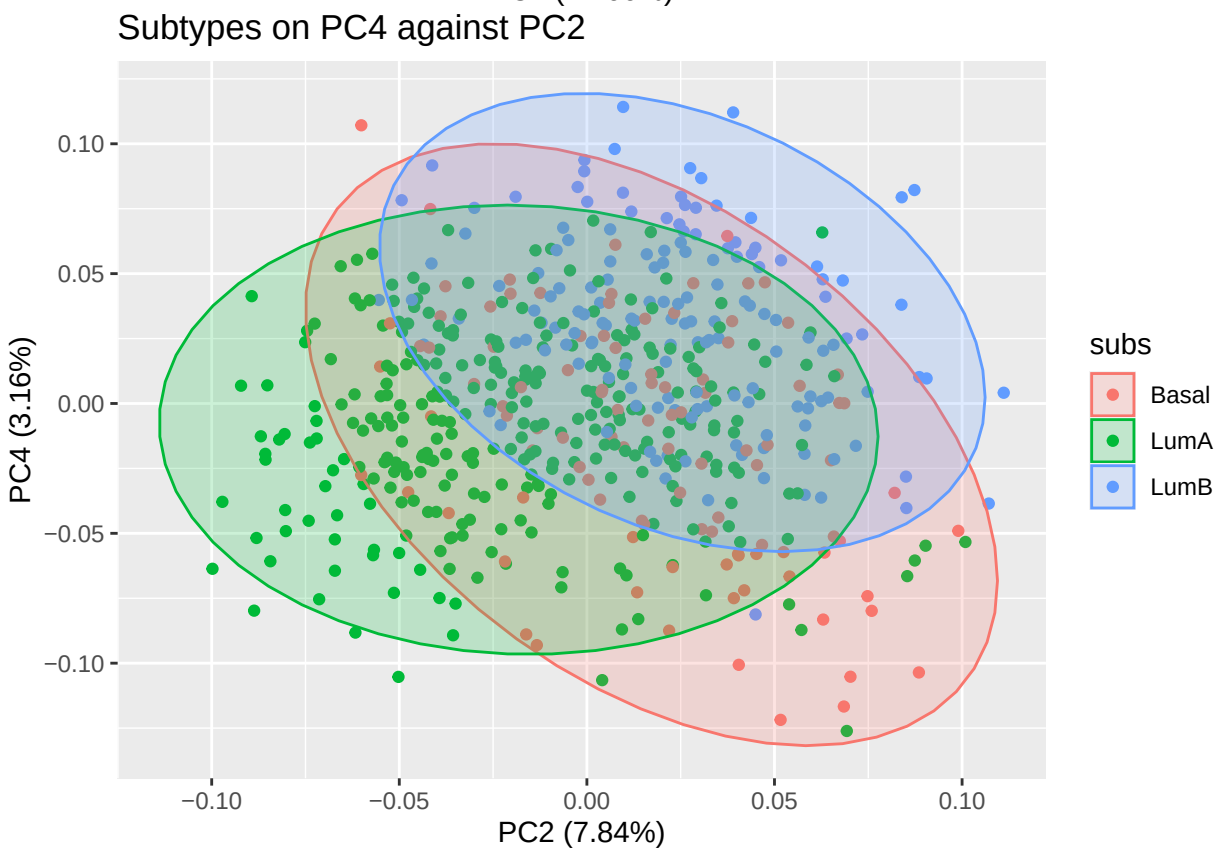
```
## Warning: `select_()` is deprecated as of dplyr 0.7.0.
## Please use `select()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

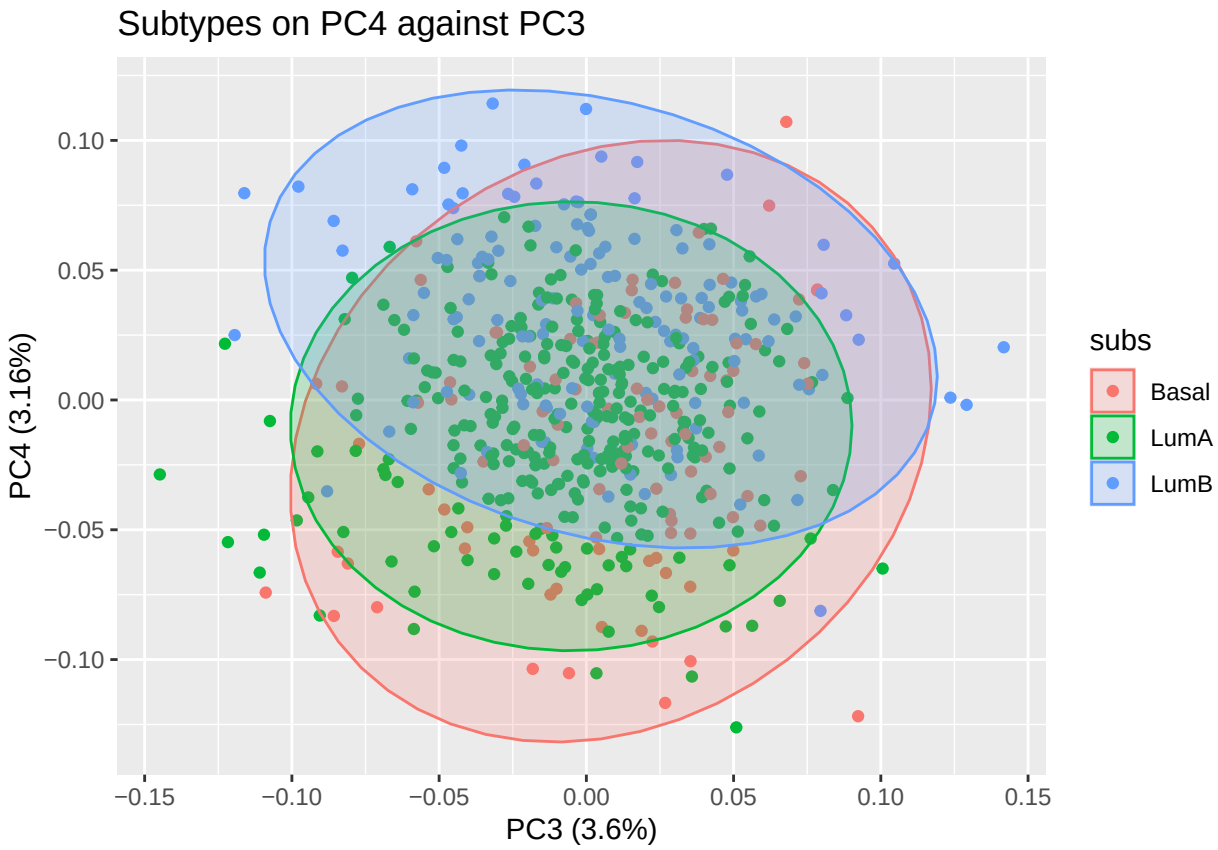


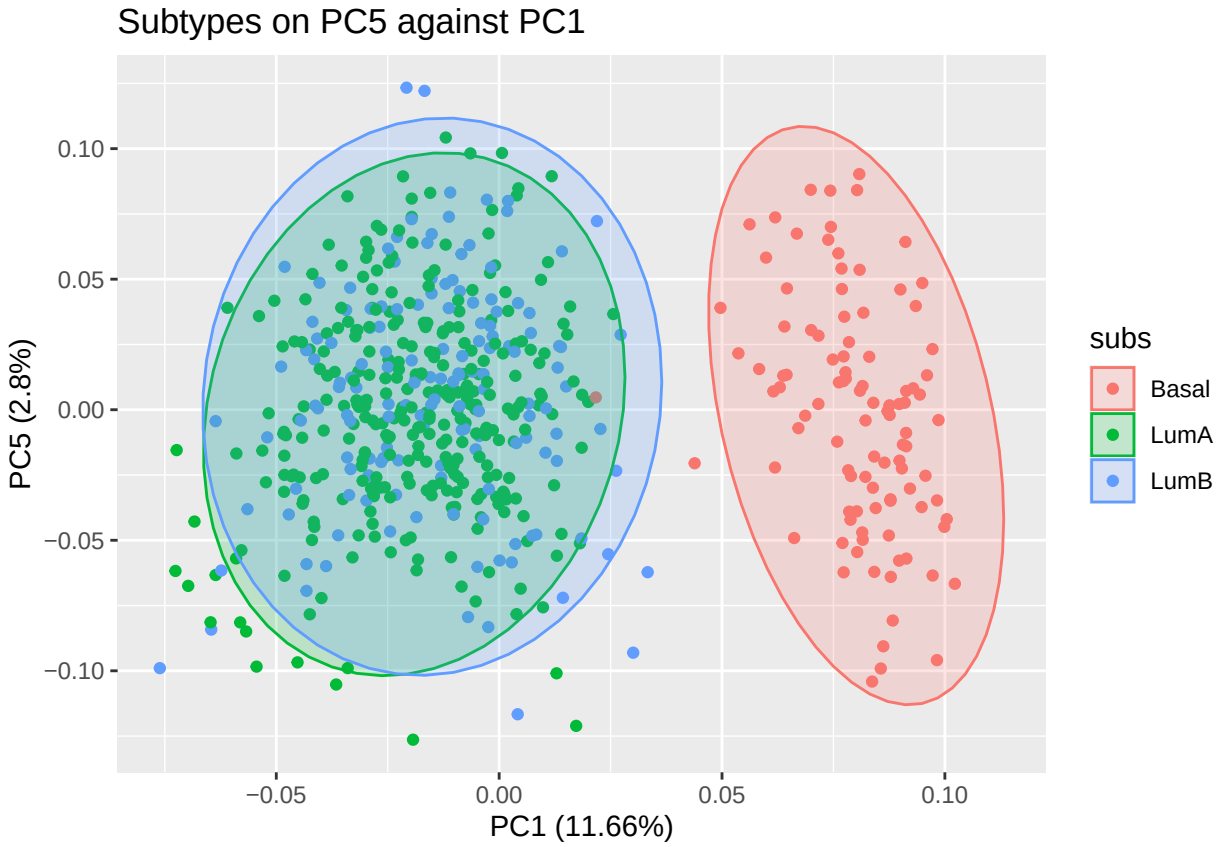

Subtypes on PC3 against PC1
PC3 (3.6%)
PC1 (11.66%)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.05
0.00
0.05
0.10
subs
Basal
LumA
LumB

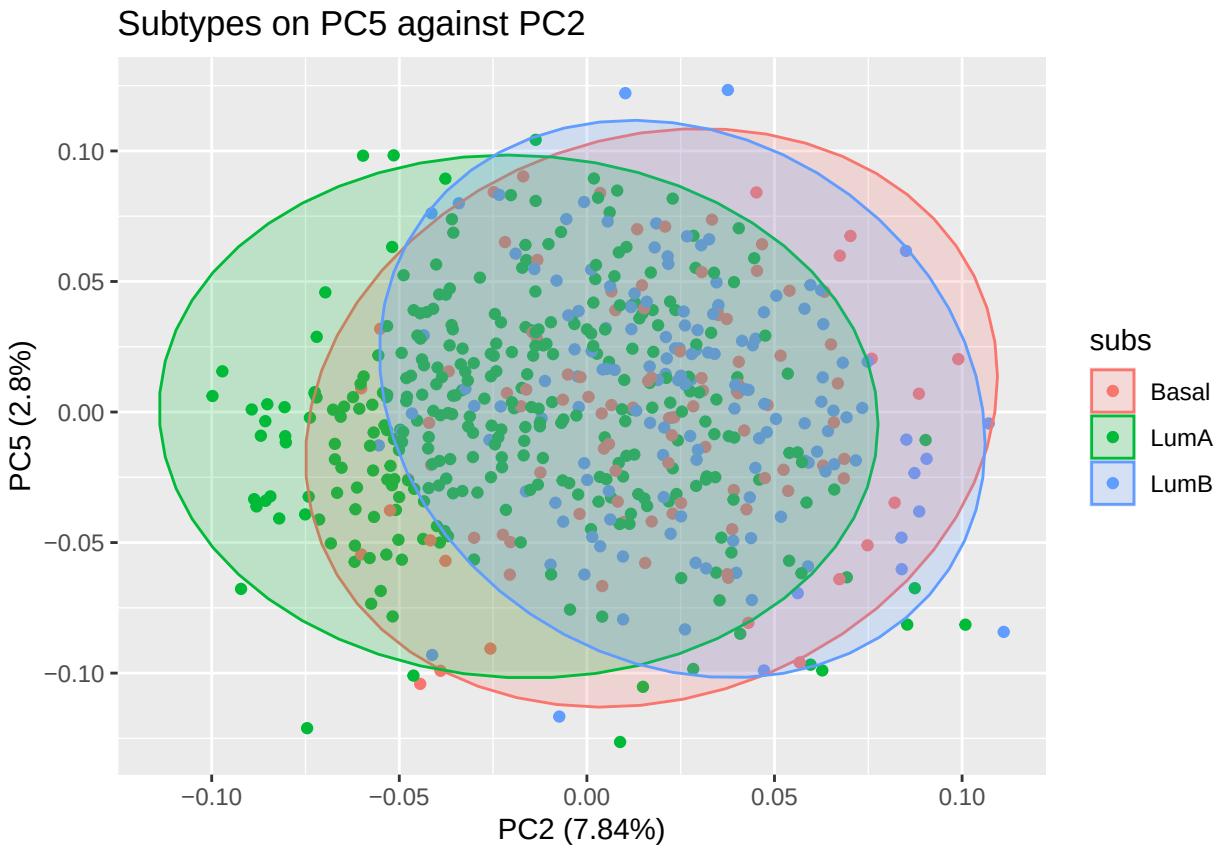

Subtypes on PC3 against PC2
PC3 (3.6%)
PC2 (7.84%)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.10
−0.05
0.00
0.05
0.10
subs
Basal
LumA
LumB

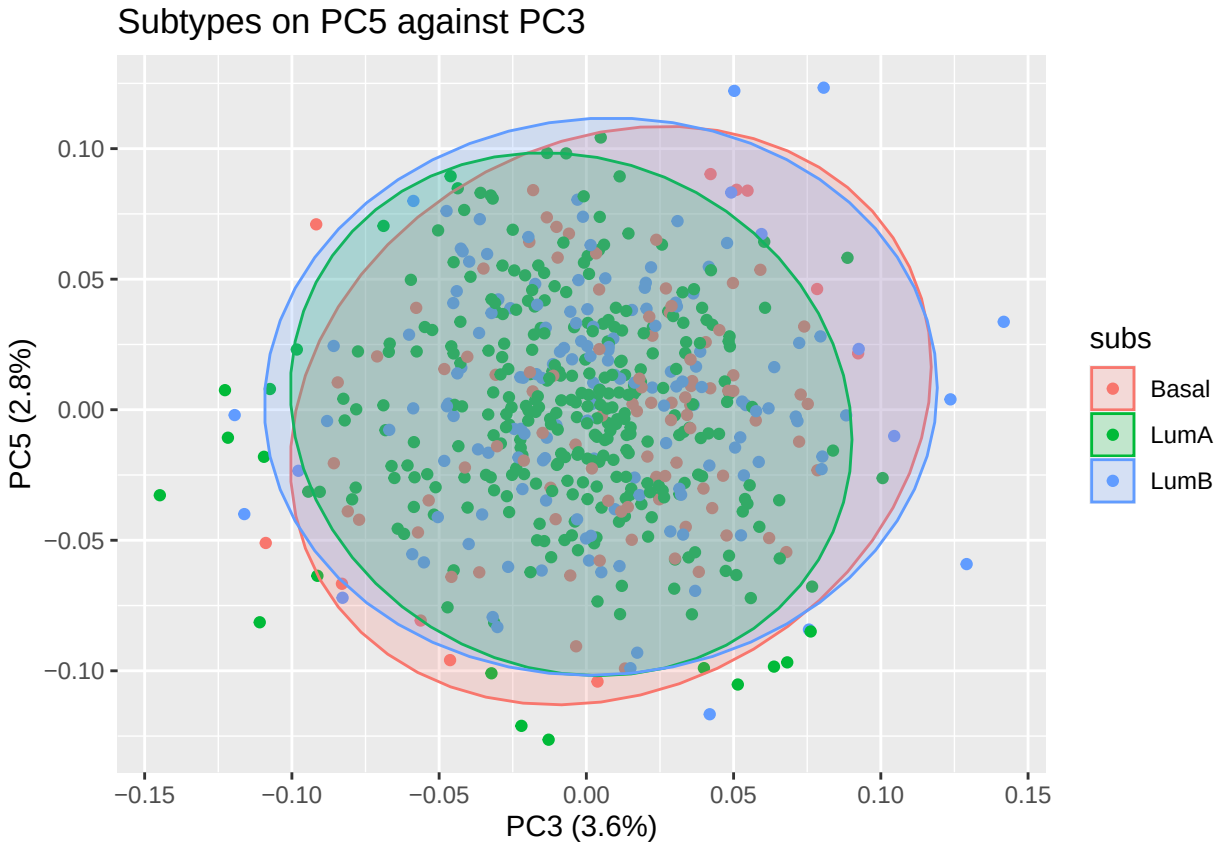

Subtypes on PC4 against PC1
PC4 (3.16%)
PC1 (11.66%)
0.10
0.05
0.00
−0.05
−0.10
−0.05
0.00
0.05
0.10
subs
Basal
LumA
LumB

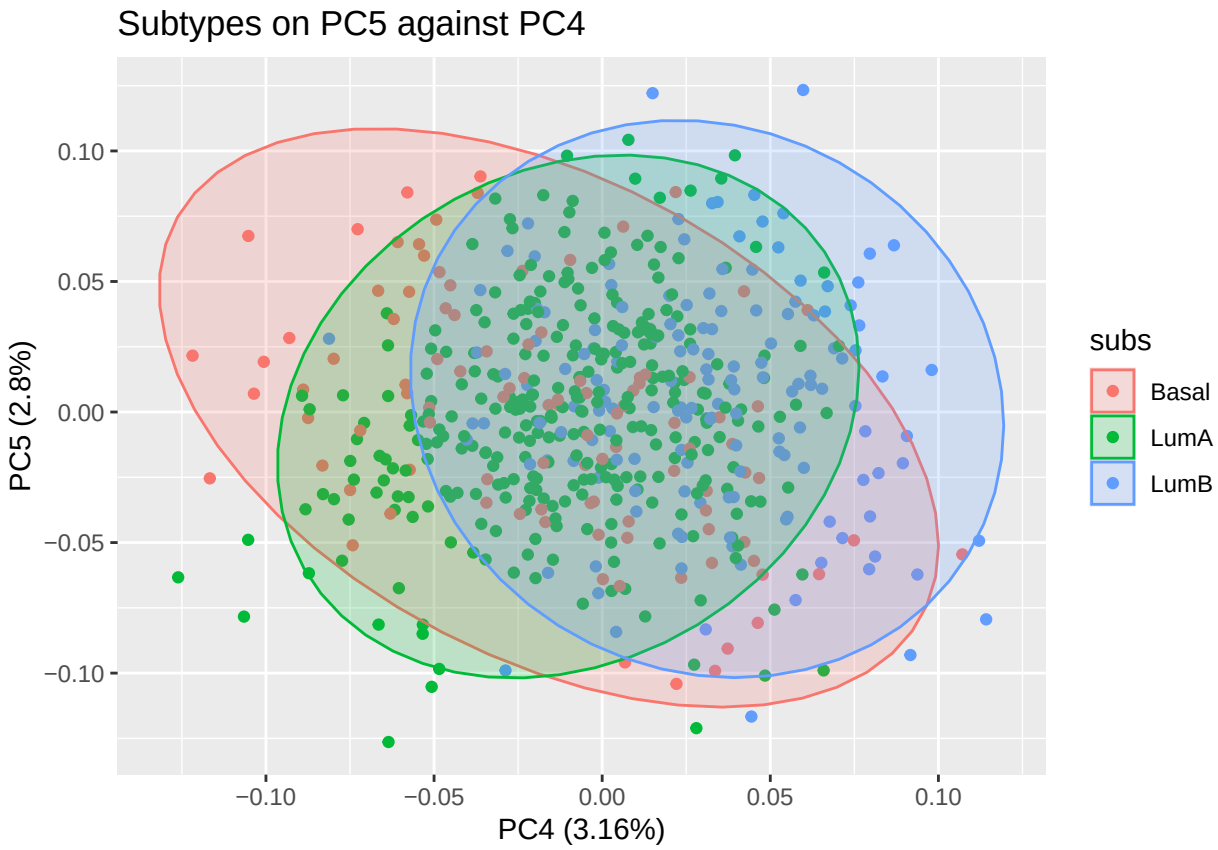

Subtypes on PC4 against PC2
PC4 (3.16%)
PC2 (7.84%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
subs
Basal
LumA
LumB


Subtypes on PC4 against PC3
PC4 (3.16%)
PC3 (3.6%)
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
subs
Basal
LumA
LumB


Subtypes on PC5 against PC1
PC5 (2.8%)
PC1 (11.66%)
0.10
0.05
0.00
−0.05
−0.10
−0.05
0.00
0.05
0.10
subs
Basal
LumA
LumB


Subtypes on PC5 against PC2
PC5 (2.8%)
PC2 (7.84%)
0.10
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
0.10
subs
Basal
LumA
LumB


Subtypes on PC5 against PC3
PC5 (2.8%)
PC3 (3.6%)
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
subs
Basal
LumA
LumB



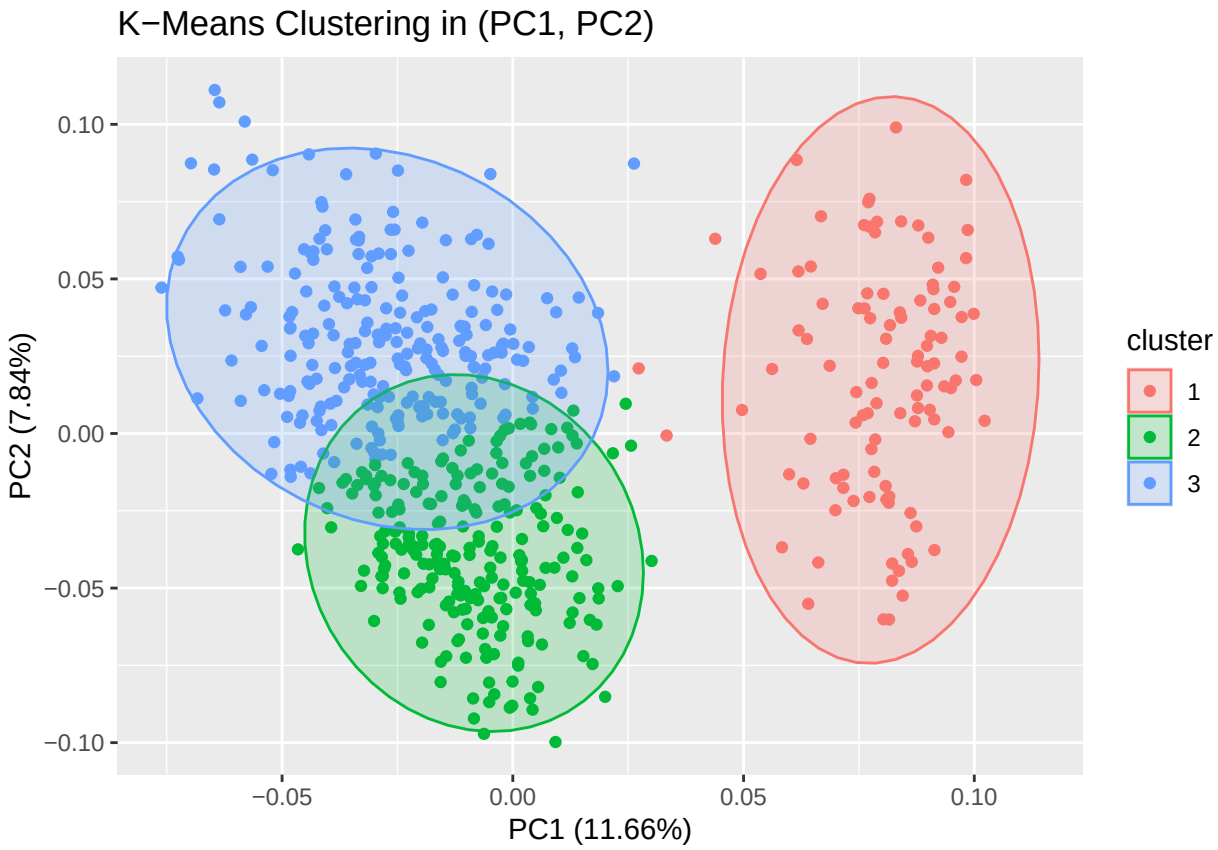
From the plots we conclude that the pair PC1 with PC2 and the pair PC1 with PC4 may provide possible separations of LumA and LumB.

Then we perform K-means with $K = 3$ on the pair PC1 with PC2 and the pair PC1 with PC4.

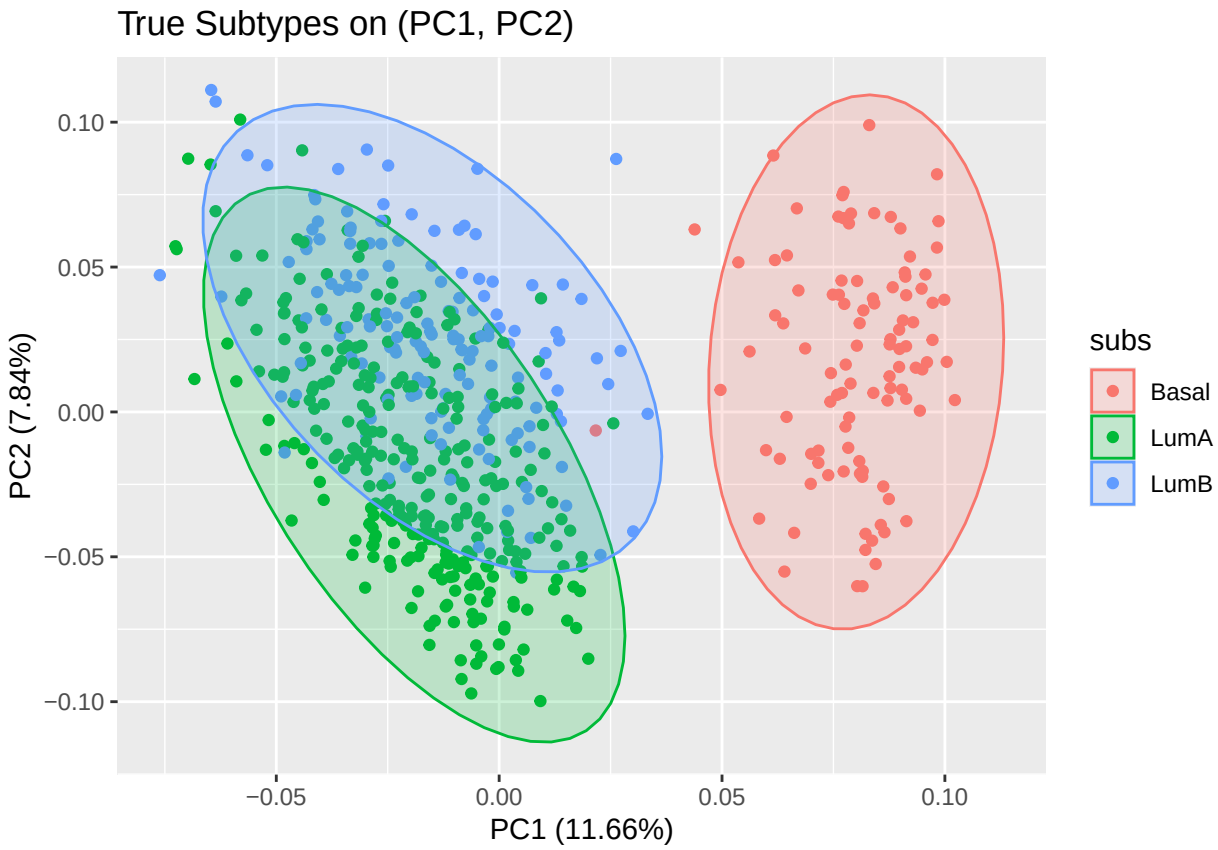
```
set.seed(1)
gene.pc12.kmeans <- kmeans(gene.pca$x[,c(1,2)], 3, nstart = 20)
gene.pc14.kmeans <- kmeans(gene.pca$x[,c(1,4)], 3, nstart = 20)
```

Plot the cluster:

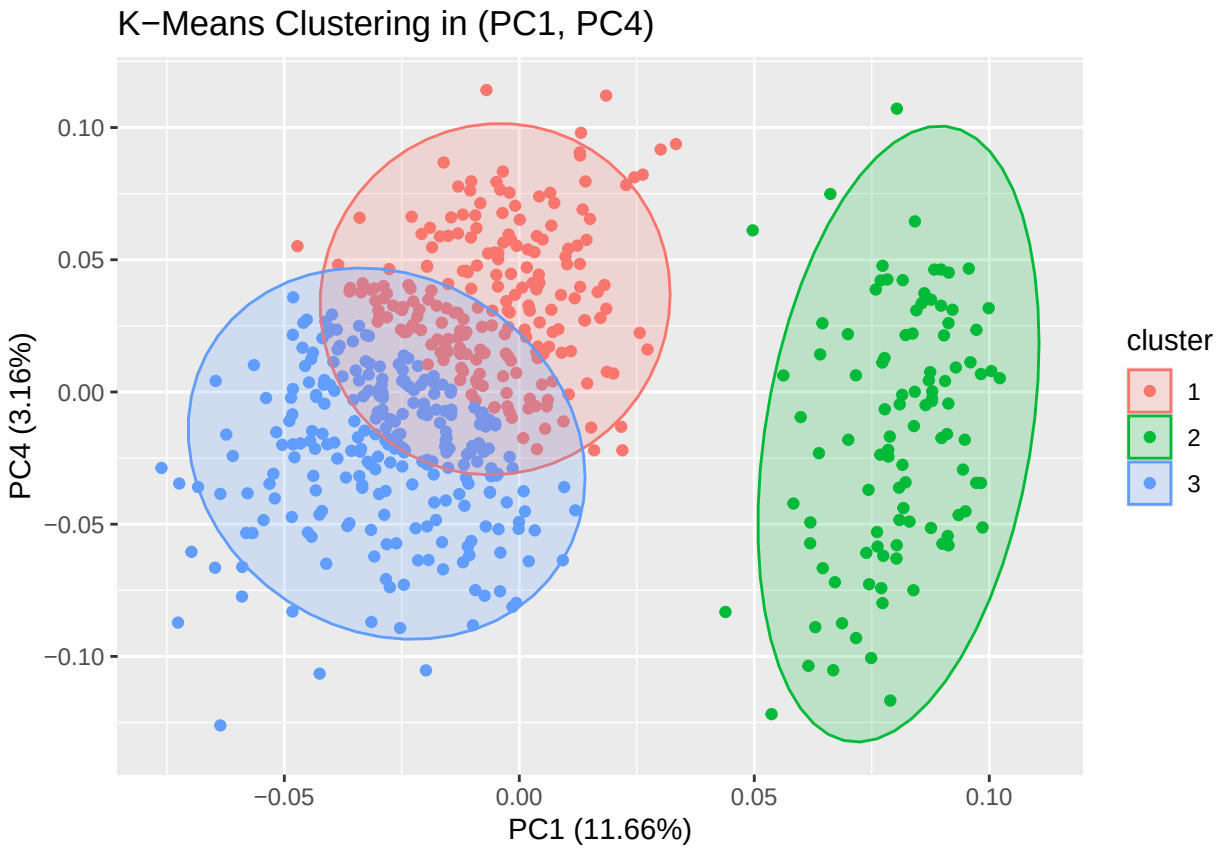
```
autoplot(gene.pca, data = data.frame(Gene, cluster = factor(gene.pc12.kmeans$cluster)),
         col = "cluster", frame = T, frame.type = "norm") +
  ggtitle("K-Means Clustering in (PC1, PC2)")
```



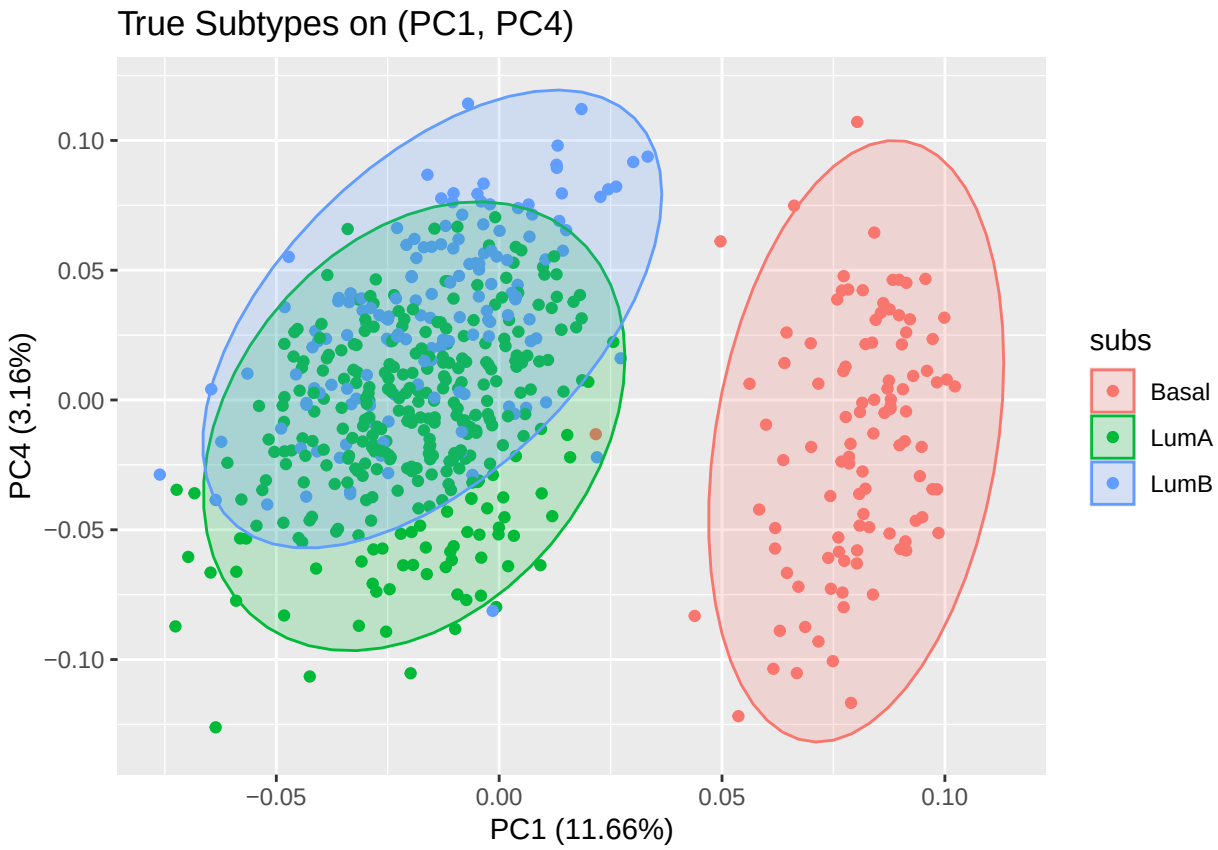

```
autoplot(gene.pca, data = TCGA, col = "subs", frame = T, frame.type = "norm") +
  ggtitle("True Subtypes on (PC1, PC2)")
```




```
autoplot(gene.pca, data = data.frame(Gene, cluster = factor(gene.pc14.kmeans$cluster)),
         y = 4, col = "cluster", frame = T, frame.type = "norm") +
  ggtitle("K-Means Clustering in (PC1, PC4)")
```




```
autoplot(gene.pca, data = TCGA, x = 1, y = 4,
         col = "subs", frame = T, frame.type = "norm") +
  ggtitle("True Subtypes on (PC1, PC4)")
```



Report the confusion matrix:

```
gene.pc12.kmeans.confusion <- table(Subtype = Subtypes,
                                    Cluster = gene.pc12.kmeans$cluster)
print("Confusion Matrix of K-Means in (PC1, PC2)", quote = FALSE)
```

```
## [1] Confusion Matrix of K-Means in (PC1, PC2)
```

```
addmargins(gene.pc12.kmeans.confusion)
```

```
##         Cluster
## Subtype   1   2   3 Sum
##   Basal 101   1   0 102
##   LumA    0 195 114 309
##   LumB    2  33 117 152
##   Sum   103 229 231 563
```

```
gene.pc14.kmeans.confusion <- table(Subtype = Subtypes,
                                    Cluster = gene.pc14.kmeans$cluster)
print("Confusion Matrix of K-Means in (PC1, PC4)", quote = FALSE)
```

```
## [1] Confusion Matrix of K-Means in (PC1, PC4)
```

```
addmargins(gene.pc14.kmeans.confusion)
```

```
##         Cluster
## Subtype   1   2   3 Sum
##   Basal   1 101   0 102
##   LumA  104   0 205 309
##   LumB  107   0  45 152
##   Sum   212 101 250 563
```

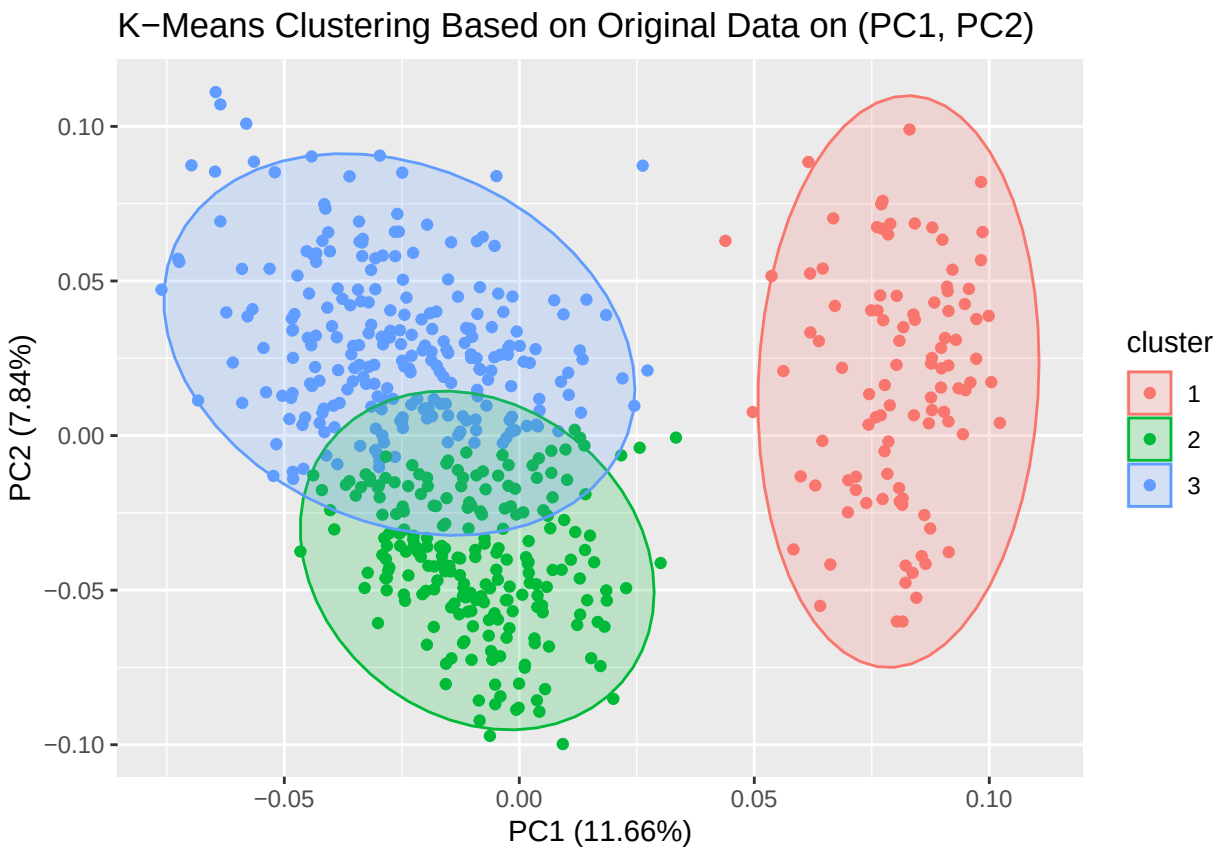
The 101 Basal type samples are separated to cluster1, while the other 1 is separated to cluster2. So we conclude that Basal type can be separated from the other 2 types clearly.

When we perform K-means with K= 3 on the first 2 PCs, the 195 LumA type samples are separated to cluster2, while the other 114 are separated to cluster3. The 2 LumB type samples are separated to cluster1, the 33 LumB type samples are separated to cluster 2, while the other 117 are separated to cluster3. We still cannot distinguish LumA type samples from LumB type samples.

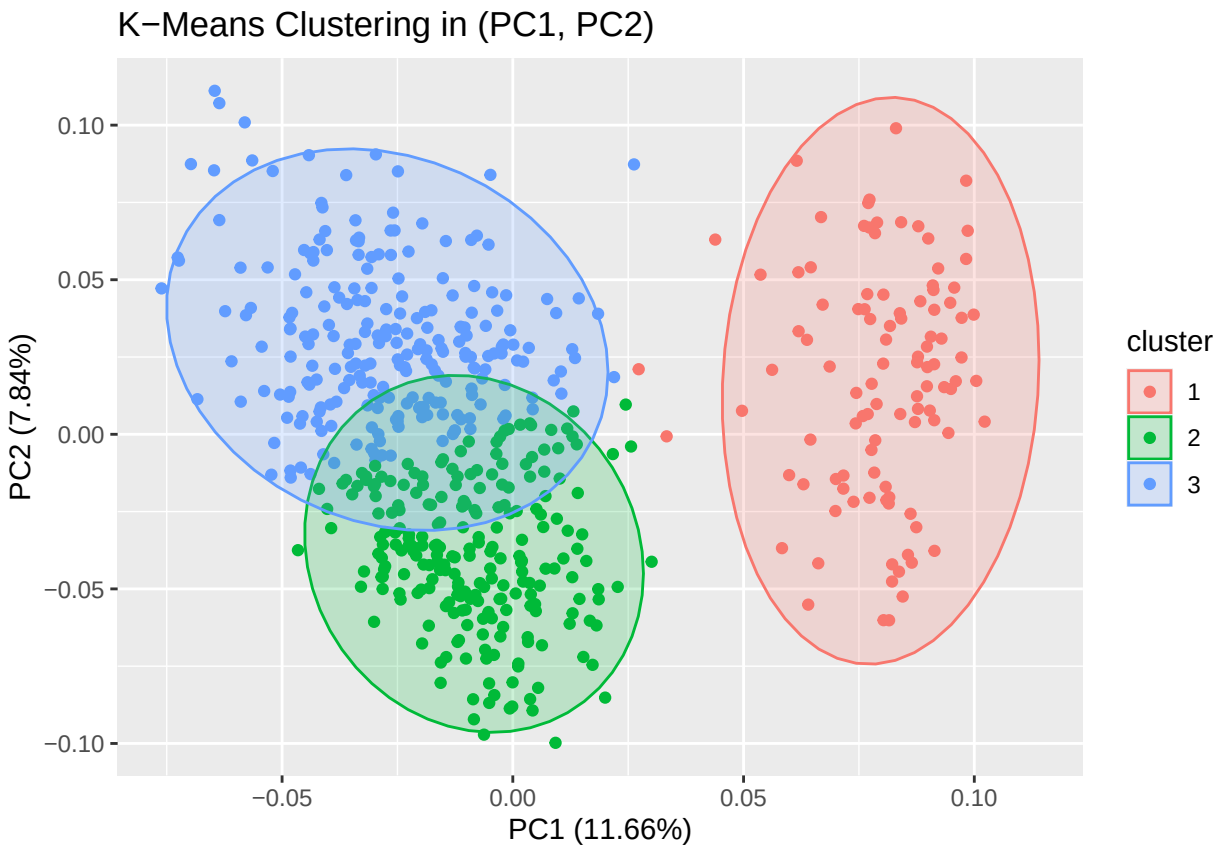
The other pair can not separate LumA type samples and LumB samples either.

Let's plot the clusters based on original data and the first 2 PCs.

```
gene.kmeans <- kmeans(Gene, 3, nstart = 20)
autoplot(gene.pca, data = data.frame(Gene, cluster = factor(gene.kmeans$cluster)),
         col = "cluster", frame = T, frame.type = "norm") +
  ggtitle("K-Means Clustering Based on Original Data on (PC1, PC2)")
```



```
autoplot(gene.pca, data = data.frame(Gene, cluster = factor(gene.pc12.kmeans$cluster)),
         col = "cluster", frame = T, frame.type = "norm") +
  ggtitle("K-Means Clustering in (PC1, PC2)")
```



We found that they look similarly. This is because the first 2 PCs can explain 19.5% of the variance, which wel captures the Euclidean distance information.

Denoise by MDS

In order to get better 2-dimensional representations, now we apply MDS with various metrics and non-metric MDS to `Gene` to obtain 2-dimensional representations.

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.6.2
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------------
```

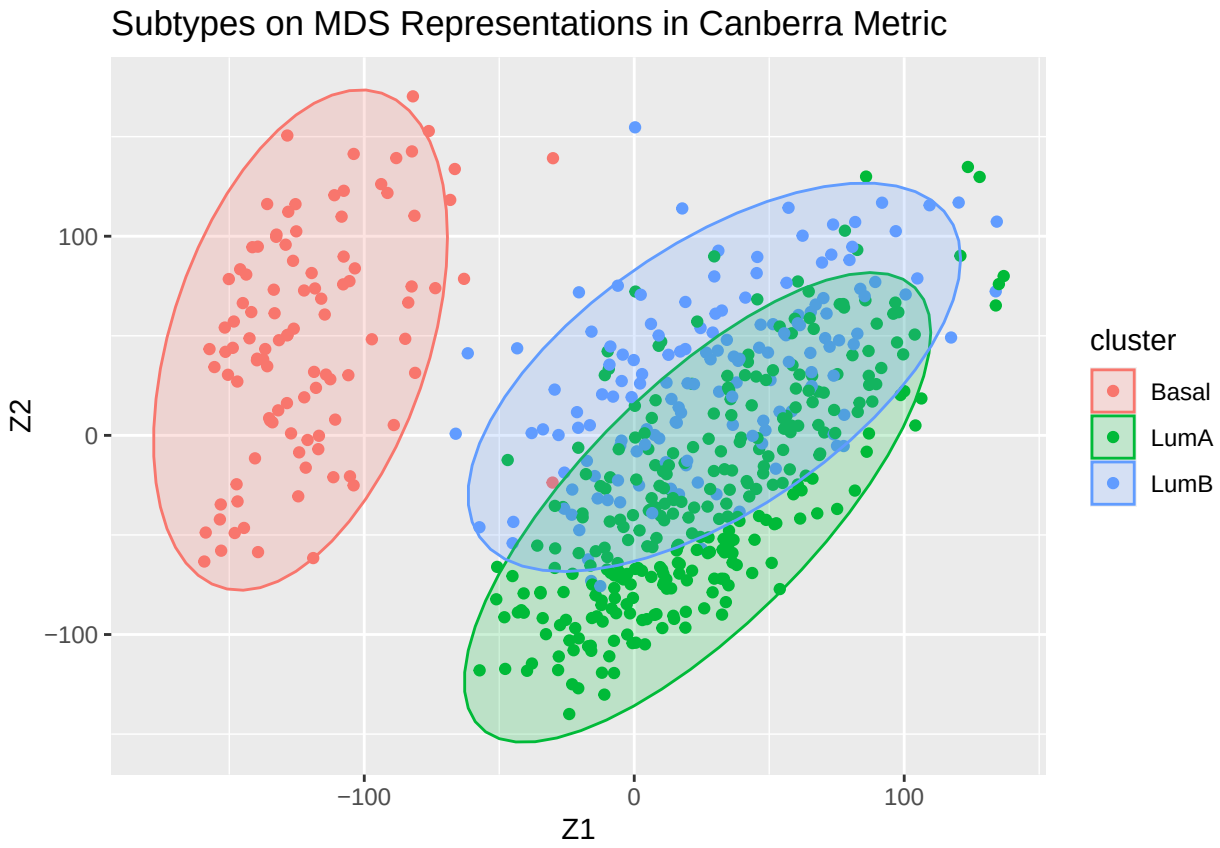
```
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4
```

```
## Warning: package 'tibble' was built under R version 3.6.2

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## Warning: package 'dplyr' was built under R version 3.6.2

## -- Conflicts ---------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

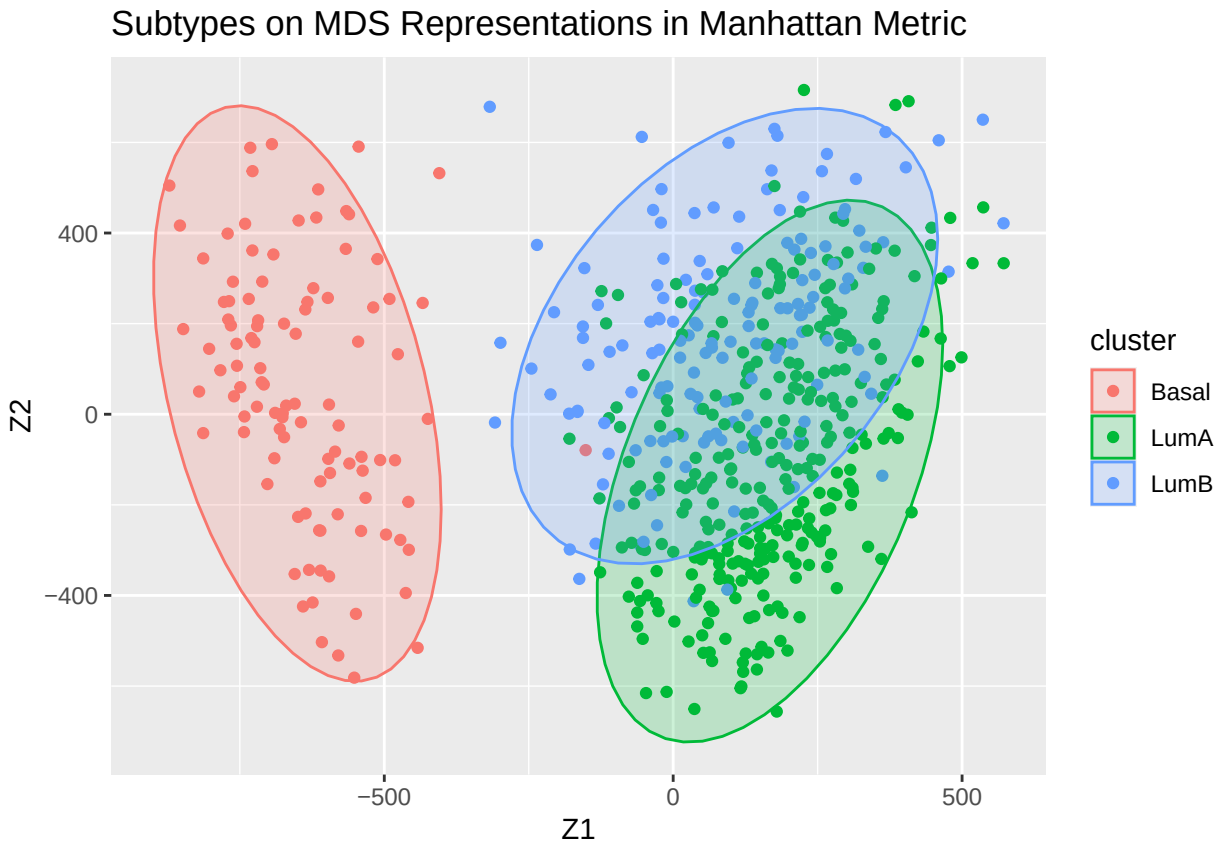
```
gene.mds <-
  data.frame(metric = c("manhattan", "minkowski", "canberra")) %>%
  group_by(metric) %>%
  do(mds = cmdscale(dist(Gene, .$metric, p = 3), eig = T),
     mds.iso = isoMDS(dist(Gene, .$metric, p = 3)))
```

```
## initial  value 35.357092
## final  value 35.356722
## converged
## initial  value 31.891964
## final  value 31.891947
## converged
## initial  value 36.192918
## iter   5 value 27.195473
## iter  10 value 24.366862
## iter  15 value 23.428685
## final  value 22.390615
## converged
```

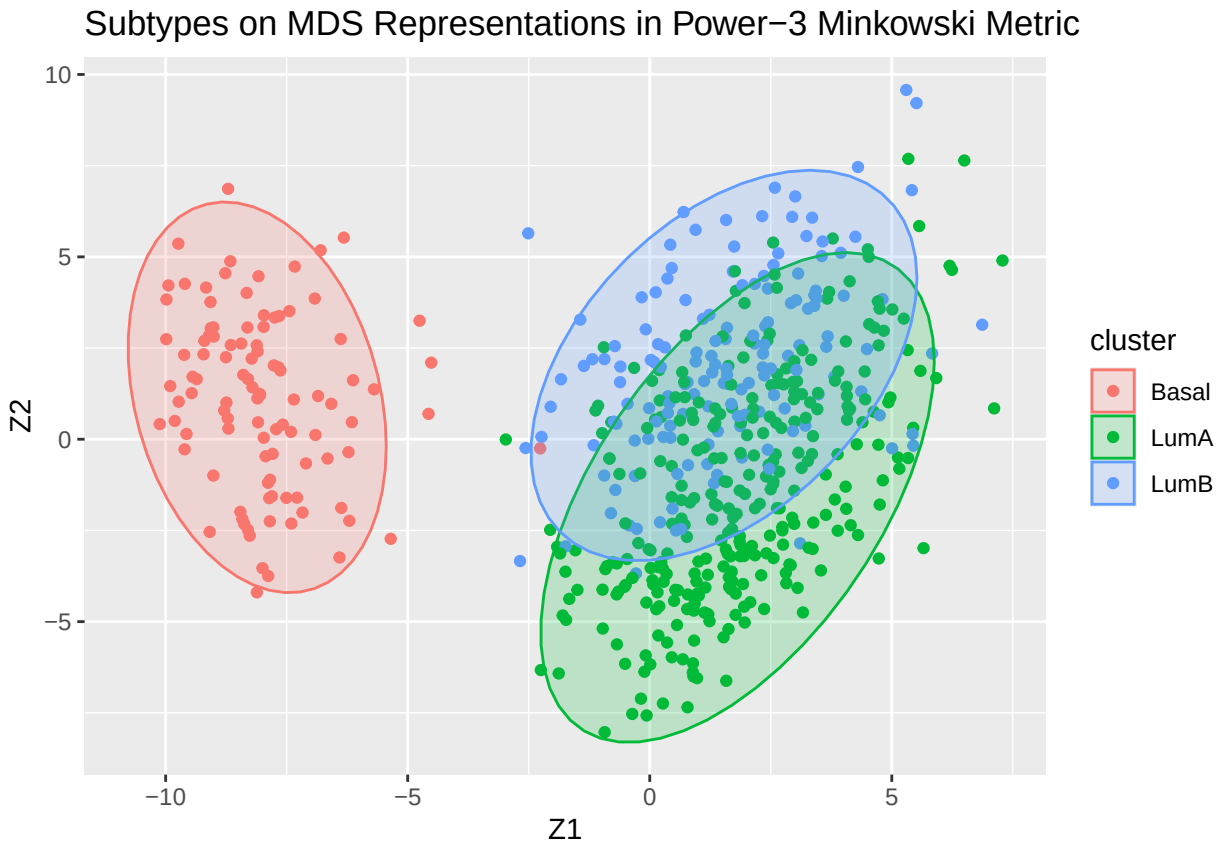
```
colnames(gene.mds$mds[[1]]$points) <- c("Z1", "Z2")
colnames(gene.mds$mds[[2]]$points) <- c("Z1", "Z2")
colnames(gene.mds$mds[[3]]$points) <- c("Z1", "Z2")
colnames(gene.mds$mds.iso[[1]]$points) <- c("Z1", "Z2")
colnames(gene.mds$mds.iso[[2]]$points) <- c("Z1", "Z2")
colnames(gene.mds$mds.iso[[3]]$points) <- c("Z1", "Z2")

ggplot(data = data.frame(gene.mds$mds[[1]]$points, cluster = Subtypes),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on MDS Representations in Canberra Metric")
```



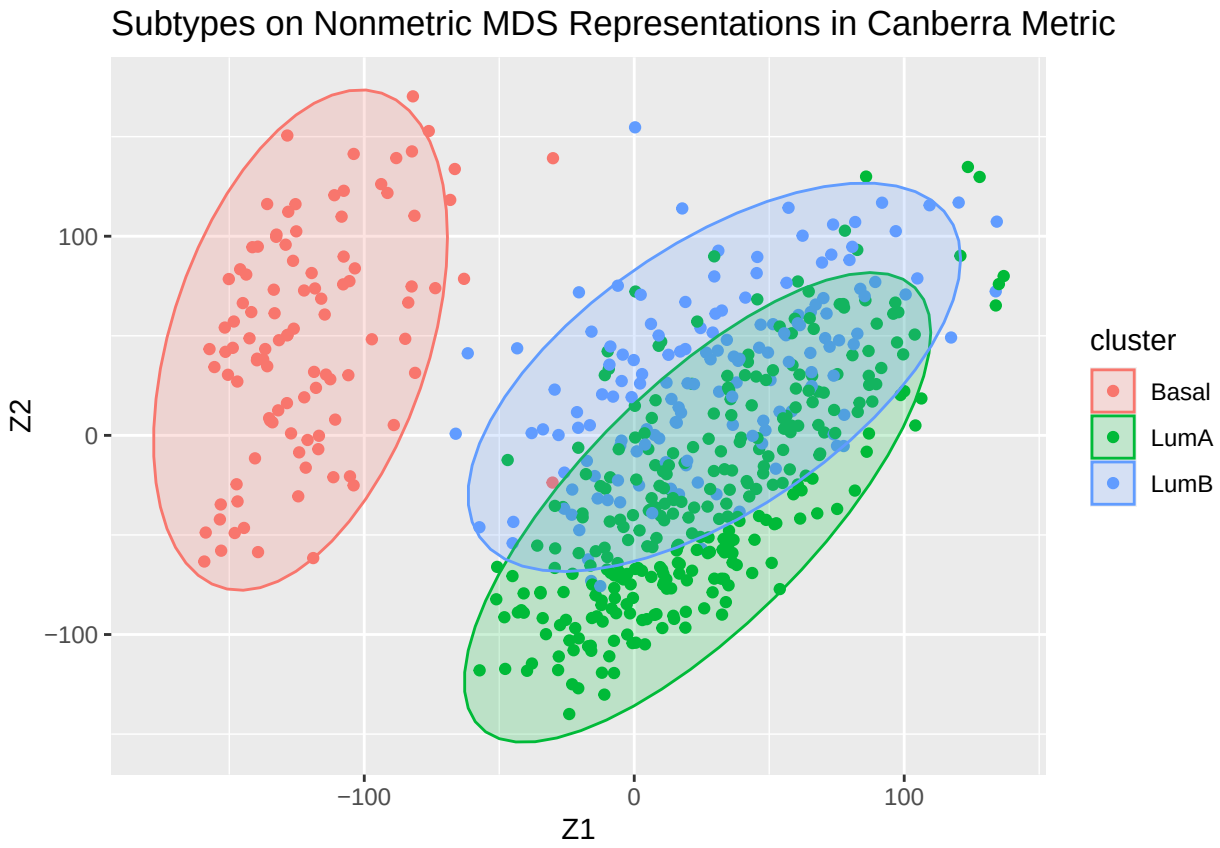

```
ggplot(data = data.frame(gene.mds$mds[[2]]$points, cluster = Subtypes),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on MDS Representations in Manhattan Metric")
```



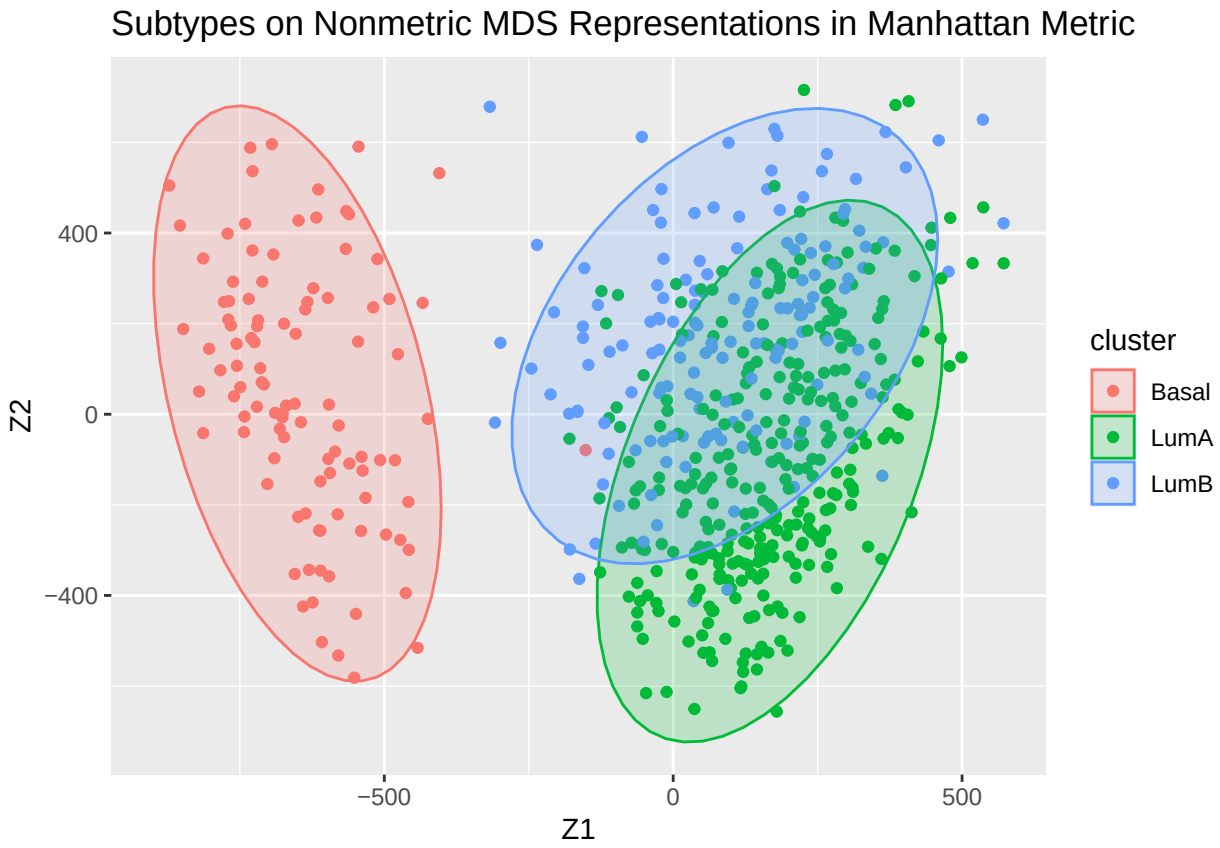

```
ggplot(data = data.frame(gene.mds$mds[[3]]$points, cluster = Subtypes),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on MDS Representations in Power-3 Minkowski Metric")
```



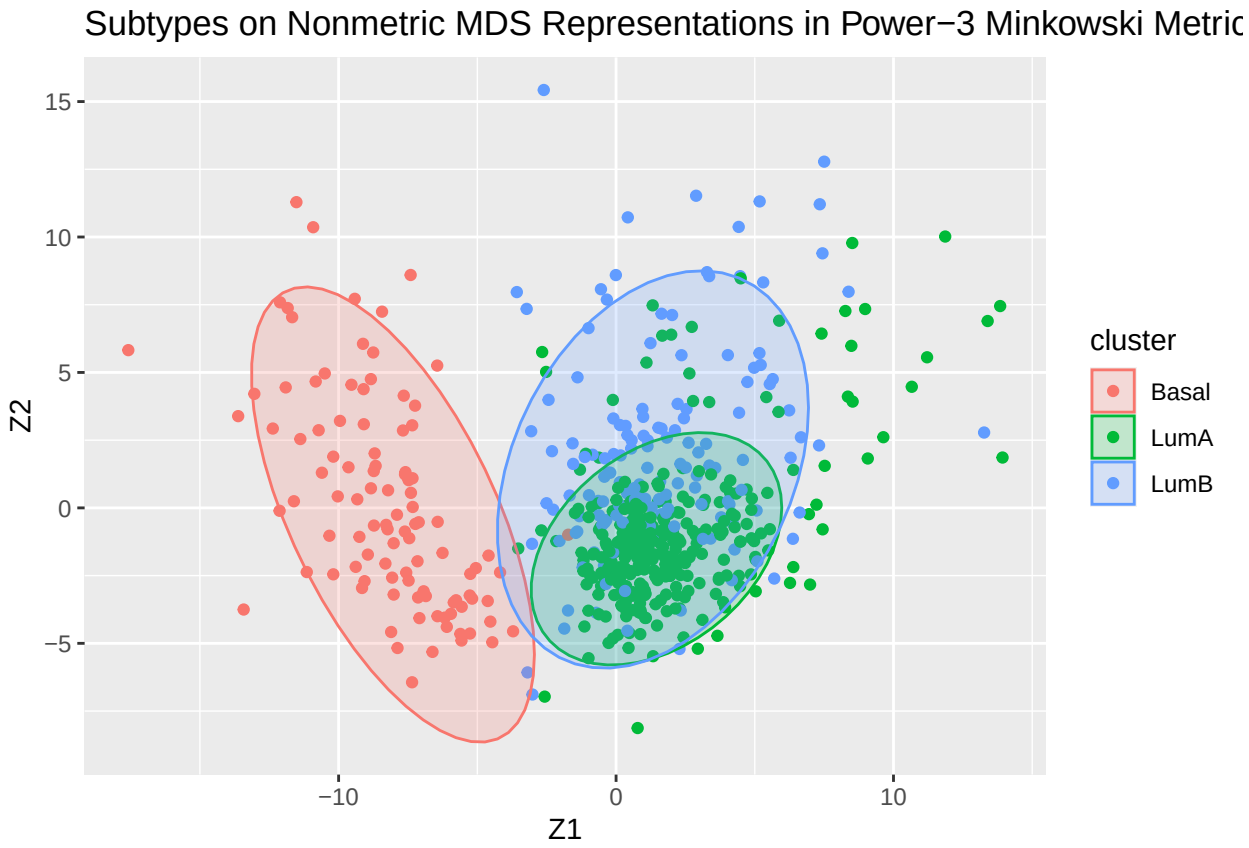

```
ggplot(data = data.frame(gene.mds$mds.iso[[1]]$points, cluster = Subtypes),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on Nonmetric MDS Representations in Canberra Metric")
```




```
ggplot(data = data.frame(gene.mds$mds.iso[[2]]$points, cluster = Subtypes),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on Nonmetric MDS Representations in Manhattan Metric")
```




```
ggplot(data = data.frame(gene.mds$mds.iso[[3]]$points, cluster = Subtypes),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on Nonmetric MDS Representations in Power-3 Minkowski Metric")
```

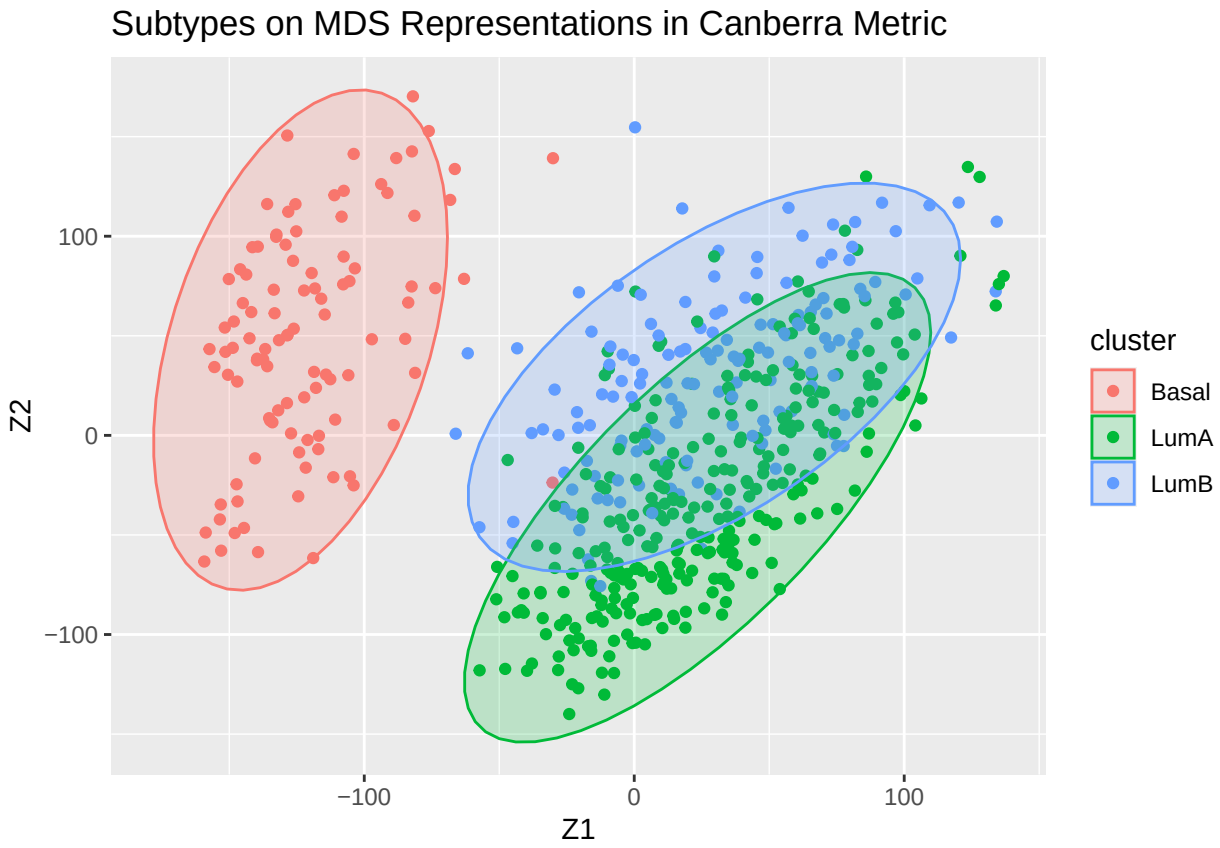


From the plot we see the metic MDS in Manhattan Metric or Canberra Metric is similar to the nonmetric MDS. Except for nonmetric MDS in power-3 Minkowski metric, all the other representations are similar that in 2-dimensional PCA representation.

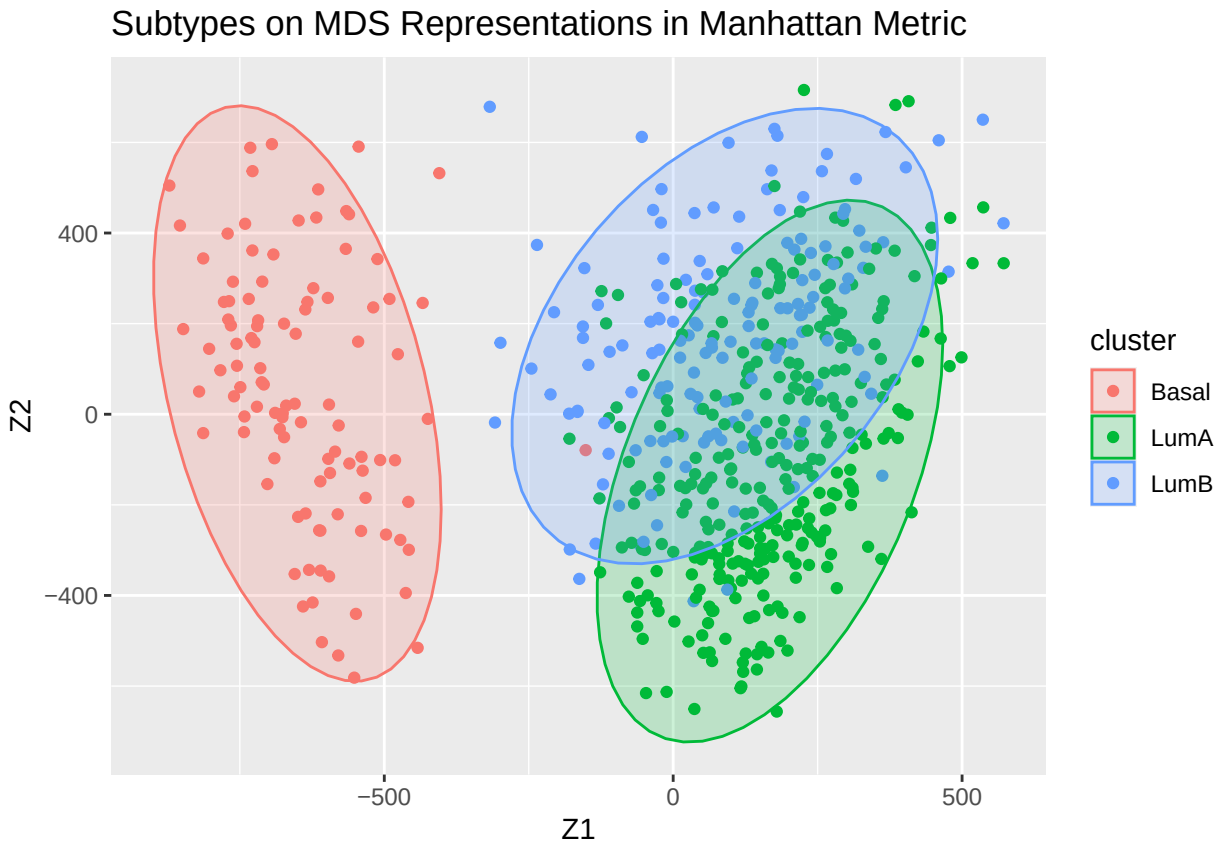
Now we perform K-means with $K = 3$ on the 2-dimensional metic MDSs in Manhattan Metric, Canberra Metric and Power-3 Minkowski respectively.

```
set.seed(1)
gene.mds.can.kmeans <- kmeans(gene.mds$mds[[1]]$points, 3, nstart = 20)
gene.mds.man.kmeans <- kmeans(gene.mds$mds[[2]]$points, 3, nstart = 20)
gene.mds.min.kmeans <- kmeans(gene.mds$mds[[3]]$points, 3, nstart = 20)
```

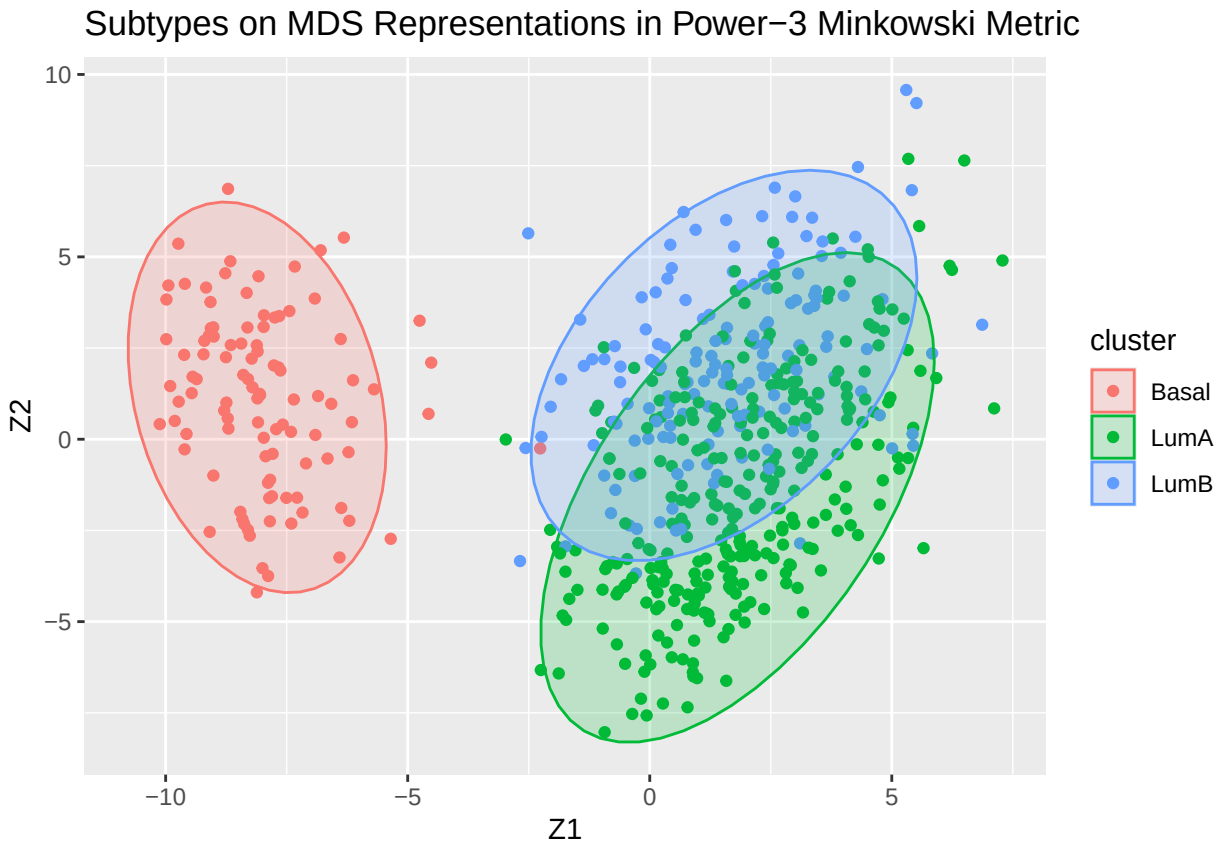
```
ggplot(data = data.frame(gene.mds$mds[[1]]$points, cluster = Subtypes),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on MDS Representations in Canberra Metric")
```



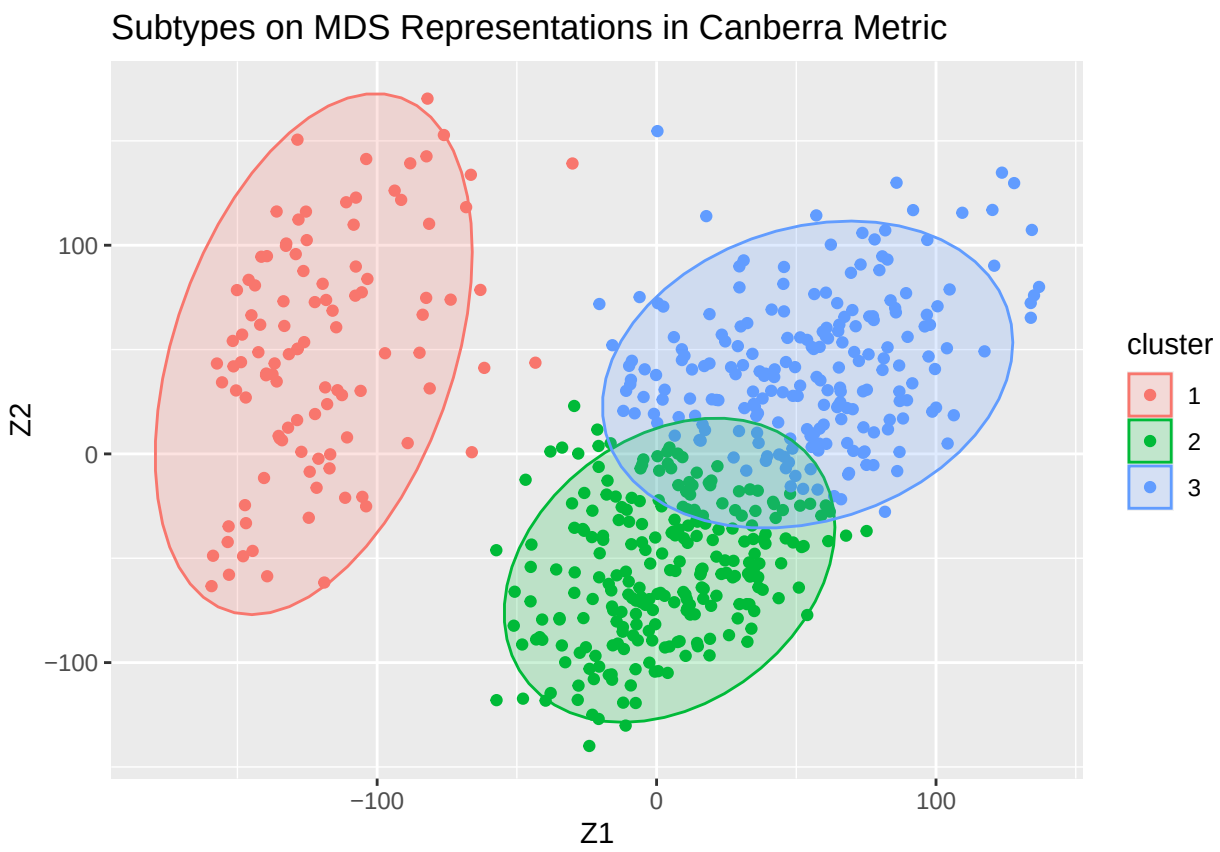

```
ggplot(data = data.frame(gene.mds$mds[[2]]$points, cluster = Subtypes),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on MDS Representations in Manhattan Metric")
```



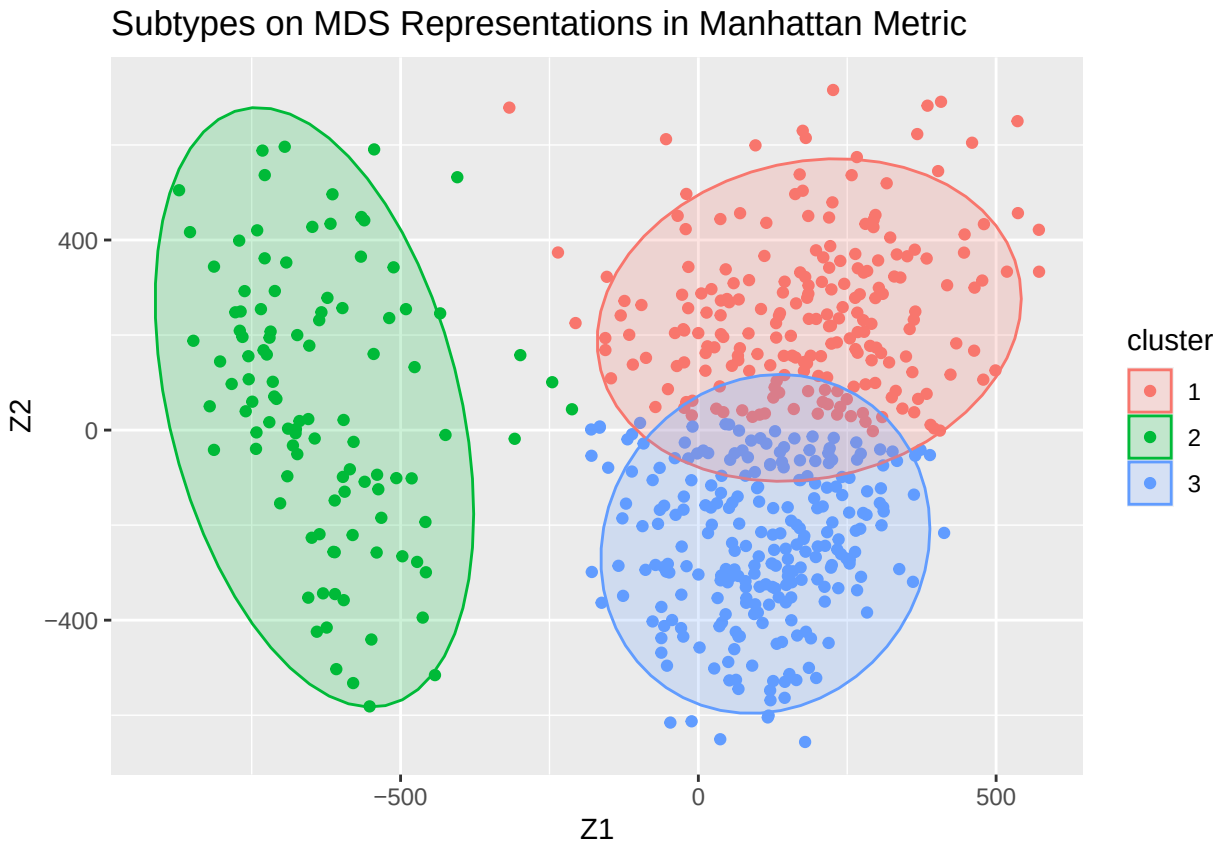

```
ggplot(data = data.frame(gene.mds$mds[[3]]$points, cluster = Subtypes),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on MDS Representations in Power-3 Minkowski Metric")
```



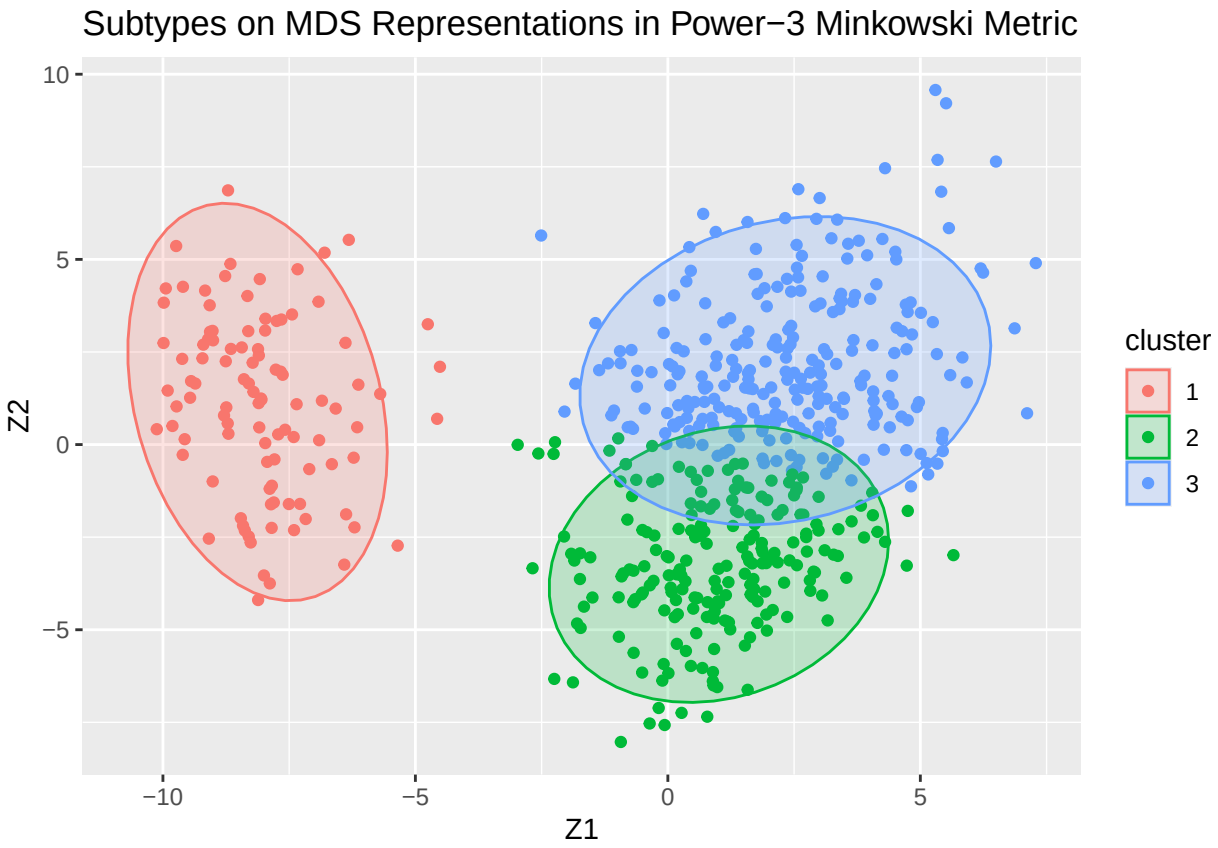

```
ggplot(data = data.frame(gene.mds$mds[[1]]$points,
                         cluster = factor(gene.mds.can.kmeans$cluster)),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on MDS Representations in Canberra Metric")
```




```
ggplot(data = data.frame(gene.mds$mds[[2]]$points,
                         cluster = factor(gene.mds.man.kmeans$cluster)),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on MDS Representations in Manhattan Metric")
```




```
ggplot(data = data.frame(gene.mds$mds[[3]]$points,
                         cluster = factor(gene.mds.min.kmeans$cluster)),
       aes(x = Z1, y = Z2, col = cluster, fill = cluster)) +
  geom_point() +
  stat_ellipse(geom = "polygon", alpha = 0.2) +
  ggtitle("Subtypes on MDS Representations in Power-3 Minkowski Metric")
```



Report the confusion matrix:

```
# Canberra
gene.mds.can.kmeans.confusion <- table(Subtype = Subtypes,
                                       Cluster = gene.mds.can.kmeans$cluster)
print("Confusion Matrix of K-Means on Metric MDS in Canberra Metric", quote = FALSE)
```

```
## [1] Confusion Matrix of K-Means on Metric MDS in Canberra Metric
```

```
addmargins(gene.mds.can.kmeans.confusion)
```

```
##         Cluster
## Subtype   1   2   3 Sum
##   Basal 101   1   0 102
##   LumA    0 205 104 309
##   LumB    3  42 107 152
##   Sum   104 248 211 563
```

```
# Manhattan
gene.mds.man.kmeans.confusion <- table(Subtype = Subtypes,
                                       Cluster = gene.mds.man.kmeans$cluster)
print("Confusion Matrix of K-Means on Metric MDS in Manhattan Metric", quote = FALSE)
```

```
## [1] Confusion Matrix of K-Means on Metric MDS in Manhattan Metric
```

```r
addmargins(gene.mds.man.kmeans.confusion)
```

```
##         Cluster
## Subtype   1   2   3 Sum
##   Basal   0 101   1 102
##   LumA  105   0 204 309
##   LumB  109   4  39 152
##   Sum   214 105 244 563
```

```r
# Power-3 Minkowski
gene.mds.min.kmeans.confusion <- table(Subtype = Subtypes,
                                       Cluster = gene.mds.min.kmeans$cluster)
print("Confusion Matrix of K-Means on Metric MDS in Power-3 Minkowski Metric", quote = FALSE)
```

```
## [1] Confusion Matrix of K-Means on Metric MDS in Power-3 Minkowski Metric
```

```r
addmargins(gene.mds.min.kmeans.confusion)
```

```
##         Cluster
## Subtype   1   2   3 Sum
##   Basal 101   1   0 102
##   LumA    0 183 126 309
##   LumB    0  24 128 152
##   Sum   101 208 254 563
```

Again all clustering results are similar to each other and no-way better than the clustering based on the original gene expression data.

Denoise by subtracting the approximation from the first PC

Suppose we might know that the first PC contains information we aren't interested in. Apply K-means with $K = 3$ to `Gene` dataset **subtracting the approximation from the first PC**.

```r
set.seed(1)
Gene.denoise <- Gene - gene.pca$x[,1] %o% gene.pca$rotation[,1]
gene.denoise.kmeans <- kmeans(Gene.denoise, 3, nstart = 20)
```

Report the confusion matrix:

```r
gene.denoise.kmeans.confusion <- table(Subtype = Subtypes,
                                       Cluster = gene.denoise.kmeans$cluster)

print("Confusion Matrix Subtype-by-Cluster", quote = FALSE)
```

```
## [1] Confusion Matrix Subtype-by-Cluster
```

```r
addmargins(gene.denoise.kmeans.confusion)
```

```
##        Cluster
## Subtype   1   2   3 Sum
##   Basal  26  56  20 102
##   LumA  172  52  85 309
##   LumB   12  60  80 152
##   Sum   210 168 185 563
```

From the confusion matrix we can conclude that the classification has really bad performance. Removing the first Principal Component actually undermines the relationship between the clustering results and the true subtypes.