# BCC pre-processing

Stephen Coleman

10/02/2020

## Data

Read in the data and check the format:

```
data_loc <- "../Raw_data/"
filenames <- c(
  "BRCA.exp.348.med.csv",
  "BRCA.348.precursor.txt",
  "rppaData-403Samp-171Ab-Trimmed.txt",
  "BRCA.Methylation.574probes.802.txt"
)

GE <- read.csv(paste0(data_loc, "BRCA.exp.348.med.csv"), header = TRUE)
miRNA <- read.csv(paste0(data_loc, "BRCA.348.precursor.txt"), header = TRUE)
Protein <- read.table(paste0(data_loc, "rppaData-403Samp-171Ab-Trimmed.txt"), header = TRUE)
Meth <- read.table(paste0(data_loc, "BRCA.Methylation.574probes.802.txt"), header = TRUE)

orig_datafiles <- list(GE, miRNA, Protein, Meth) %>%
  magrittr::set_names(c("GE", "miRNA", "Protein", "Meth"))

dim(GE)
```

```
## [1] 17814   349
```

```
dim(miRNA)
```

```
## [1] 1046  349
```

```
dim(Protein)
```

```
## [1] 171 404
```

```
dim(Meth)
```

```
## [1] 574 802
```

```
# View the head of the data
head(GE[, 1:5])
```

```
##       NAME TCGA.A1.A0SH.01A.11R.A084.07 TCGA.A1.A0SJ.01A.11R.A084.07
## 1    ELMO2                     0.162208                     0.577708
## 2  CREB3L1                     1.338000                    -0.483500
## 3    RPS11                     0.044063                    -0.258062
## 4    PNMA1                     0.249500                     0.682250
## 5     MMP2                     1.843833                    -0.531000
## 6 C10orf90                     0.184500                     0.127500
```

```
##   TCGA.A1.A0SK.01A.12R.A084.07 TCGA.A1.A0SO.01A.22R.A084.07
## 1                     1.113042                    -0.375208
## 2                    -1.558500                    -1.628750
## 3                     0.119813                    -0.542063
## 4                     0.867750                    -0.139750
## 5                    -1.674833                    -1.838167
## 6                    -0.843750                    -0.363750
```

```
head(miRNA[, 1:5])
```

```
##           Gene TCGA.A8.A07E.01A.11R TCGA.A8.A09C.01A.11R TCGA.A8.A084.01A.21R
## 1 hsa-let-7a-1            4629.4498            5437.9616             7340.861
## 2 hsa-let-7a-2            9384.9411           10872.5343            14676.347
## 3 hsa-let-7a-3            4742.0152            5596.5660             7457.106
## 4   hsa-let-7b           31208.3577           25291.9778            71135.465
## 5   hsa-let-7c            2414.6069            1250.5346             1003.195
## 6   hsa-let-7d             537.4601             597.8165              606.083
##   TCGA.A8.A091.01A.11R
## 1            9232.1420
## 2           18276.1584
## 3            9207.9908
## 4           85803.7544
## 5            1438.2719
## 6             419.4695
```

```
# head(Protein[, 1:5])
# head(Meth[, 1:5])
```

For integrative clutstering we need common samples across datasets. Find these.

```
# Match columns (samples) between sources
namesExp <- names(GE)[2:349]
namesmiRNA <- names(miRNA)[2:349]
namesProtein <- names(Protein)[2:404]
namesMeth <- names(Meth)

head(namesExp)
```

```
## [1] "TCGA.A1.A0SH.01A.11R.A084.07" "TCGA.A1.A0SJ.01A.11R.A084.07"
## [3] "TCGA.A1.A0SK.01A.12R.A084.07" "TCGA.A1.A0SO.01A.22R.A084.07"
## [5] "TCGA.A2.A04N.01A.11R.A115.07" "TCGA.A2.A04P.01A.31R.A034.07"
```

```
head(namesProtein)
```

```
## [1] "TCGA.C8.A138.01A.21.A13D.20" "TCGA.AO.A03L.01A.31.A13A.20"
## [3] "TCGA.A2.A0SV.01A.21.A13A.20" "TCGA.A2.A0SW.01A.21.A13A.20"
## [5] "TCGA.BH.A0C0.01A.11.A13B.20" "TCGA.AN.A0AK.01A.11.A13B.20"
```

```
# Matching samples present
namesExp <- substr(namesExp, 1, 16)
namesmiRNA <- substr(namesmiRNA, 1, 16)
namesProtein <- substr(namesProtein, 1, 16)

MatchProt <- match(namesExp, namesProtein, nomatch = 0)
MatchMeth <- match(namesExp, namesMeth, nomatch = 0)
```

Convert to matrix format and set row names

```
miRNA_names <- miRNA[, 1]
miRNA <- miRNA[, 2:349]
miRNA.mat <- as.matrix(miRNA[, order(namesmiRNA)]) %>%
  set_rownames(miRNA_names)

Protein.mat <- Protein[, 2:404]
Protein.mat <- as.matrix(Protein.mat[, MatchProt]) %>%
  set_rownames(Protein[, 1])

Meth.mat <- as.matrix(Meth[, MatchMeth]) %>%
  set_rownames(row.names(Meth))

Exp.mat      <- as.matrix(GE[,2:349])

reduced_matrix_data <- list(Exp.mat, miRNA.mat, Protein.mat, Meth.mat) %>%
  magrittr::set_names(c("GE", "miRNA", "Protein", "Meth"))
```

How much missingness is present?

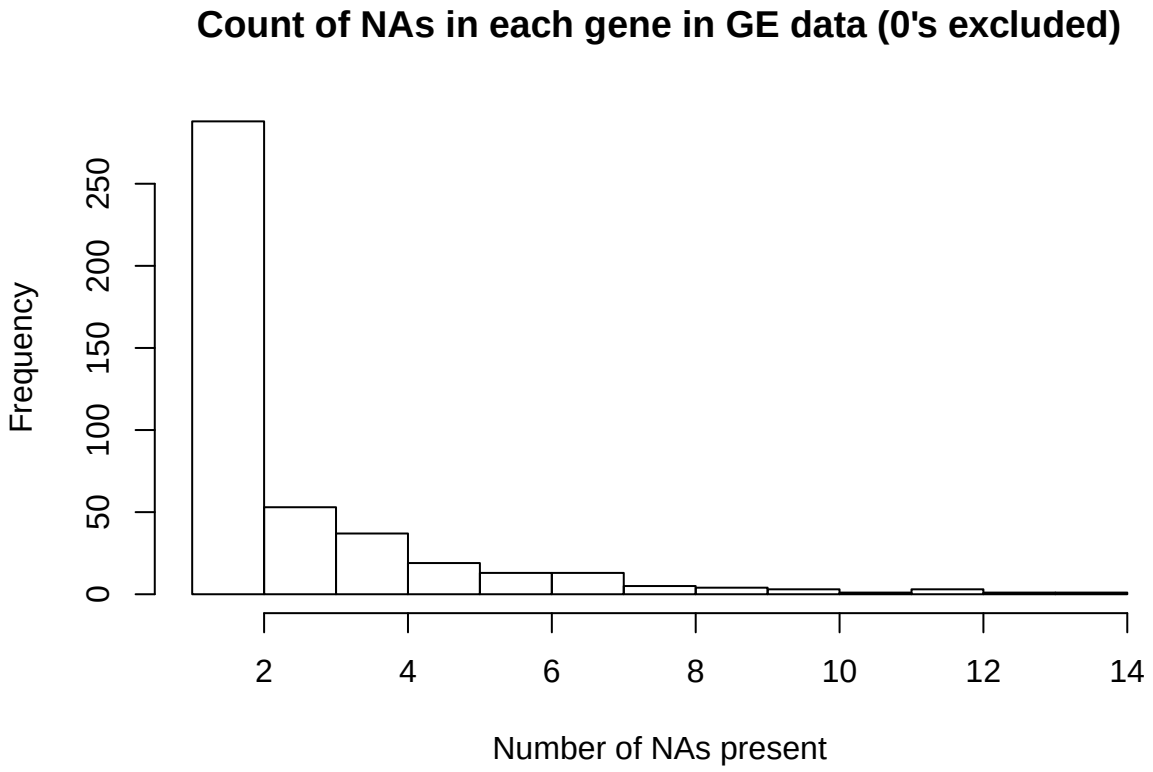
```
# How many missing entries in our data
print(lapply(reduced_matrix_data, function(x){
  sum(is.na(x))
}
))
```

```
## $GE
## [1] 1119
##
## $miRNA
## [1] 0
##
## $Protein
## [1] 0
##
## $Meth
## [1] 0
```

```
# The only dataset with NAs is the Gene expression data
dim(reduced_matrix_data$GE)
```

```
## [1] 17814   348
```

```
ge_missingness <- rowSums(is.na(reduced_matrix_data$GE))
ge_missingness[ge_missingness > 0] %>%
  hist(main = "Count of NAs in each gene in GE data (0's excluded)",
       xlab = "Number of NAs present")
```




As the Gene Expression contains actual NA's we will impute missing values using `knn` with $k = 10$ (the default setting).

```
# Impute missing values via KNN (K=10) for the Gene expression data
Exp.mat      <- impute.knn(Exp.mat)
Exp.mat      <- Exp.mat$data %>%
  set_rownames(GE[,1])

reduced_matrix_data$GE <- Exp.mat
```

We might also be interested in the number of 0's present in the datasets as this could represent missing data.

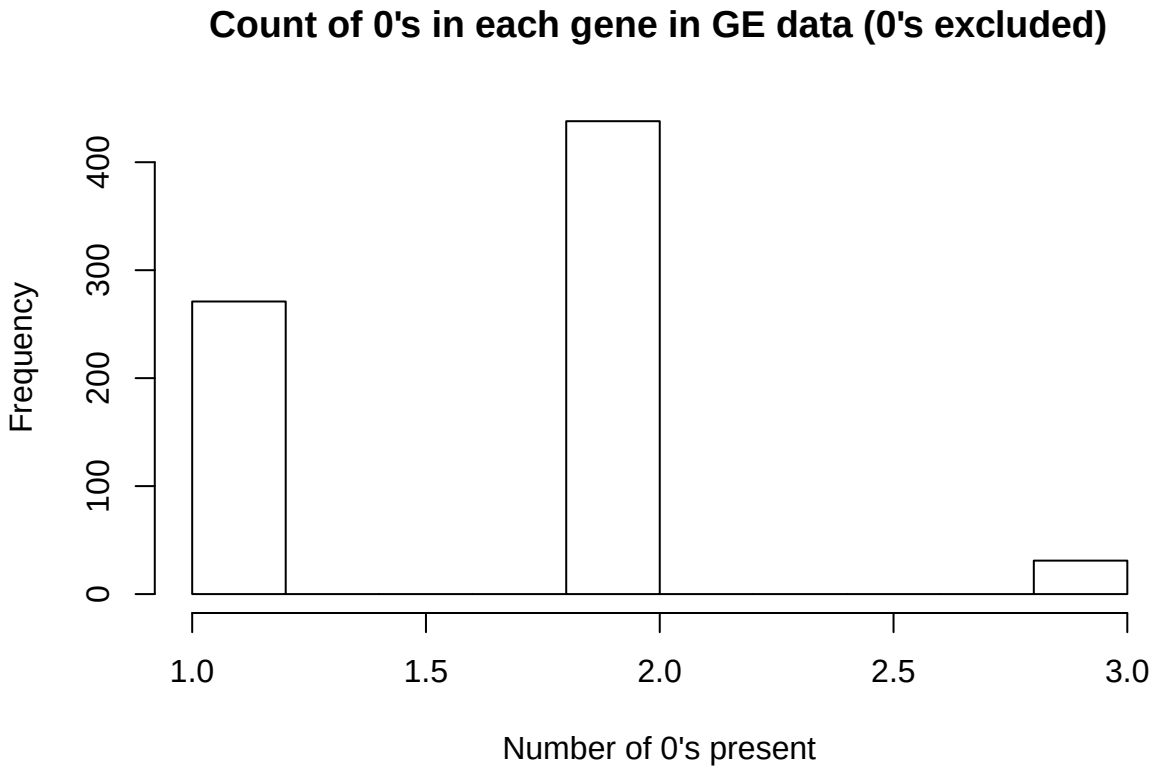
```
# Check how many 0's there are in the datasets (possibly these are missing points too!)
print(lapply(reduced_matrix_data, function(x){
  sum(x == 0)
}
))
```

```
## $GE
## [1] 1240
##
## $miRNA
## [1] 211498
##
## $Protein
## [1] 0
##
## $Meth
## [1] 0
```

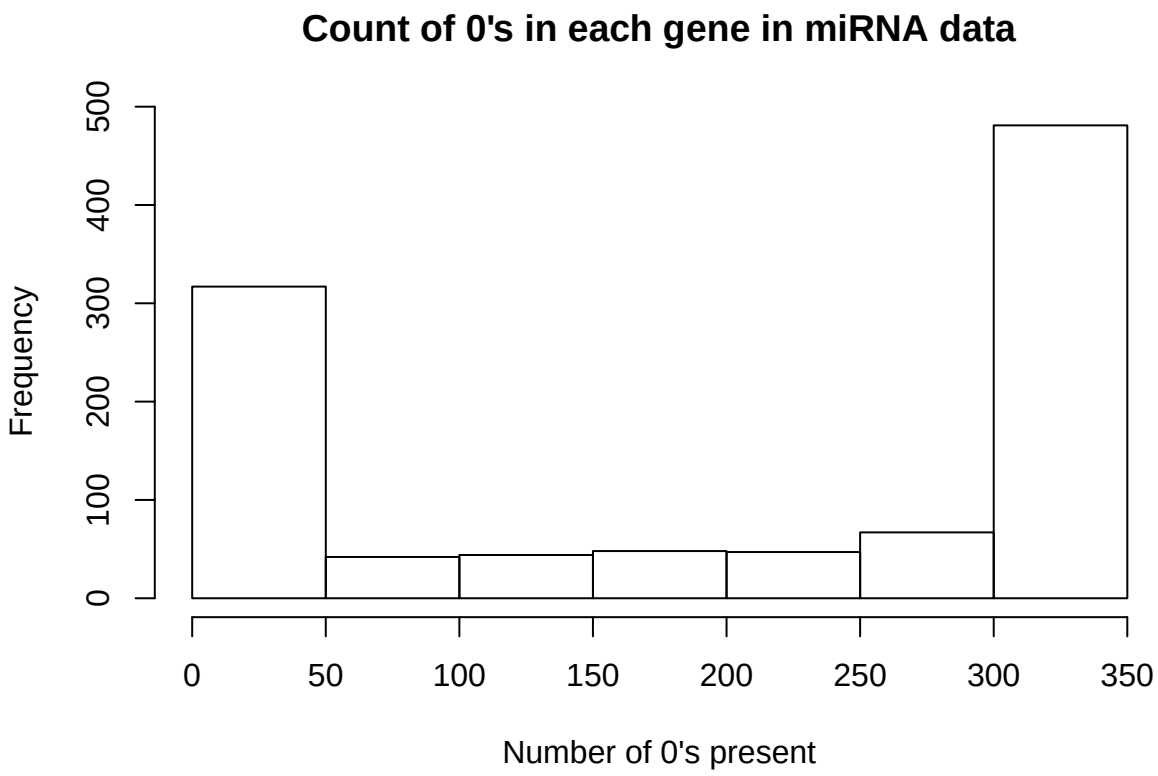
```
# The GE and miRNA datasets both have 0 entries, check how many genes have 0's
print(lapply(reduced_matrix_data, function(x){
  sum(rowSums(x == 0) > 0)
}
))
```

```
## $GE
## [1] 740
##
## $miRNA
## [1] 840
##
## $Protein
## [1] 0
##
## $Meth
## [1] 0
```

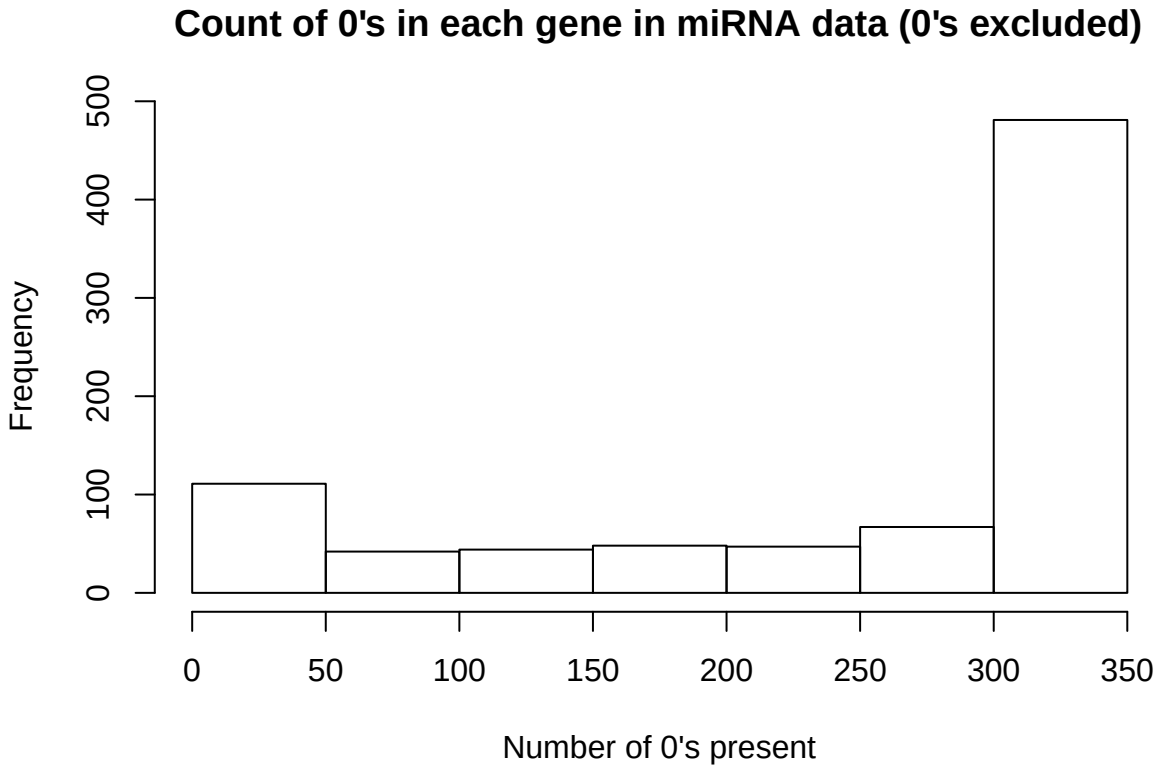
```
ge_zeroness <- rowSums(reduced_matrix_data$GE == 0)
ge_zeroness[ge_zeroness > 0] %>%
  hist(main = "Count of 0's in each gene in GE data (0's excluded)",
       xlab = "Number of 0's present")
```




```
miRNA_zeroness <- rowSums(reduced_matrix_data$miRNA == 0)
miRNA_zeroness %>%
  hist(main = "Count of 0's in each gene in miRNA data",
       xlab = "Number of 0's present")
```




```
miRNA_zeroness[miRNA_zeroness > 0] %>%
  hist(main = "Count of 0's in each gene in miRNA data (0's excluded)",
       xlab = "Number of 0's present")
```



```
# There's a far greater number of 0 entries in the miRNA compared to the GE;
# also, no gene in the GE dataset has more than 3 associated 0 entries
```

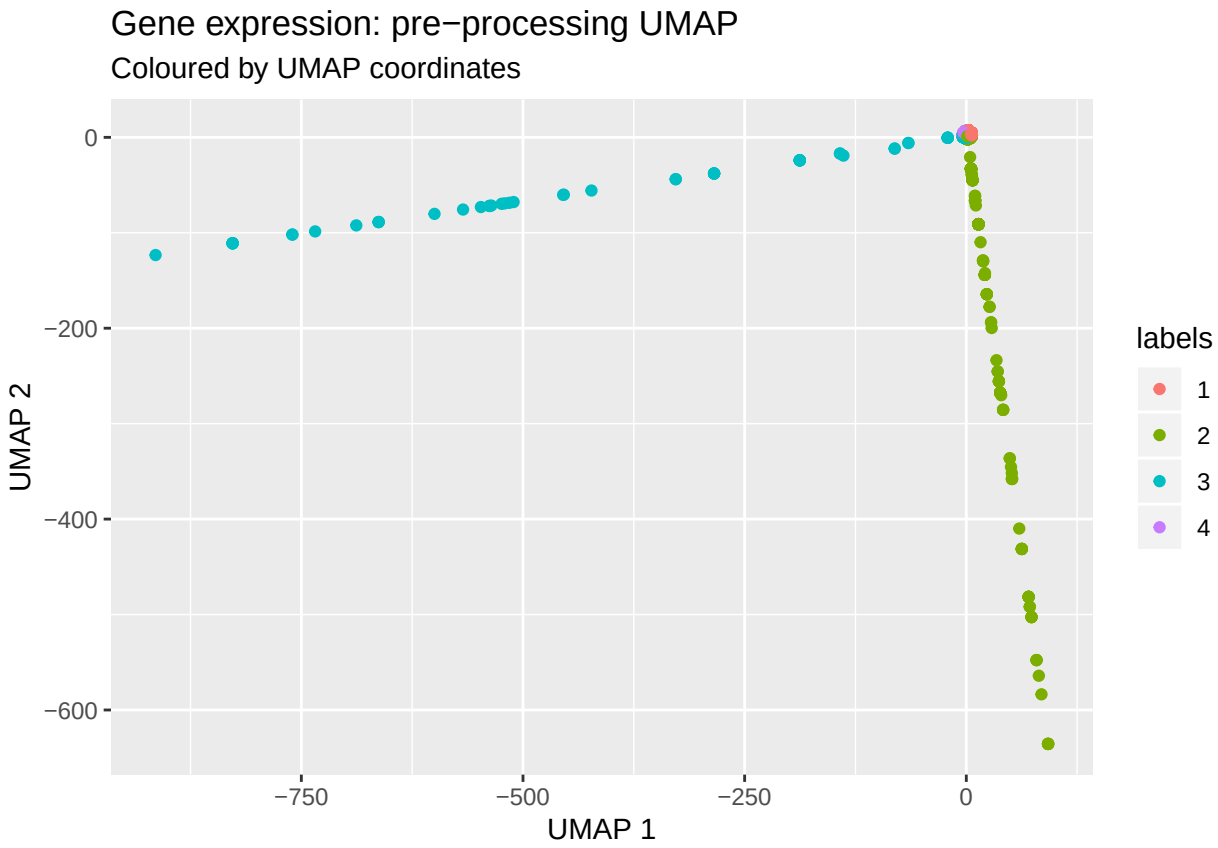
Now we visualise the effect of the preprocessing steps by mapping the points to a 2D UMAP; colouring based upon their UMAP coordinates. We split the points based upon the median of the points in each direction, so points are assigned values based upon:

Table 1: Labels defined by UMAP coordinates.

| Condition.in.UMAP.1 | Condition.in.UMAP.2 | Label |
|---|---|---|
| Greater than median | Greater than median | 1 |
| Greater than median | Less than median | 2 |
| Less than median | Greater than median | 3 |
| Less than median | Less than median | 4 |

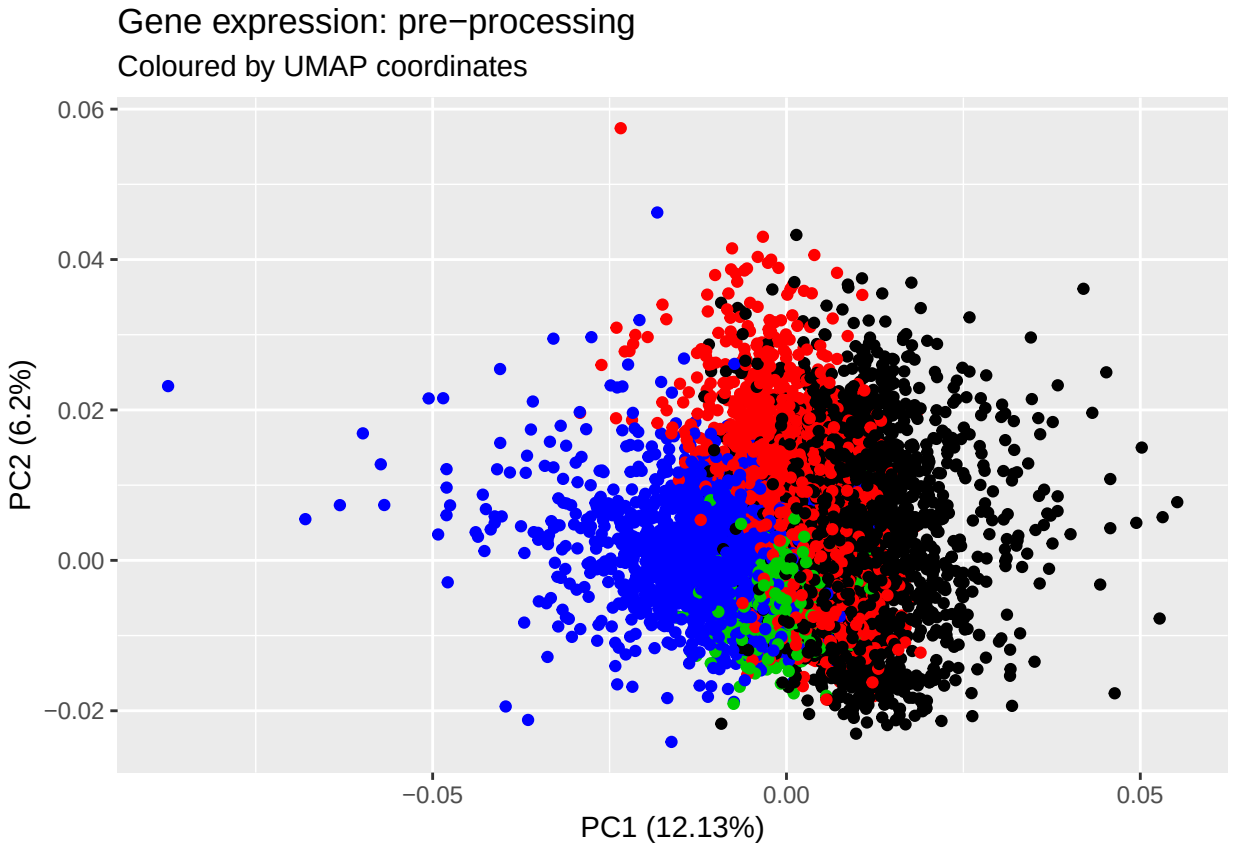
First consider the

```
# Do PCA
ge_pca <- prcomp(Exp.mat)

# Visaulise the first two components
# autoplot(ge_pca)

# Apply UMAP to the data
ge_umap <- umap(Exp.mat)

# Find labels from the UMAP to track how transform changes layout of data
ge_labels <- makeUMAPLabels(ge_umap$layout)
ge_plt_data <- makeUMAPPlotData(ge_umap$layout, ge_labels)
plotUMAP(ge_plt_data)  +
  labs(title = "Gene expression: pre-processing UMAP",
       subtitle = "Coloured by UMAP coordinates",
       x = "UMAP 1",
       y = "UMAP 2")
```

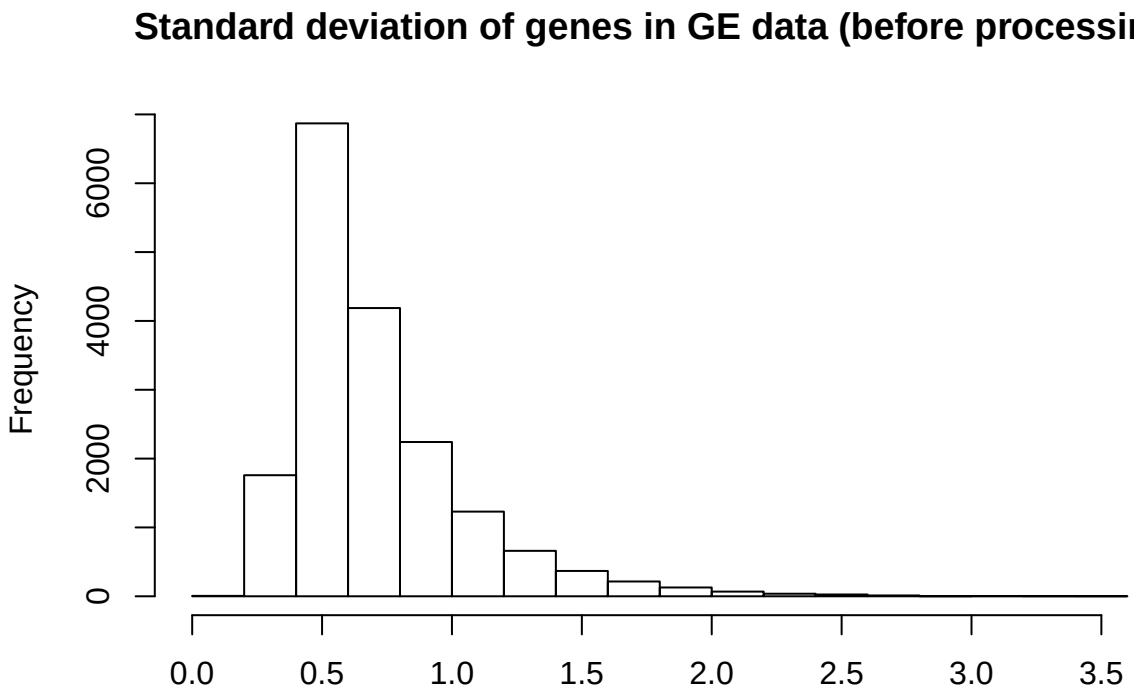



```
# Save UMAP plot
# ggsave(umap_file_names[1])

# Plot PCA with UMAP defined labels
autoplot(ge_pca, data = Exp.mat, colour = ge_labels) +
  labs(title = "Gene expression: pre-processing",
       subtitle = "Coloured by UMAP coordinates")
```

```
## Warning in if (value %in% columns) {: the condition has length > 1 and only the
## first element will be used
```




```
# ggsave(file_names[[1]][1])

Exp.mat %>%
  apply(1, sd) %>%
  hist(main = "Standard deviation of genes in GE data (before processing)")
```

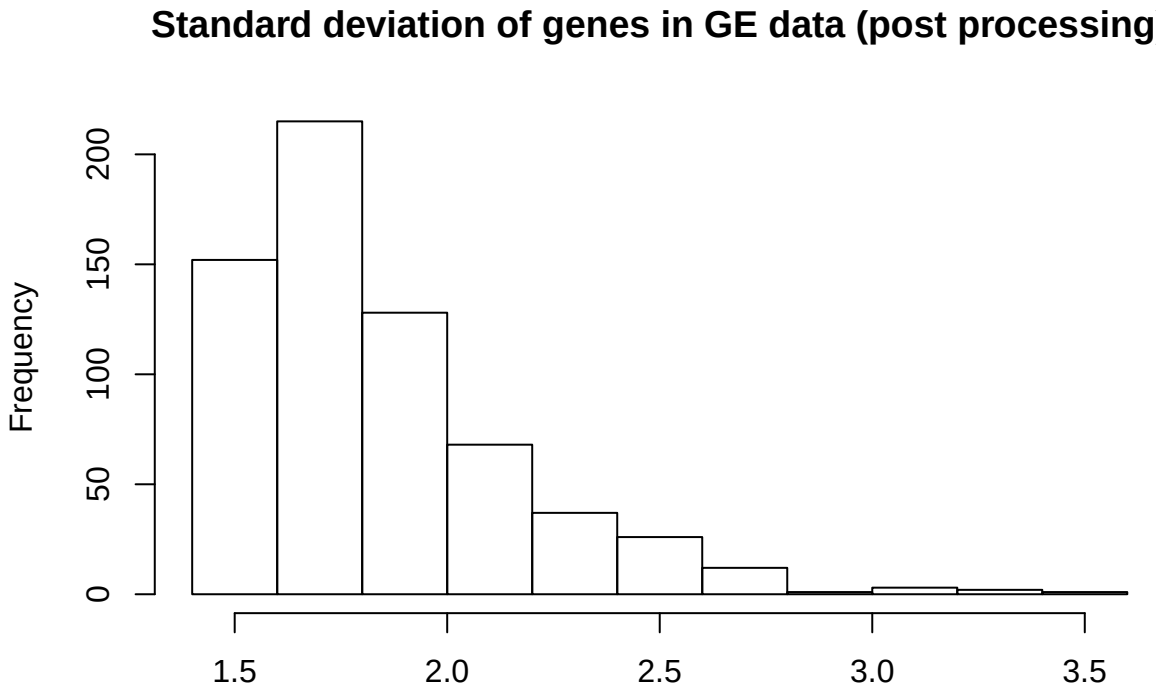



.

```
processedExpression  <- Exp.mat[apply(Exp.mat,1,sd)>1.5,] ###Filter to select only most variable genes
print(ncol(Exp.mat) - ncol(processedExpression))
```

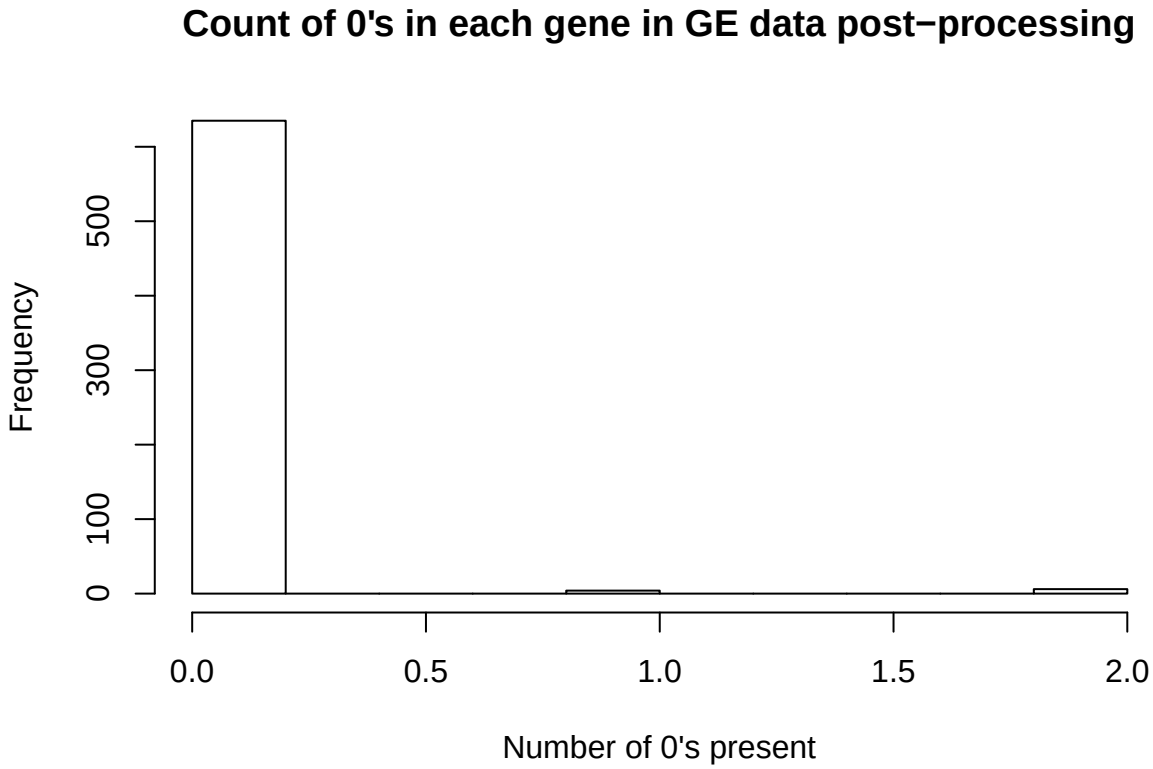
```
## [1] 0
```

```
processedExpression %>%
  apply(1, sd) %>%
  hist(main = "Standard deviation of genes in GE data (post processing)")
```

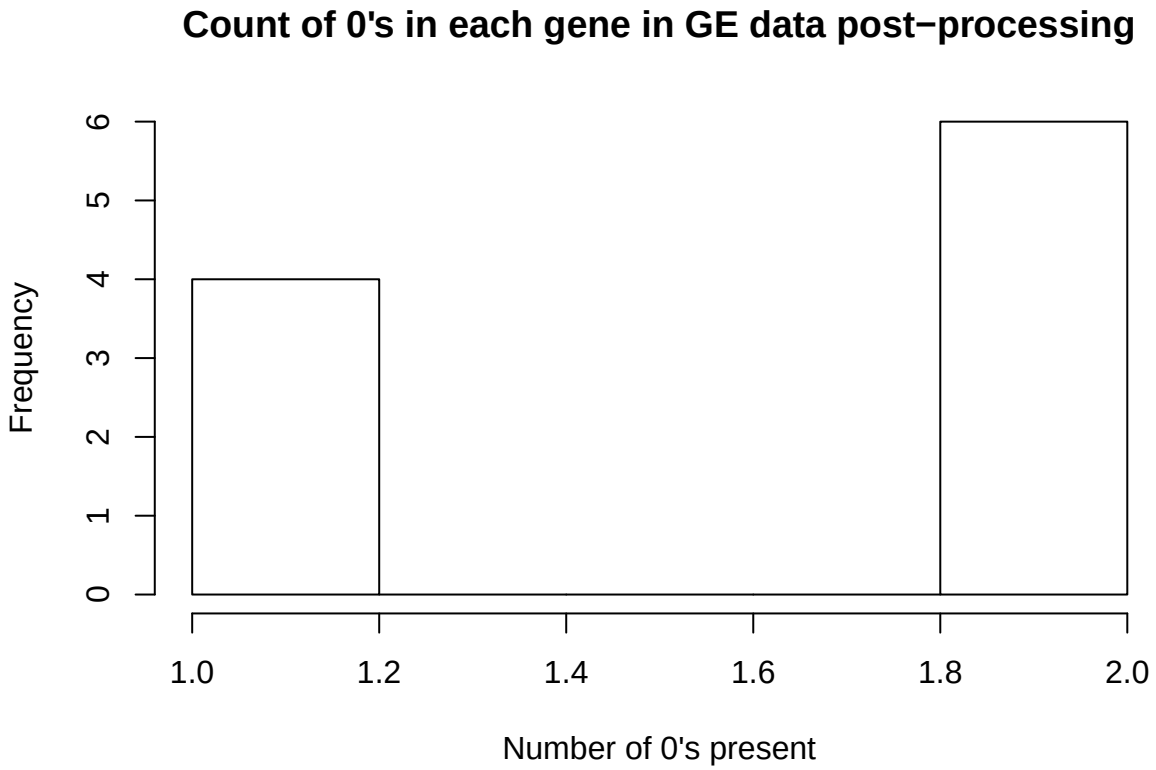



.

```
ge_zeroness_post <- rowSums(processedExpression == 0)

ge_zeroness_post %>%
  hist(main = "Count of 0's in each gene in GE data post-processing",
       xlab = "Number of 0's present")
```

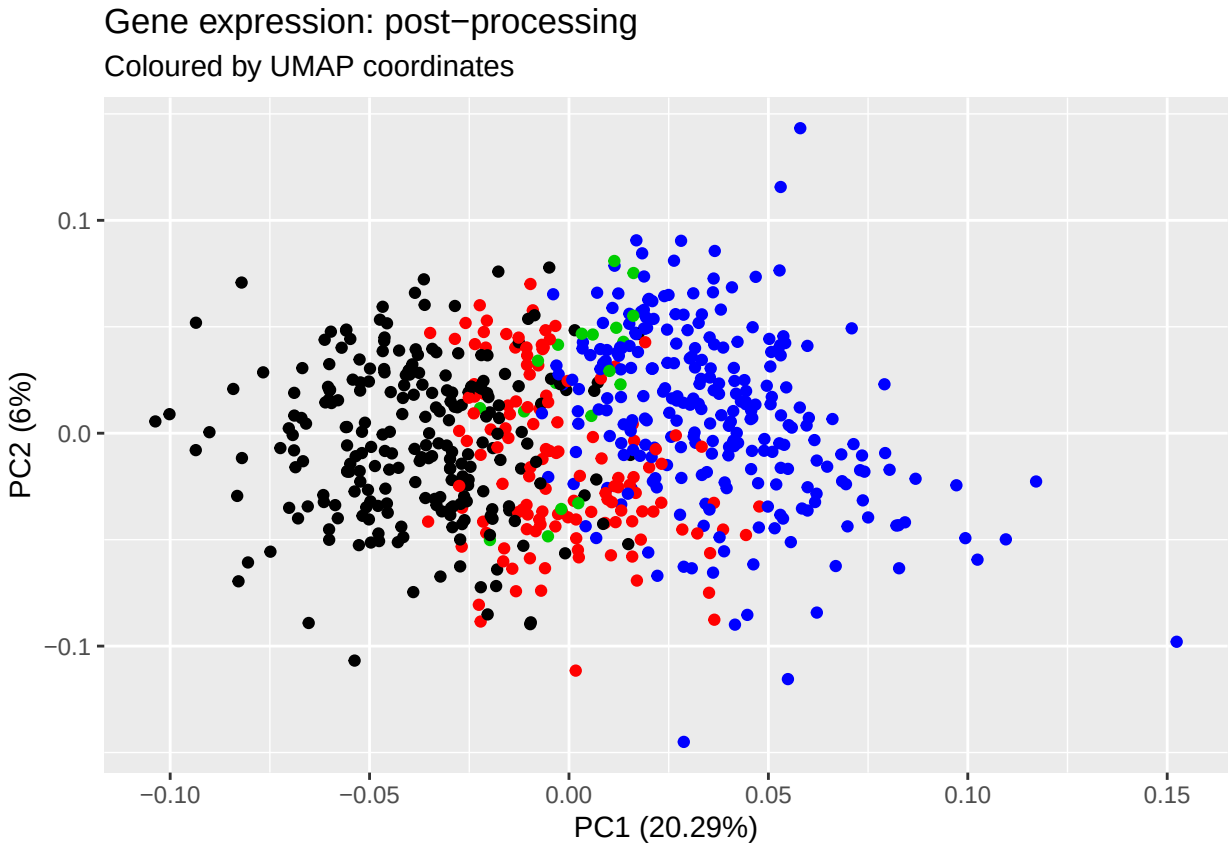


```r
ge_zeroness_post[ge_zeroness_post > 0] %>%
  hist(main = "Count of 0's in each gene in GE data post-processing",
       xlab = "Number of 0's present")
```

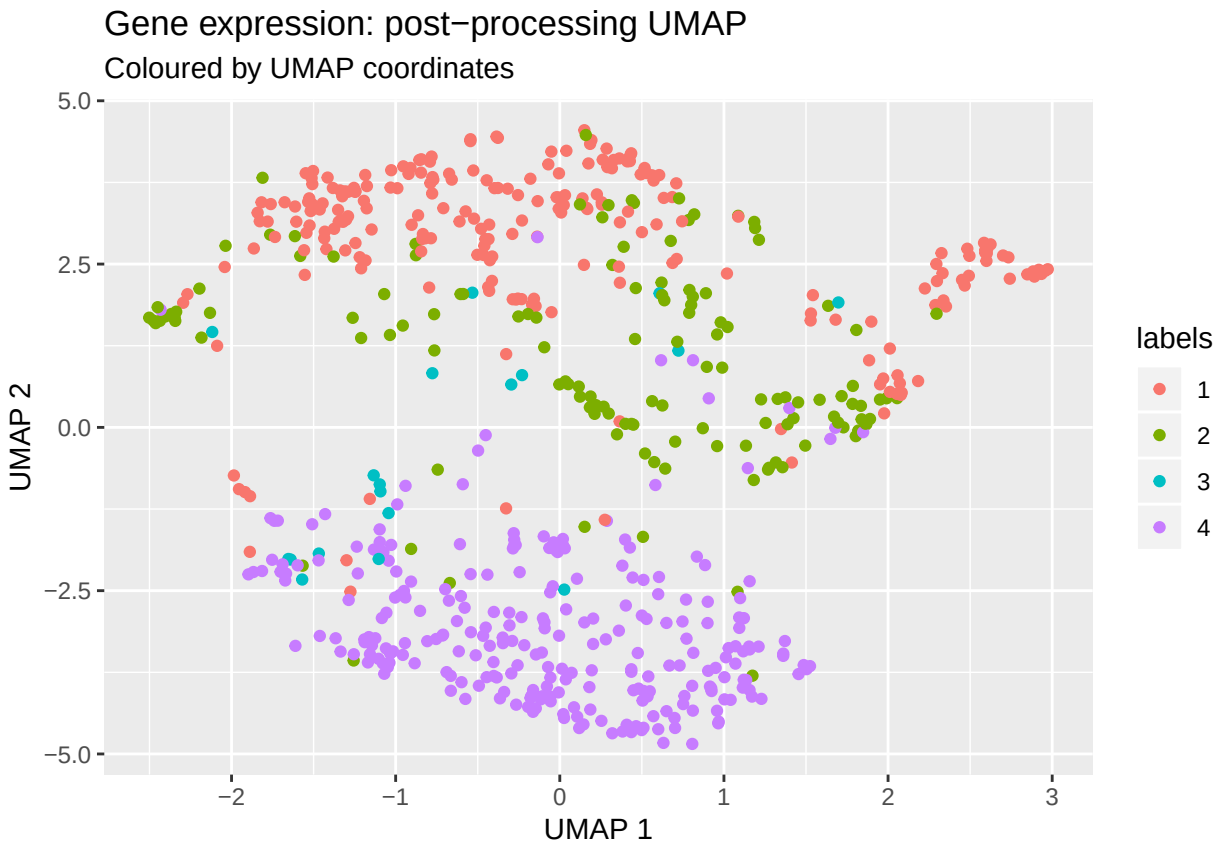



```r
# Look at PCs after transform with the same labelling
p_ge_pca <- prcomp(processedExpression)
# autoplot(p_ge_pca)
autoplot(p_ge_pca, data = processedExpression, colour = ge_labels[apply(Exp.mat,1,sd)>1.5]) +
  labs(title = "Gene expression: post-processing",
       subtitle = "Coloured by UMAP coordinates")
```

```
## Warning in if (value %in% columns) {: the condition has length > 1 and only the
## first element will be used
```

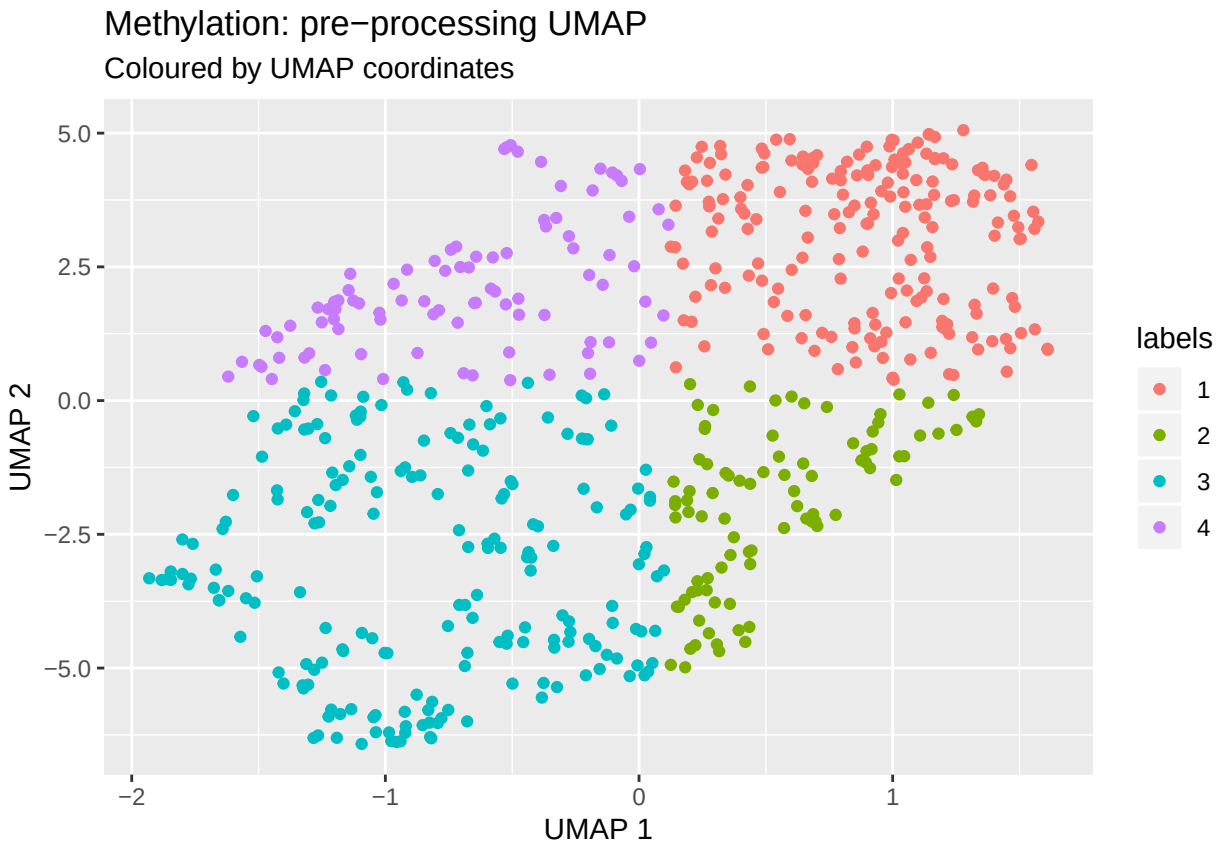



```
# ggsave(file_names[[2]][1])

p_ge_umap <- umap(processedExpression)
p_ge_plt_data <- makeUMAPPlotData(p_ge_umap$layout, ge_labels[apply(Exp.mat,1,sd)>1.5])
plotUMAP(p_ge_plt_data)  +
  labs(title = "Gene expression: post-processing UMAP",
       subtitle = "Coloured by UMAP coordinates",
       x = "UMAP 1",
       y = "UMAP 2")
```



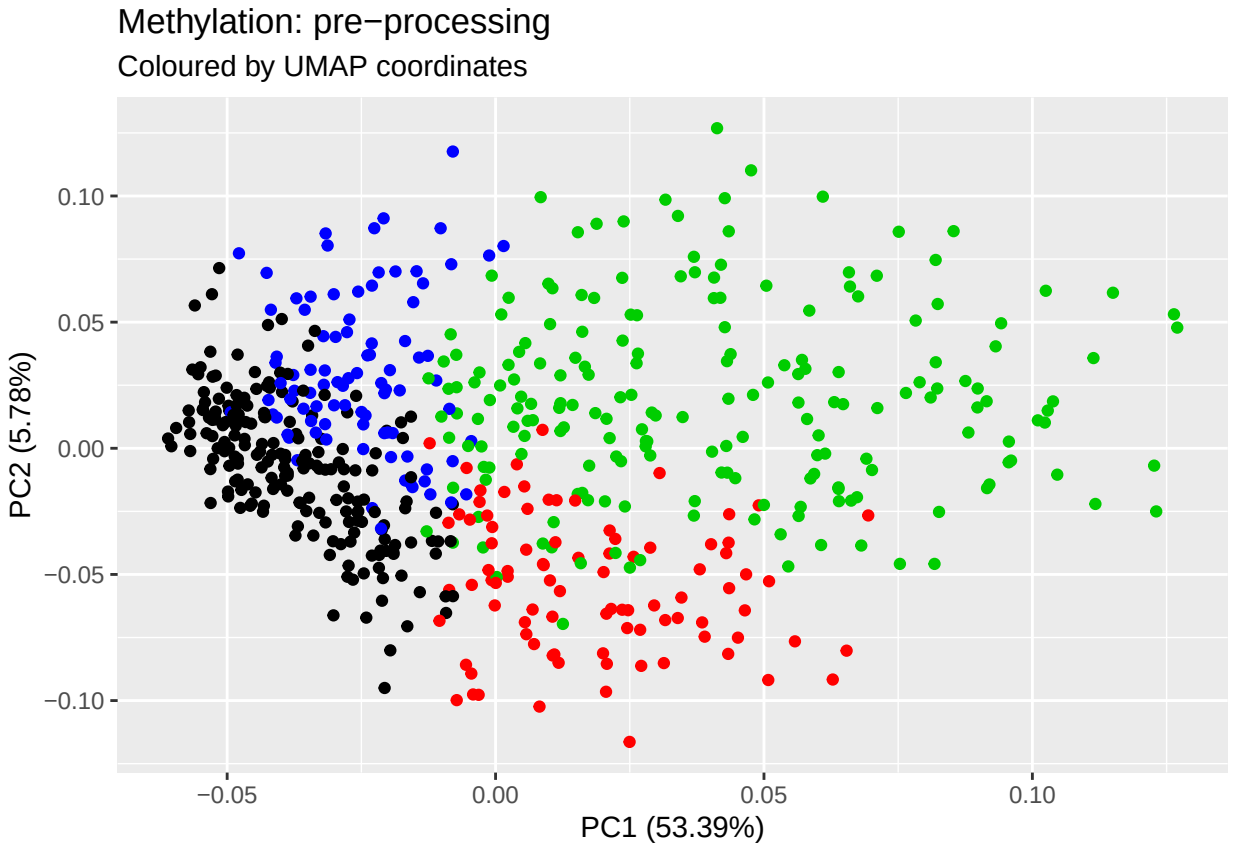

```
# ggsave("~/Documents/PhD/Year_1/Consensus_clustering/Analysis/BCC_TCGA_data/Data/gene_expression_umap_
```

```
meth_pca <- prcomp(Meth.mat)

meth_umap <- umap(Meth.mat)


meth_labels <- makeUMAPLabels(meth_umap$layout)
meth_plt_data <- makeUMAPPlotData(meth_umap$layout, meth_labels)
plotUMAP(meth_plt_data)  +
  labs(title = "Methylation: pre-processing UMAP",
       subtitle = "Coloured by UMAP coordinates",
       x = "UMAP 1",
       y = "UMAP 2")
```

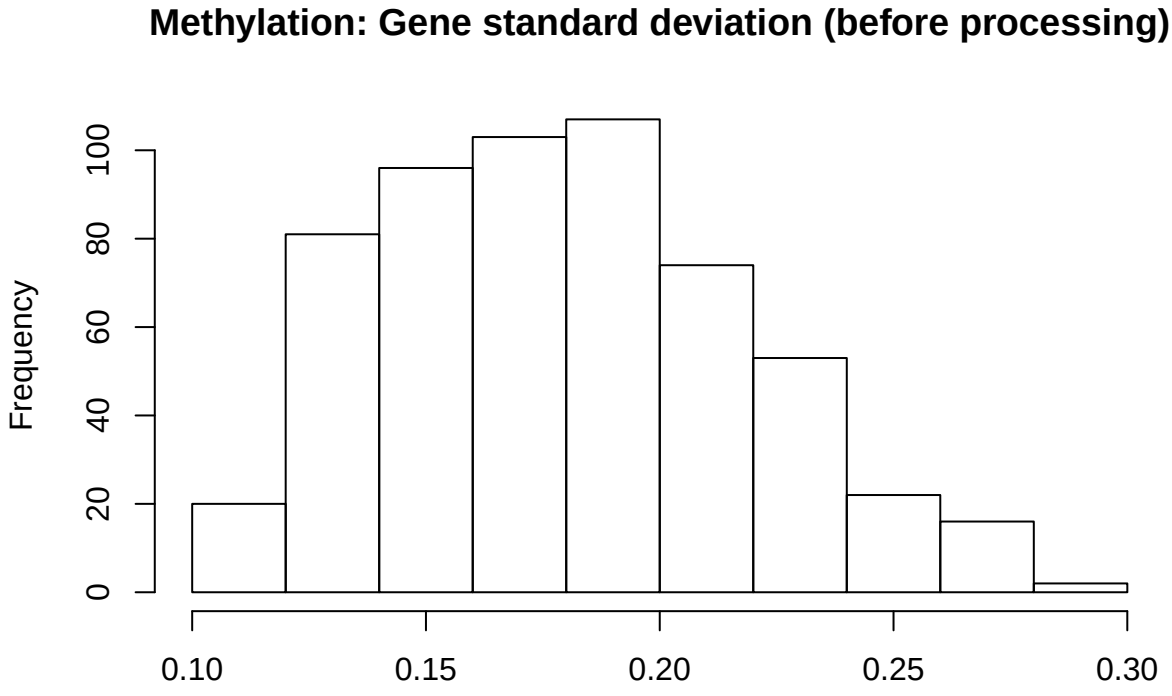



```
# ggsave(umap_file_names[2])


autoplot(meth_pca, data = Meth.mat, colour = meth_labels) +
  labs(title = "Methylation: pre-processing",
       subtitle = "Coloured by UMAP coordinates")
```

```
## Warning in if (value %in% columns) {: the condition has length > 1 and only the
## first element will be used
```

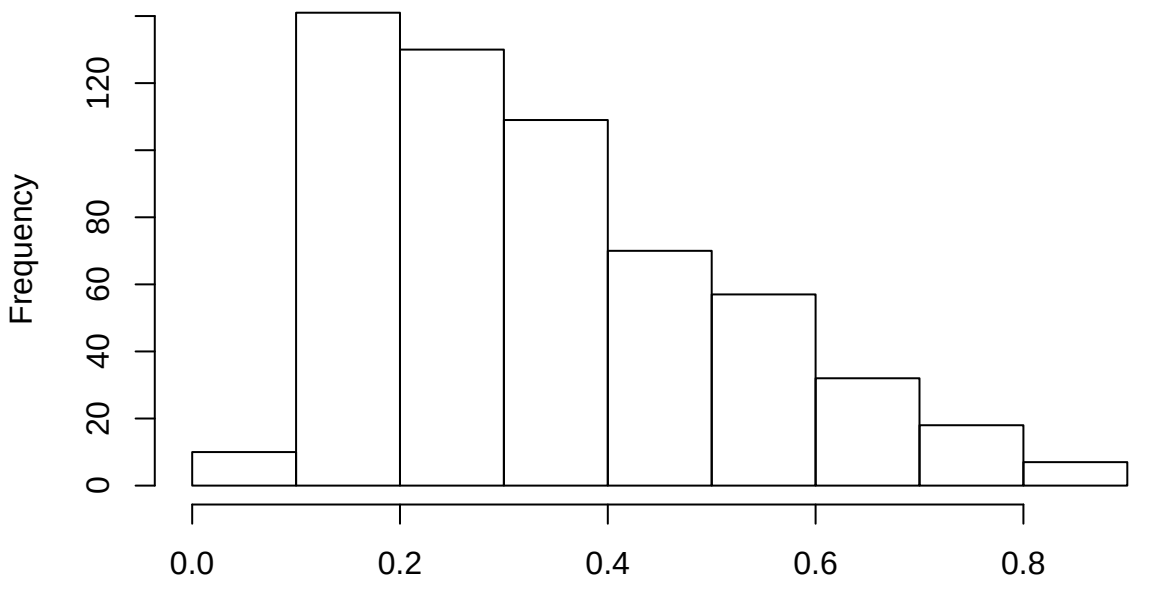



```
# ggsave(file_names[[1]][2])

Meth.mat %>%
  apply(1, sd) %>%
  hist(main = "Methylation: Gene standard deviation (before processing)")
```




.

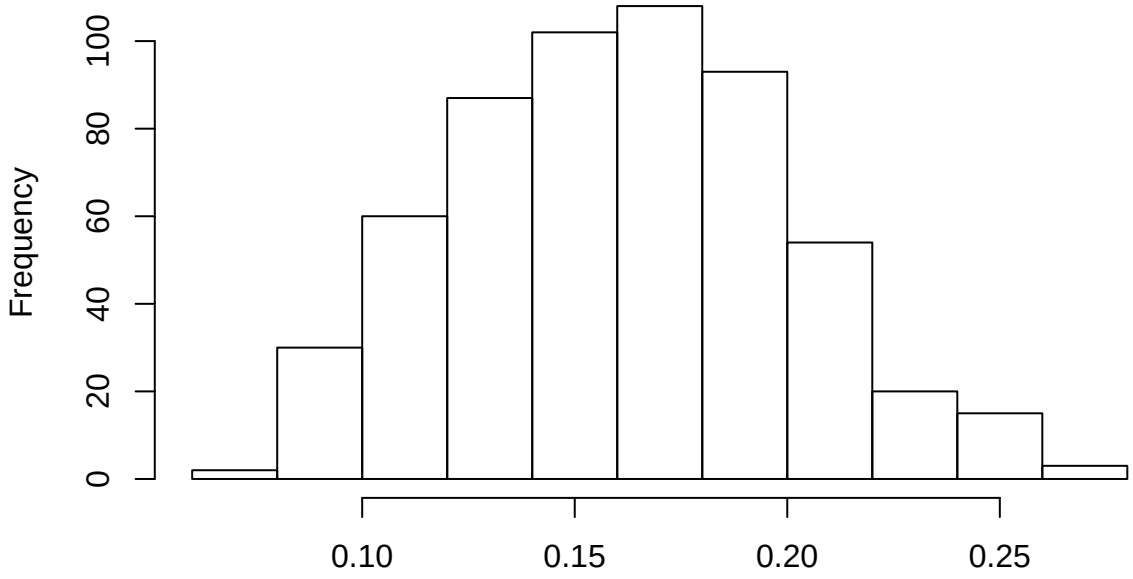
```
Meth.mat %>%
  apply(1, mean) %>%
  hist(main = "Methylation: Gene mean (after processing)")
```




.

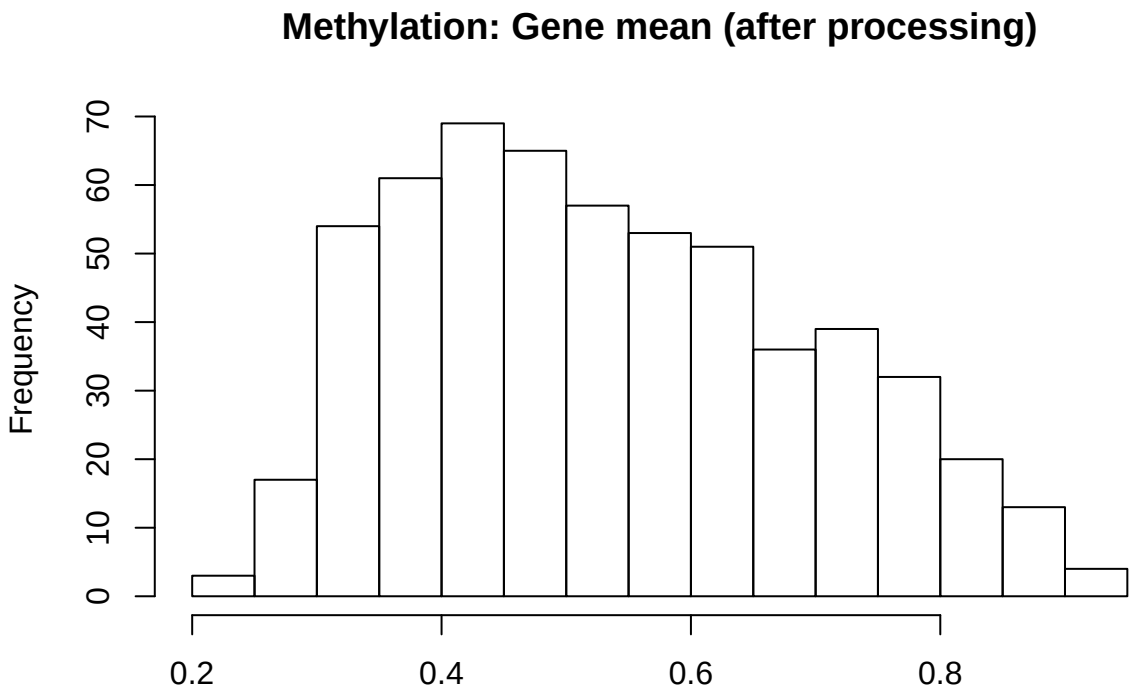
```
processedMethylation <- sqrt(Meth.mat)    ##take square root of methylation data
p_meth_pca <- prcomp(processedMethylation)
```

```
processedMethylation %>%
  apply(1, sd) %>%
  hist(main = "Methylation: Gene standard deviation (after processing)")
```




.

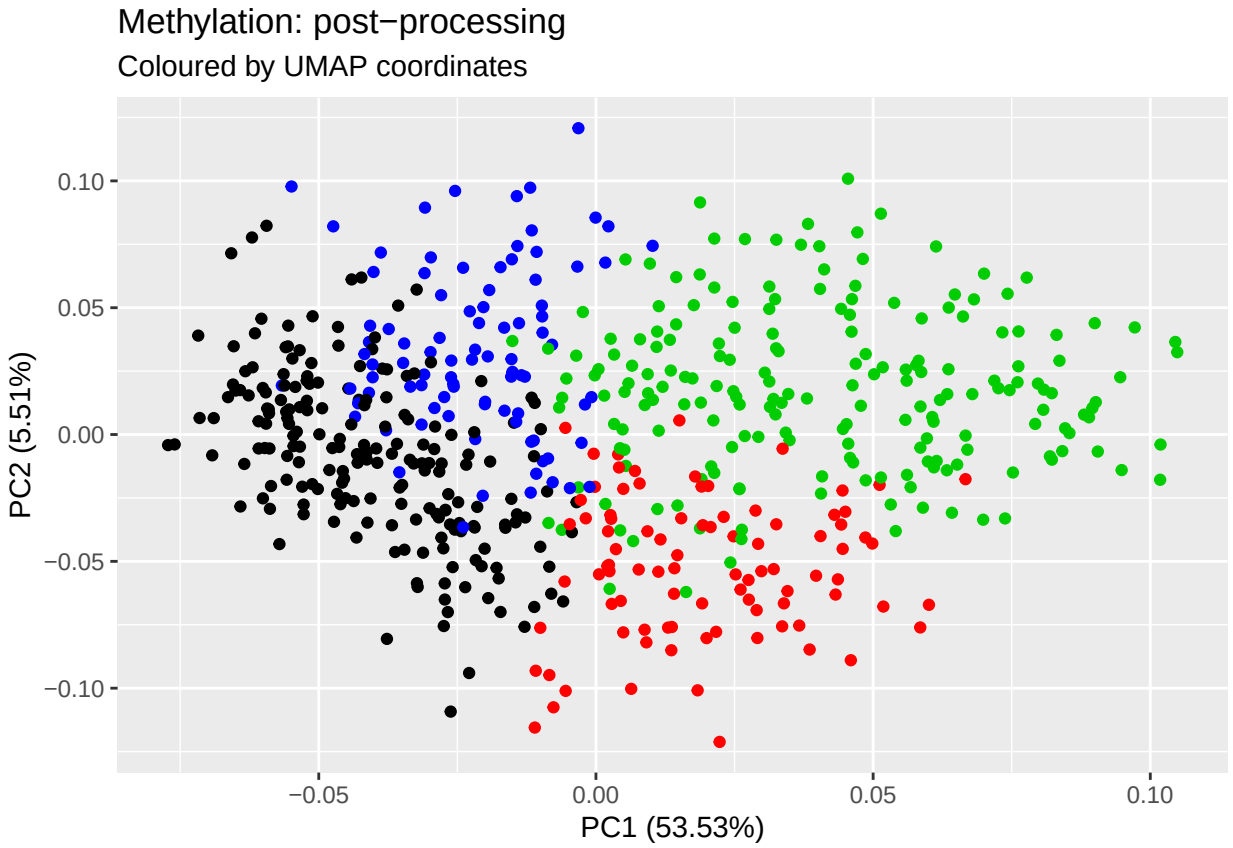
```
processedMethylation %>%
  apply(1, mean) %>%
  hist(main = "Methylation: Gene mean (after processing)")
```



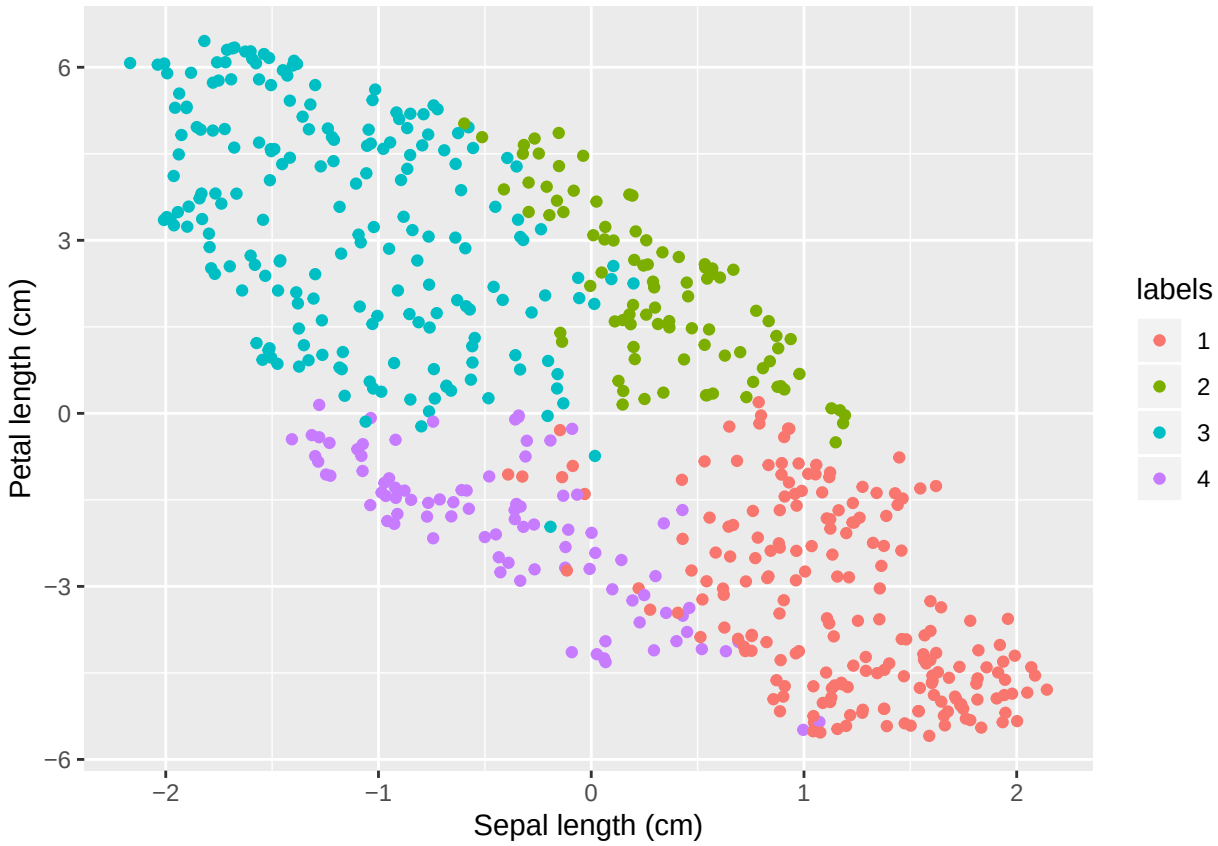

.

```
autoplot(p_meth_pca, data = processedMethylation, colour = meth_labels) +
  labs(title = "Methylation: post-processing",
       subtitle = "Coloured by UMAP coordinates")
```

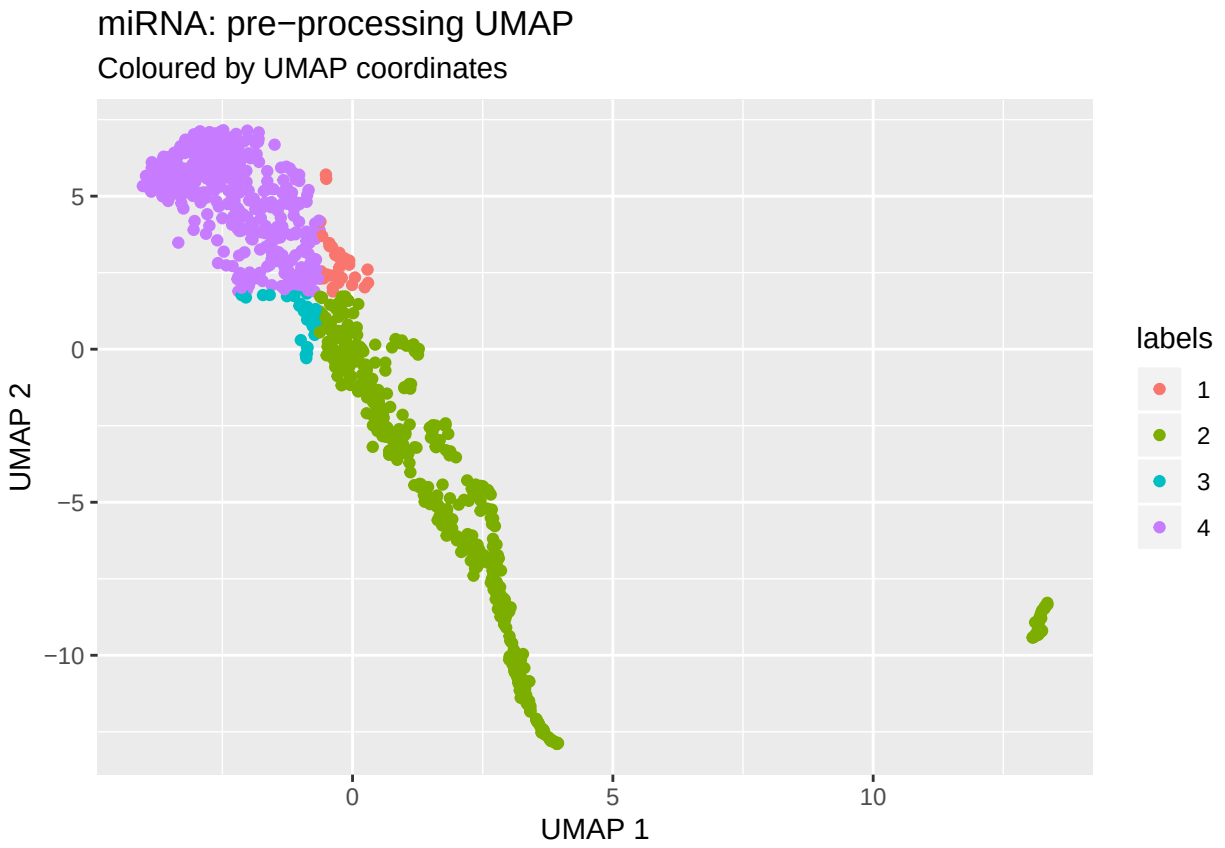
```
## Warning in if (value %in% columns) {: the condition has length > 1 and only the
## first element will be used
```




```
# ggsave(file_names[[2]][2])


p_meth_umap <- umap(processedMethylation)
p_meth_plt_data <- makeUMAPPlotData(p_meth_umap$layout, meth_labels)
plotUMAP(p_meth_plt_data) +
  labs(y="Petal length (cm)", x = "Sepal length (cm)")
```

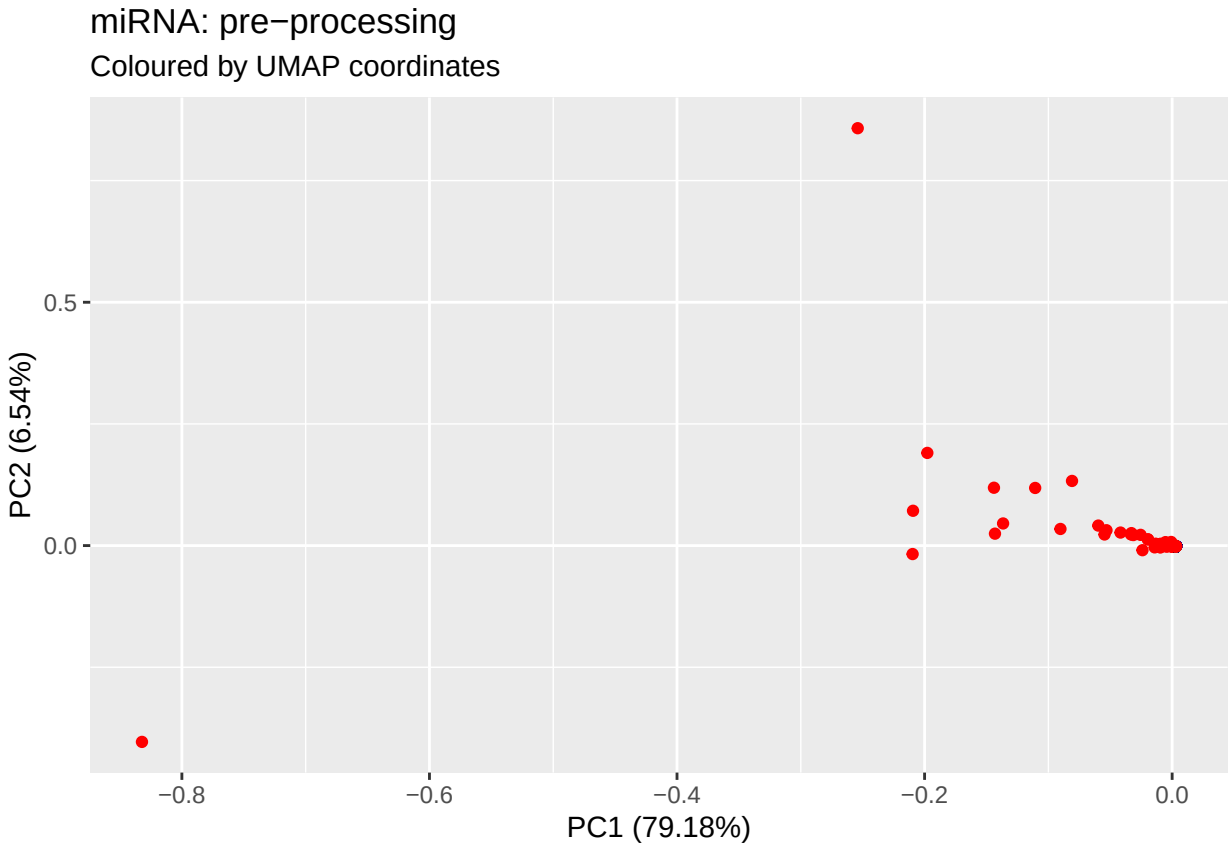



```
# Apply PCA for global structure
miRNA_pca <- prcomp(miRNA.mat)

# Apply UMAP to see local structure (should not be strongly affected by the transforms)
miRNA_umap <- umap(miRNA.mat)

# Create labels to record approximate clustering from UMAP
miRNA_labels <- makeUMAPLabels(miRNA_umap$layout)

# Create a data.frame of UMAP coordinates and labelling
miRNA_plt_data <- makeUMAPPlotData(miRNA_umap$layout, miRNA_labels)

# Plot
plotUMAP(miRNA_plt_data) +
  labs(title = "miRNA: pre-processing UMAP",
       subtitle = "Coloured by UMAP coordinates",
       x = "UMAP 1",
       y = "UMAP 2")
```




```
# ggsave(umap_file_names[3])


# miRNA_exclude <- row.names(miRNA_plt_data)[miRNA_plt_data$x >10]
# autoplot(prcomp(miRNA.mat[! row.names(miRNA.mat) %in% miRNA_exclude, ]))
autoplot(miRNA_pca, data = miRNA.mat, colour = miRNA_labels) + #, scale = 0) +
  labs(title = "miRNA: pre-processing",
       subtitle = "Coloured by UMAP coordinates")  # +
```

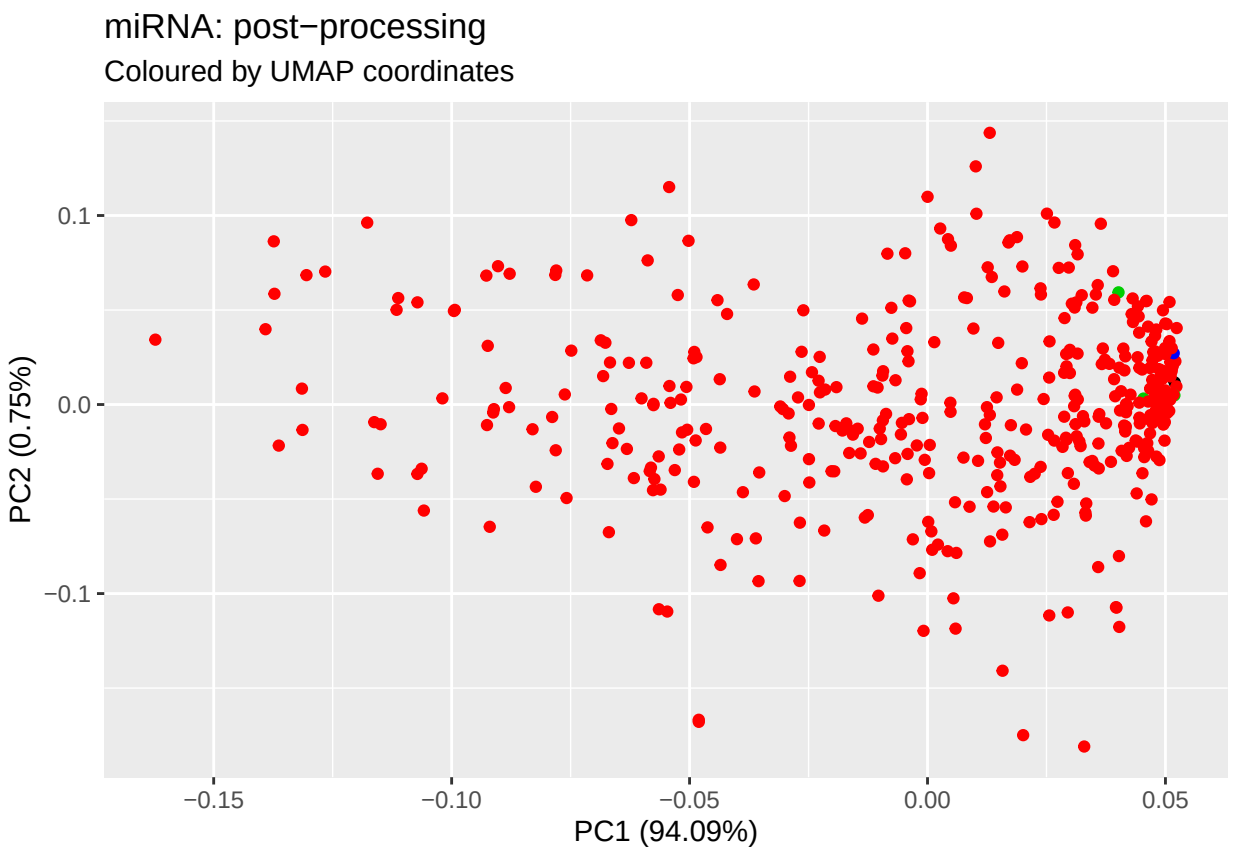
```
## Warning in if (value %in% columns) {: the condition has length > 1 and only the
## first element will be used
```



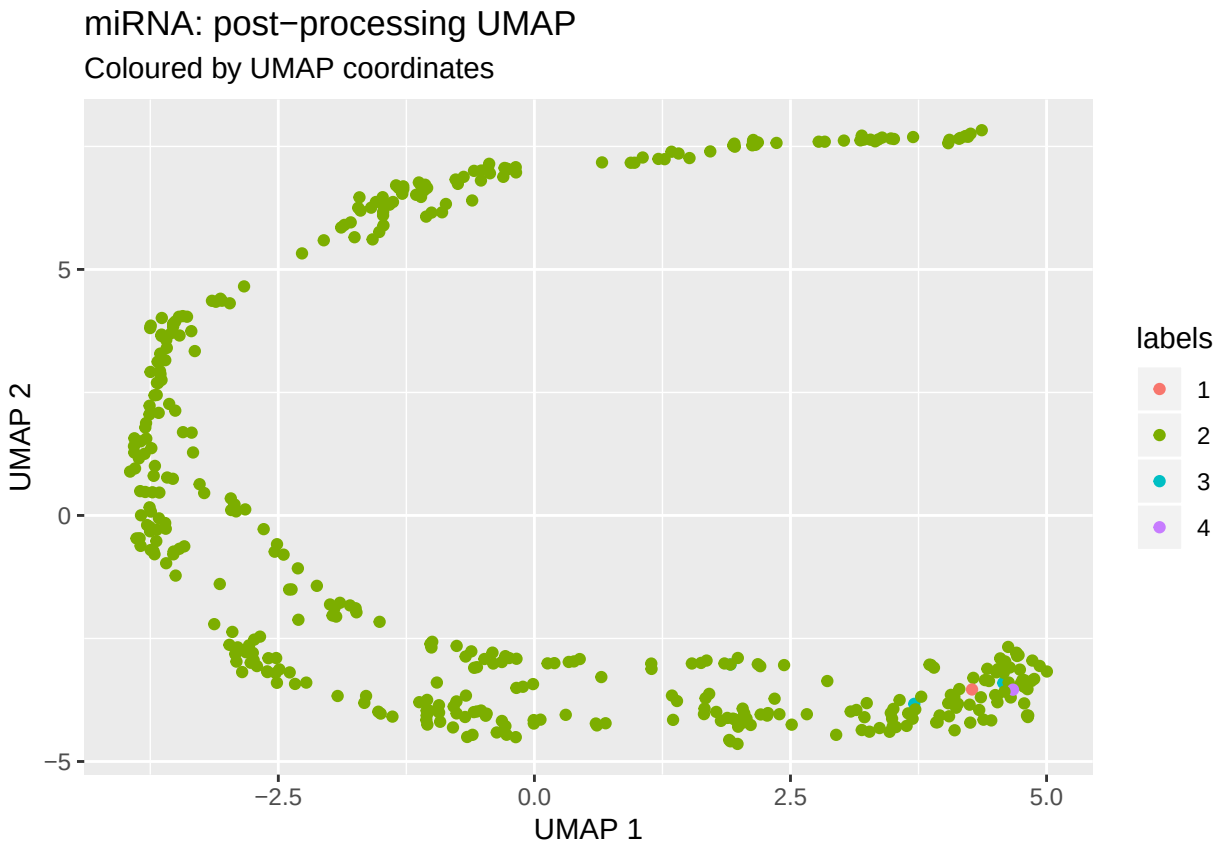

```
  # xlim(4e4, 5e4) +
  # ylim(5.5e3, 6.1e3)
# miRNA_pca$x[(miRNA_pca$x[,1] < 4e4),1:2]
#
# summary(miRNA_pca$x[,1:2])
#
# miRNA_exclude <- row.names(miRNA_pca$x)[(miRNA_pca$x[,1] < 4e4 & miRNA_pca$x[,2] < 6.1e3)]
# indices_to_drop <- ! row.names(miRNA.mat) %in% miRNA_exclude
# miRNA_reduced <- miRNA.mat[which(indices_to_drop),]
# miRNA_labels_reduced <- miRNA_labels[which(indices_to_drop)]
#
# miRNA_pca_red <- prcomp(miRNA_reduced)
#
# autoplot(miRNA_pca_red, data = miRNA_reduced, colour = miRNA_labels_reduced) +
#   labs(title = "miRNA: pre-processing",
#        subtitle = "Coloured by UMAP coordinates")

# Save the PCA plot with UMAP colouring
# ggsave(file_names[[1]][3])


# Transform the data
miRNA_to_drop <- rowSums(miRNA.mat==0) < 348*0.5
miRNA.mat    <- miRNA.mat[which(miRNA_to_drop),]
processedmiRNA       <- log(1+miRNA.mat) ##take log of miRNA data

# Take the PCA of the transformed data
```

```
p_miRNA_pca <- prcomp(processedmiRNA)

# Plot with the same labelling as previosuly applied
autoplot(p_miRNA_pca, data = processedmiRNA, colour = miRNA_labels[which(miRNA_to_drop)]) +
  labs(title = "miRNA: post-processing",
       subtitle = "Coloured by UMAP coordinates")
```

```
## Warning in if (value %in% columns) {: the condition has length > 1 and only the
## first element will be used
```




```
# ggsave(file_names[[2]][3])

p_miRNA_umap <- umap(processedmiRNA)
p_miRNA_plt_data <- makeUMAPPlotData(p_miRNA_umap$layout, miRNA_labels[which(miRNA_to_drop)])
plotUMAP(p_miRNA_plt_data) +
  labs(title = "miRNA: post-processing UMAP",
       subtitle = "Coloured by UMAP coordinates",
       x = "UMAP 1",
       y = "UMAP 2")
```



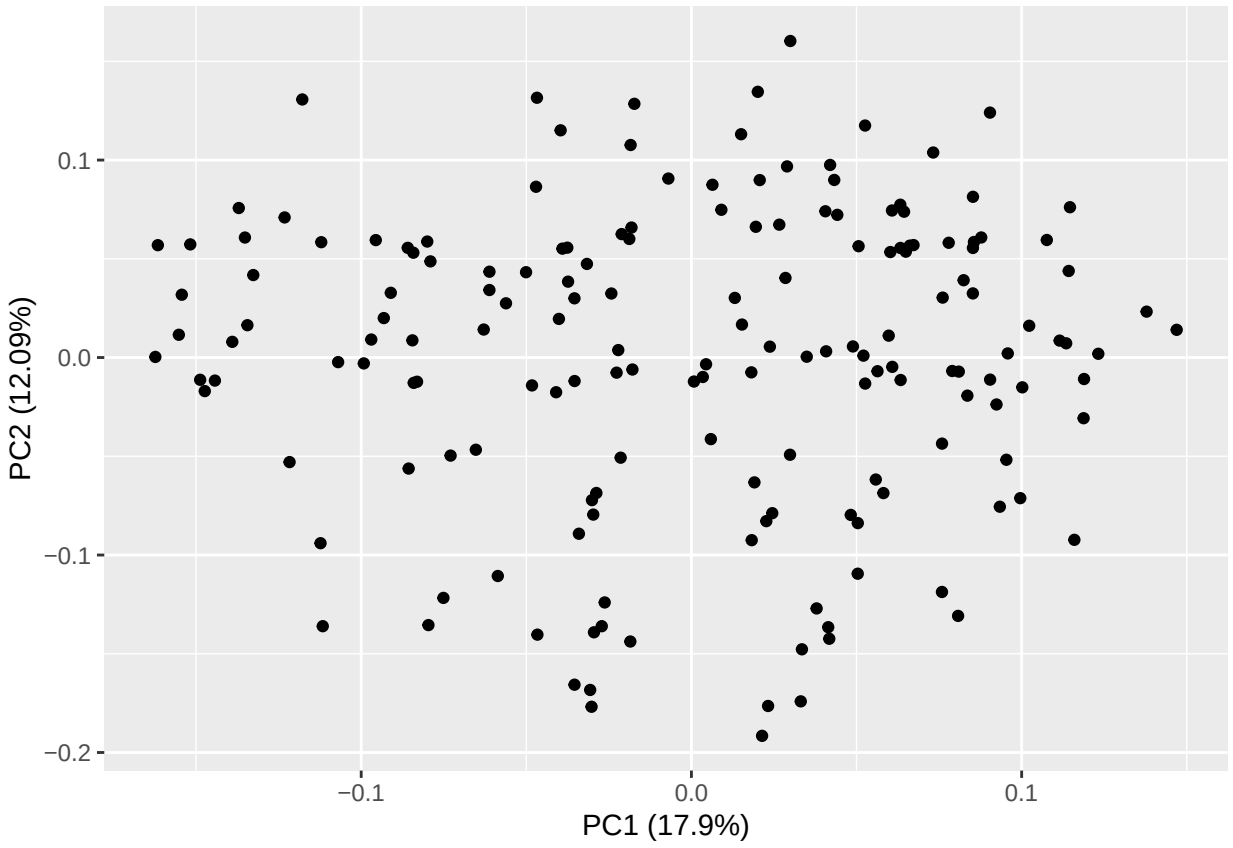

```
# ggsave("~/Documents/PhD/Year_1/Consensus_clustering/Analysis/BCC_TCGA_data/Data/miRNA_umap_post_proce


# Remove miRNAs with > 50% 0 entries
print(dim(miRNA.mat))
```

```
## [1] 423 348
```

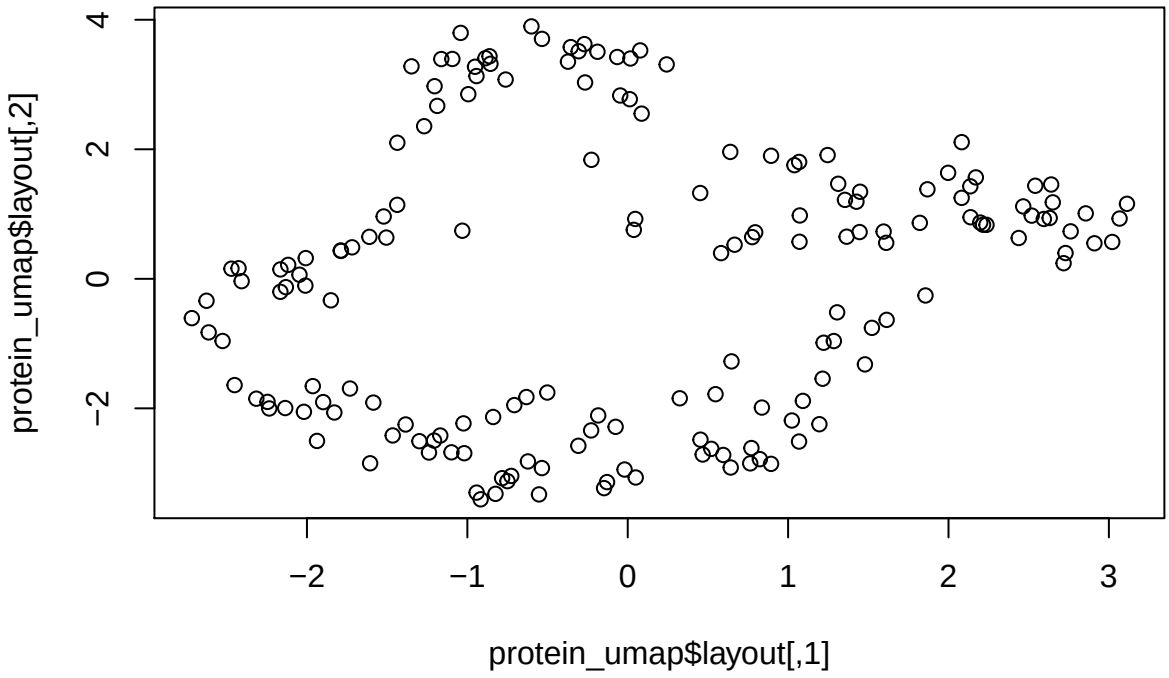
```
miRNA.mat    <- miRNA.mat[rowSums(miRNA.mat==0) < 348*0.5,]
print(dim(miRNA.mat))
```

```
## [1] 423 348
```

```
protein_pca <- prcomp(Protein.mat)
autoplot(protein_pca)
```

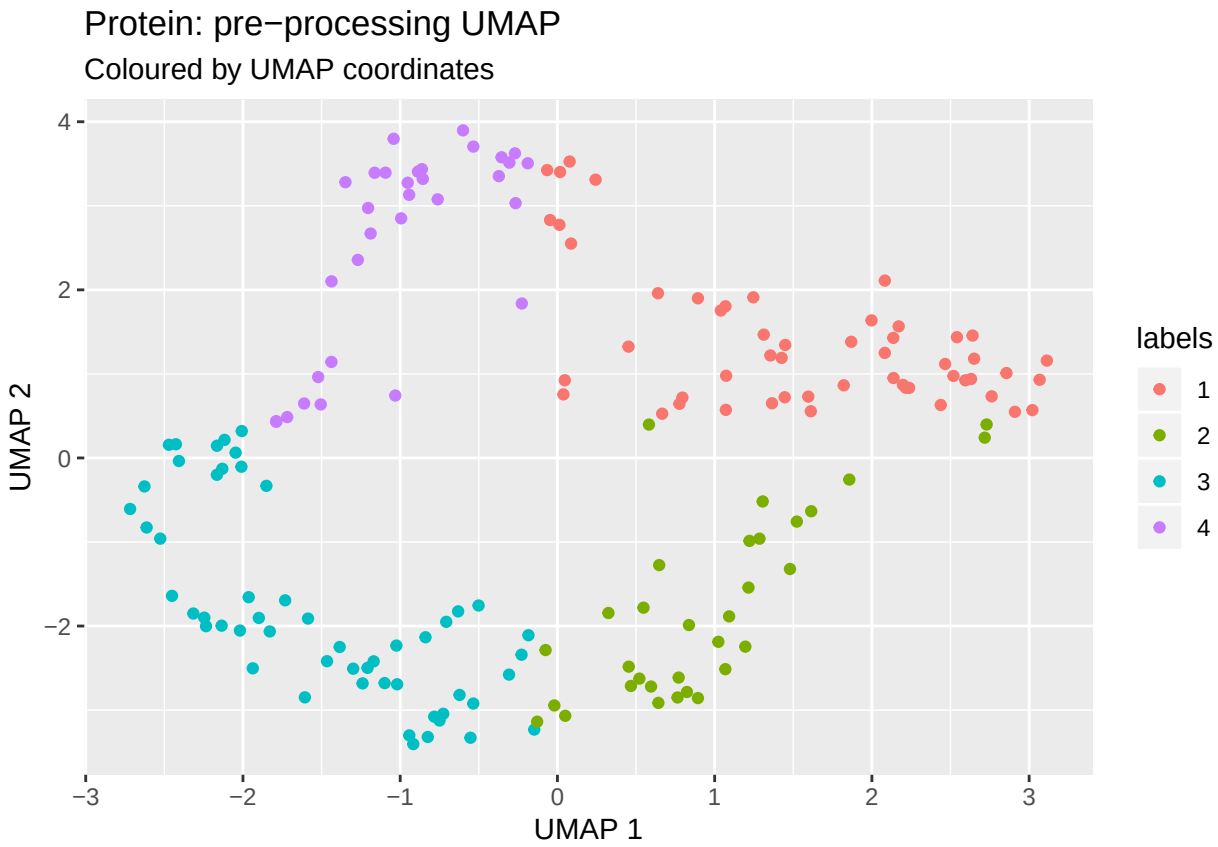



```
protein_umap <- umap(Protein.mat)
plot(protein_umap$layout)
```



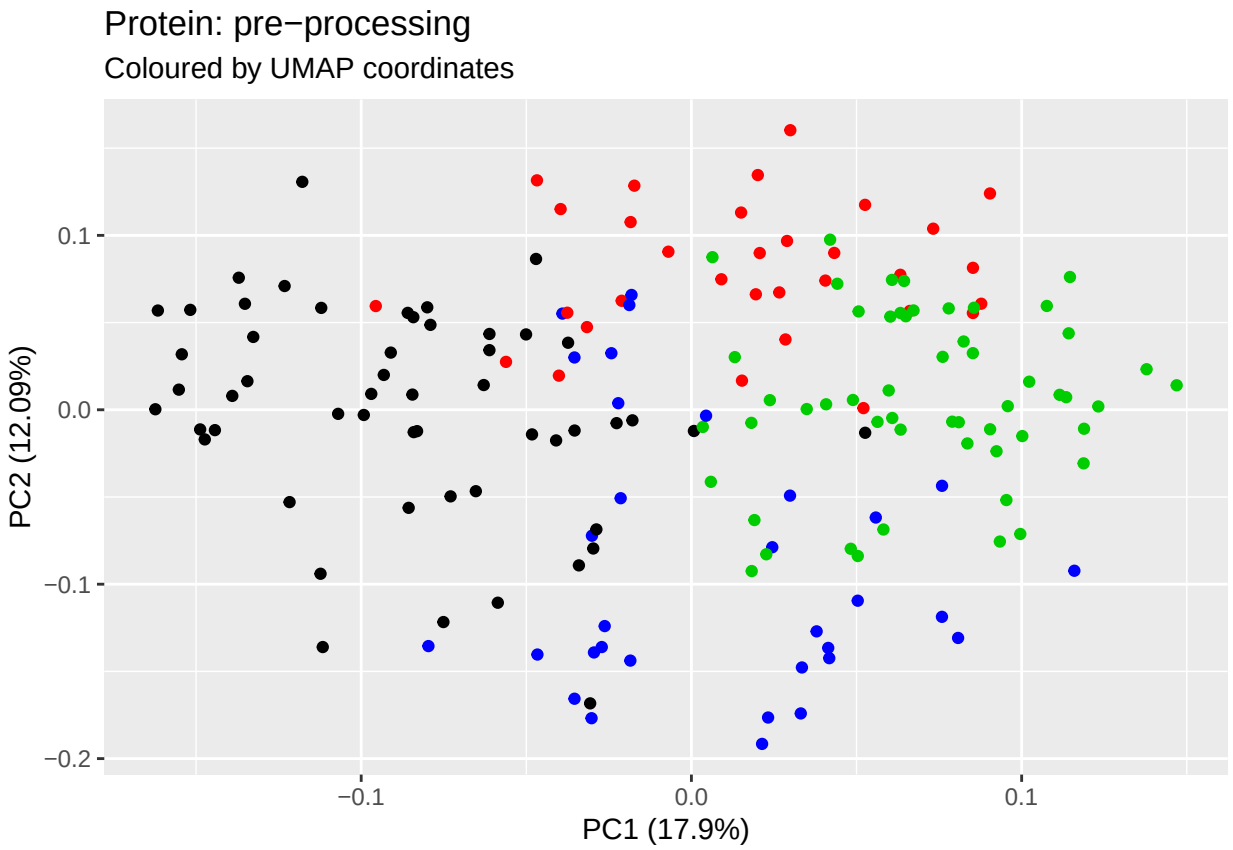

```
protein_labels <- makeUMAPLabels(protein_umap$layout)
protein_plt_data <- makeUMAPPlotData(protein_umap$layout, protein_labels)
plotUMAP(protein_plt_data) +
```

```
  labs(title = "Protein: pre-processing UMAP",
       subtitle = "Coloured by UMAP coordinates",
       x = "UMAP 1",
       y = "UMAP 2")
```



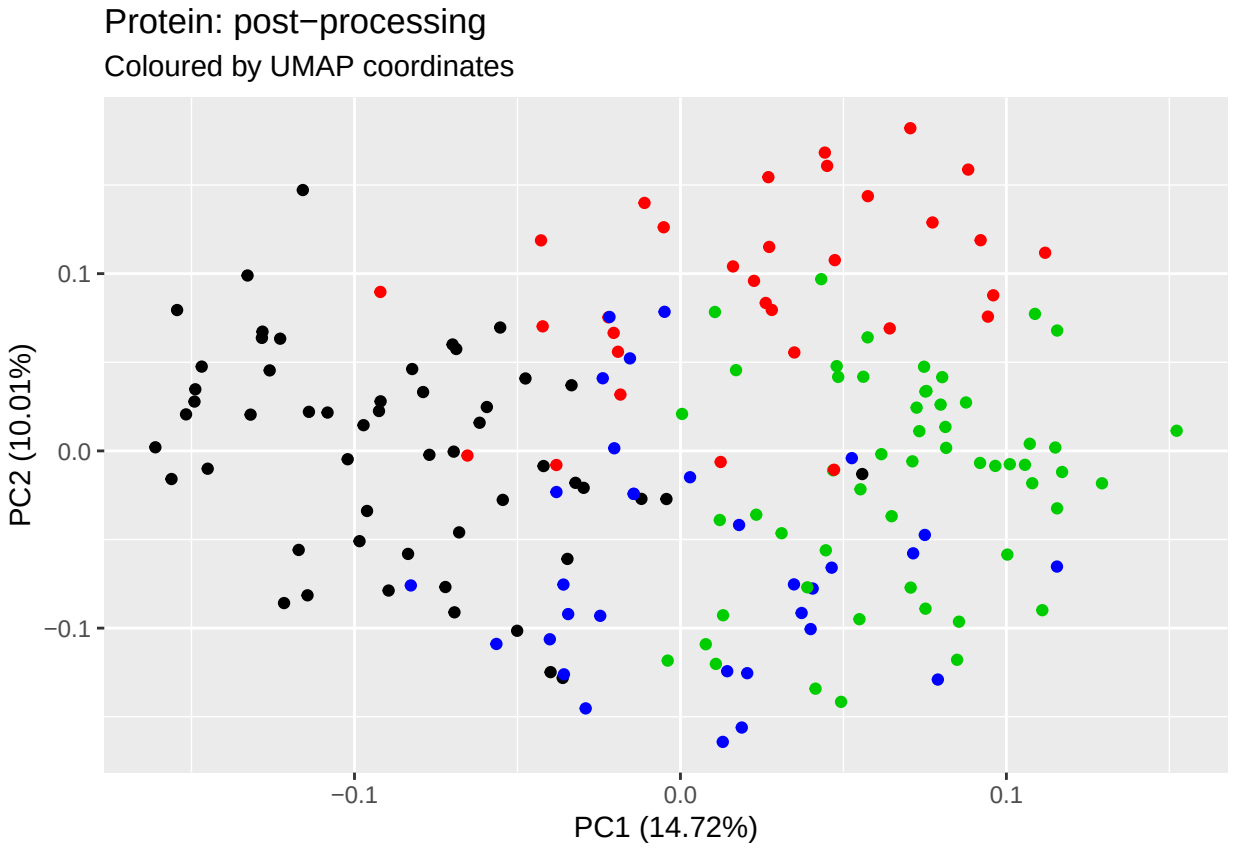

```
# ggsave(umap_file_names[4])

autoplot(protein_pca, data = Protein.mat, colour = protein_labels) +
  labs(title = "Protein: pre-processing",
       subtitle = "Coloured by UMAP coordinates")
```

```
## Warning in if (value %in% columns) {: the condition has length > 1 and only the
## first element will be used
```

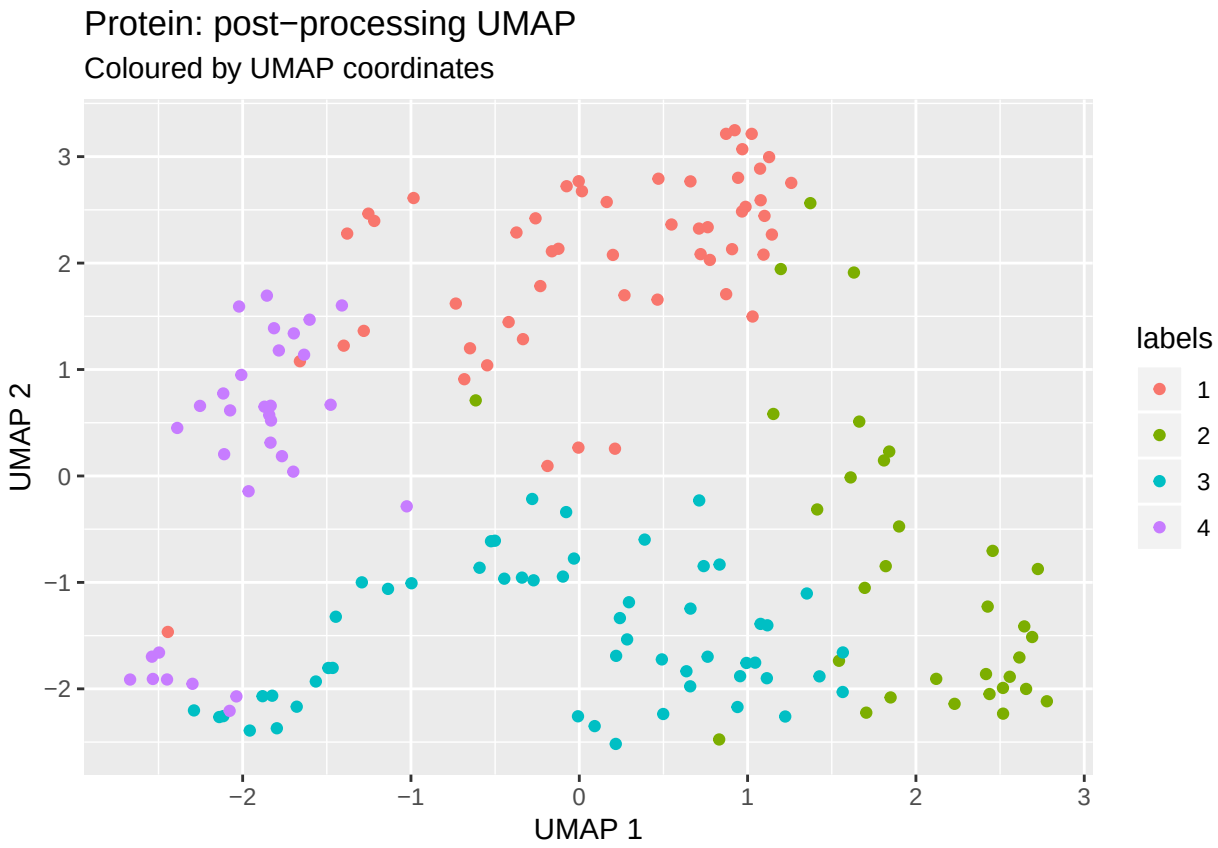



```
# ggsave(file_names[[1]][4])


processedProtein     <- scale(Protein.mat,center=TRUE,scale=TRUE) #Column center/scale protein
p_protein_pca <- prcomp(processedProtein)
autoplot(p_protein_pca, data = processedProtein, colour = protein_labels)  +
  labs(title = "Protein: post-processing",
       subtitle = "Coloured by UMAP coordinates")
```

```
## Warning in if (value %in% columns) {: the condition has length > 1 and only the
## first element will be used
```




```
# ggsave(file_names[[2]][4])


p_protein_umap <- umap(processedProtein)
p_protein_plt_data <- makeUMAPPlotData(p_protein_umap$layout, protein_labels)
plotUMAP(p_protein_plt_data) +
  labs(title = "Protein: post-processing UMAP",
       subtitle = "Coloured by UMAP coordinates",
       x = "UMAP 1",
       y = "UMAP 2")
```


Protein: post−processing UMAP
Coloured by UMAP coordinates
UMAP 2
UMAP 1
3
2
1
0
−1
−2
−2
−1
0
1
2
3
labels
1
2
3
4