Unsupervised_learning

SiCheng Zhao

2022-06-18

Import required library

```
library(iClusterPlus)
```

```
## Loading required package: parallel
```

```
library(GenomicRanges)
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
```

```
## Loading required package: S4Vectors
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
## Loading required package: IRanges

## Loading required package: GenomeInfoDb
```

```
library(gplots)
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:IRanges':
##
##     space

## The following object is masked from 'package:S4Vectors':
##
##     space

## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(lattice)
library('creditmodel')
```

```
## Package 'creditmodel' version 1.3.1
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:GenomicRanges':
##
##     subtract
```

```
library(pheatmap)
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.15.2
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'

## The following object is masked from 'package:stats':
##
##     cutree
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:GenomicRanges':
##
##     intersect, setdiff, union

## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect

## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(Rtsne)
library(cluster)
library(ggplot2)
```

Import data set

```r
# setwd("/Users/scz/Documents/AI Medical and Medician /Trimiter3/Precision Medicine")
data <- read.csv(file = 'brca_data_w_subtypes.csv')
# gene expression
GE = as.data.frame(data[, 0:604])
# Copy number mutation
CNA = as.data.frame(data[, 605:1464])
# Somatic mutation
SM = as.data.frame(data[, 1465:1713])
# Protein expression
PE = as.data.frame(data[, 1714:1936])
```

Data Cleaning

low variance filter

```r
# filter low variance with 0.97 maximum percent of unique values
GE <- low_variance_filter(GE, lvp = 0.97)
CNA = low_variance_filter(CNA, lvp = 0.97)
SM = low_variance_filter(SM, lvp = 0.97)
PE = low_variance_filter(PE, lvp = 0.97)
```

low expression filtering

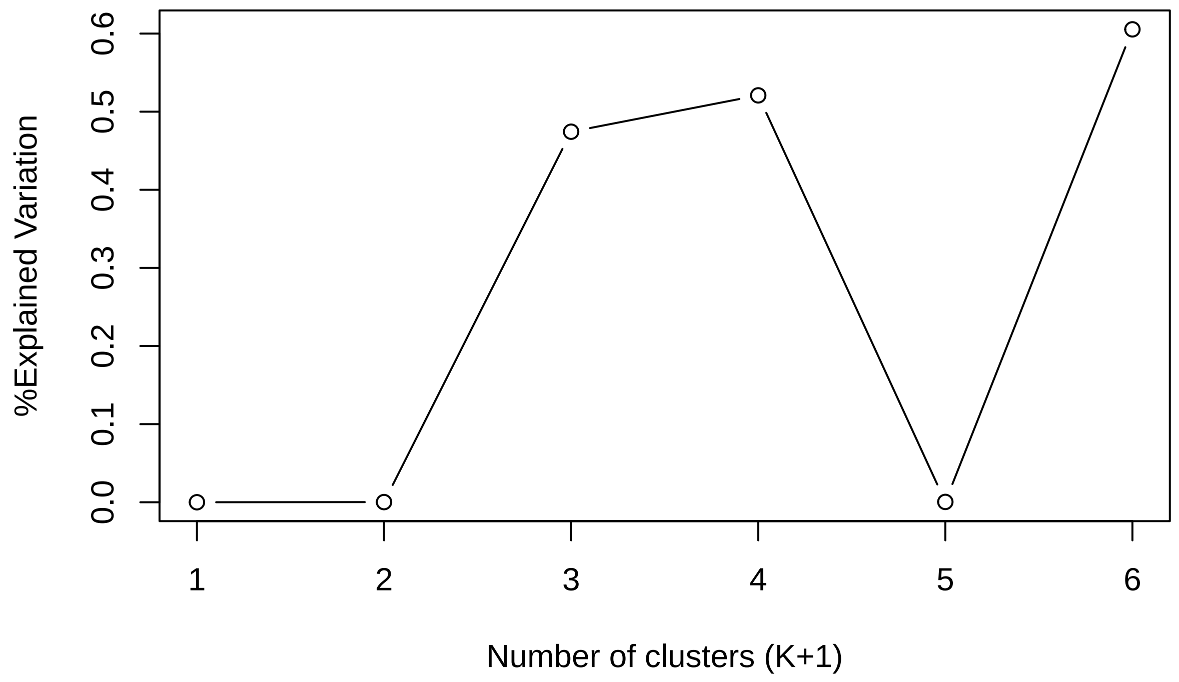
```r
# get the top 10 expression of mult-omics data
## Gene expression
GE_sort <- GE %>% apply(2, sum) %>% order(decreasing=TRUE) # group by expression
GE_new  <- GE[, GE_sort[0:10]]
## CNA
CNA_sort <- CNA %>% apply(2, abs) %>% apply(2, sum) %>% order(decreasing=TRUE)  # group by expression
CNA_new  <- CNA[, CNA_sort[0:10]]
## somatic mutation
SM_sort <- SM %>% apply(2, abs) %>% apply(2, sum) %>% order(decreasing=TRUE)    # group by expression
SM_new  <- SM[, SM_sort[0:10]]
## protein expression
PE_sort <- PE %>% apply(2, abs) %>% apply(2, sum) %>% order(decreasing=TRUE)    # group by expression
PE_new  <- PE[, PE_sort[0:10]]
```

transfer data to matrix

```r
# transfer to matrix
GE_new <- as.matrix(GE_new)
CNA_new <- as.matrix(CNA_new)
SM_new <- as.matrix(SM_new)
PE_new <- as.matrix(PE_new)
```

load result from icluster

```r
# load saved file
output2=alist()
files=grep("cv2.fit",dir())
for(i in 1:length(files)){
  load(dir()[files[i]])
  output2[[i]]=cv2.fit
}
# number of outputs
nK = length(output2)
# BIC value
BIC = getBIC(output2)
# variance of outputs
devR = getDevR(output2)

# get variance of mininum BIC
minBICid = apply(BIC,2,which.min)
devRatMinBIC = rep(NA,nK)
for(i in 1:nK){
  devRatMinBIC[i] = devR[minBICid[i],i]
}

# plot variance of mininum BIC according to number of cluster
plot(1:(nK+1),c(0,devRatMinBIC),type="b",xlab="Number of clusters (K+1)",
     ylab="%Explained Variation")
```



The optimal number of latent variables is where the curve of %Explained variation levels off. By examining the plot shown above, three is considered to be the optimal number.

Function define

```r
# normalization
cal_z_score <- function(x){
  (x - mean(x)) / sd(x)
}
# data transfer
sw_fun <- function(x) switch(x, '1'='cluster_1', '2'='cluster_2', '3'='cluster_3')
```

Data modelling

Gene expression

```r
# pheatmap
rownames(GE_new) <- paste0("p_", seq(nrow(GE_new)))
GE_norm <- t(apply(GE_new, 1, cal_z_score))
GE_heatmap <- pheatmap(GE_norm, silent = TRUE, cluster_cols = FALSE)
# get cluster from hiearchical clutering
GE_col <- cutree(tree = as.dendrogram(GE_heatmap$tree_row), k = 3)
GE_col <- data.frame(cluster = sapply(GE_col, sw_fun))
# get result from icluster
iclusters  = output2[[2]]$fit[[306]]$clusters
iclusters <- as.factor(iclusters)
GE_col$iclusters <- iclusters

# t_sne data
GE_dist <- daisy(GE_norm)   # compute distance using euclidean distance
tsne_GE <- Rtsne(GE_dist, is_distance = TRUE)
tsne_GE_data <- tsne_GE$Y %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(cluster = factor(GE_col$cluster),
         name = rownames(GE_col))
```

CNA

```r
# pheatmap
rownames(CNA_new) <- paste0("p_", seq(nrow(CNA_new)))
# get cluster from hiearchical clutering
CNA_heatmap <- pheatmap(CNA_new, clustering_distance_rows="manhattan", silent = TRUE, cluster_cols = FA
CNA_col <- cutree(tree = as.dendrogram(CNA_heatmap$tree_row), k = 3)
CNA_col <- data.frame(cluster = sapply(CNA_col, sw_fun))
# add icluster into a column
CNA_col$iclusters <- iclusters

# t_sne
CNA_dist <- daisy(CNA, metric = "manhattan")  # compute distance using manhattan distance
tsne_CNA <- Rtsne(CNA_dist, is_distance = TRUE)
tsne_CNA_data <- tsne_CNA$Y %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(cluster = factor(CNA_col$cluster),
         name = rownames(CNA_col))
```

Somatic mutation

```
# pheatmap
rownames(SM_new) <- paste0("p_", seq(nrow(SM_new)))
# get cluster from hiearchical clutering
SM_heatmap <- pheatmap(SM_new, clustering_distance_rows="manhattan", silent = TRUE, cluster_cols = FALS
SM_col <- cutree(tree = as.dendrogram(SM_heatmap$tree_row), k = 2)
SM_col <- data.frame(cluster = sapply(SM_col, sw_fun))

SM_col$iclusters <- iclusters
# t_sne
SM_dist <- daisy(SM, metric = "manhattan")     # compute distance using manhattan distance
```

```
## Warning in daisy(SM, metric = "manhattan"): binary variable(s) 1, 2, 3, 4, 5,
## 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
## 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46,
## 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66
## treated as interval scaled
```

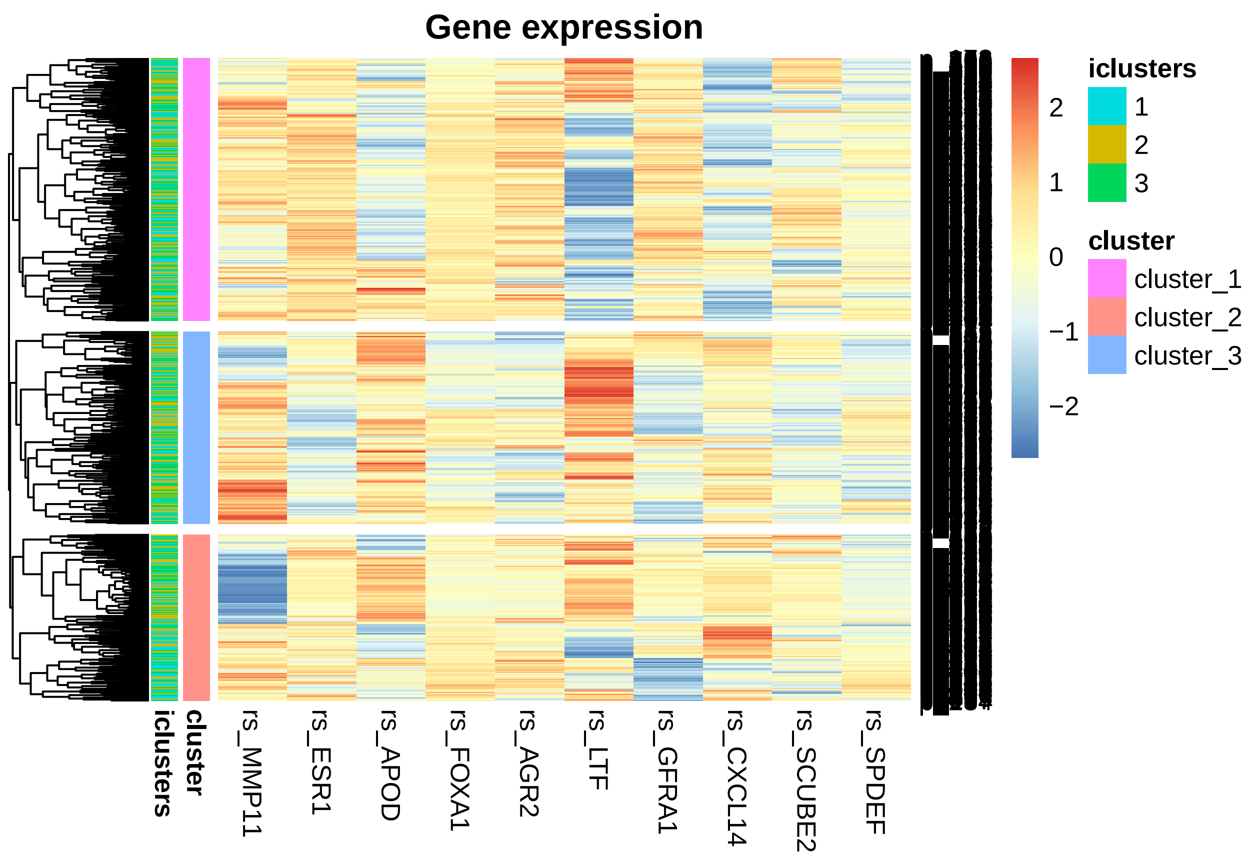
```
tsne_SM <- Rtsne(SM_dist, is_distance = TRUE)
tsne_SM_data <- tsne_SM$Y %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(cluster = factor(SM_col$cluster),
         name = rownames(SM_col))
```

Protein expression

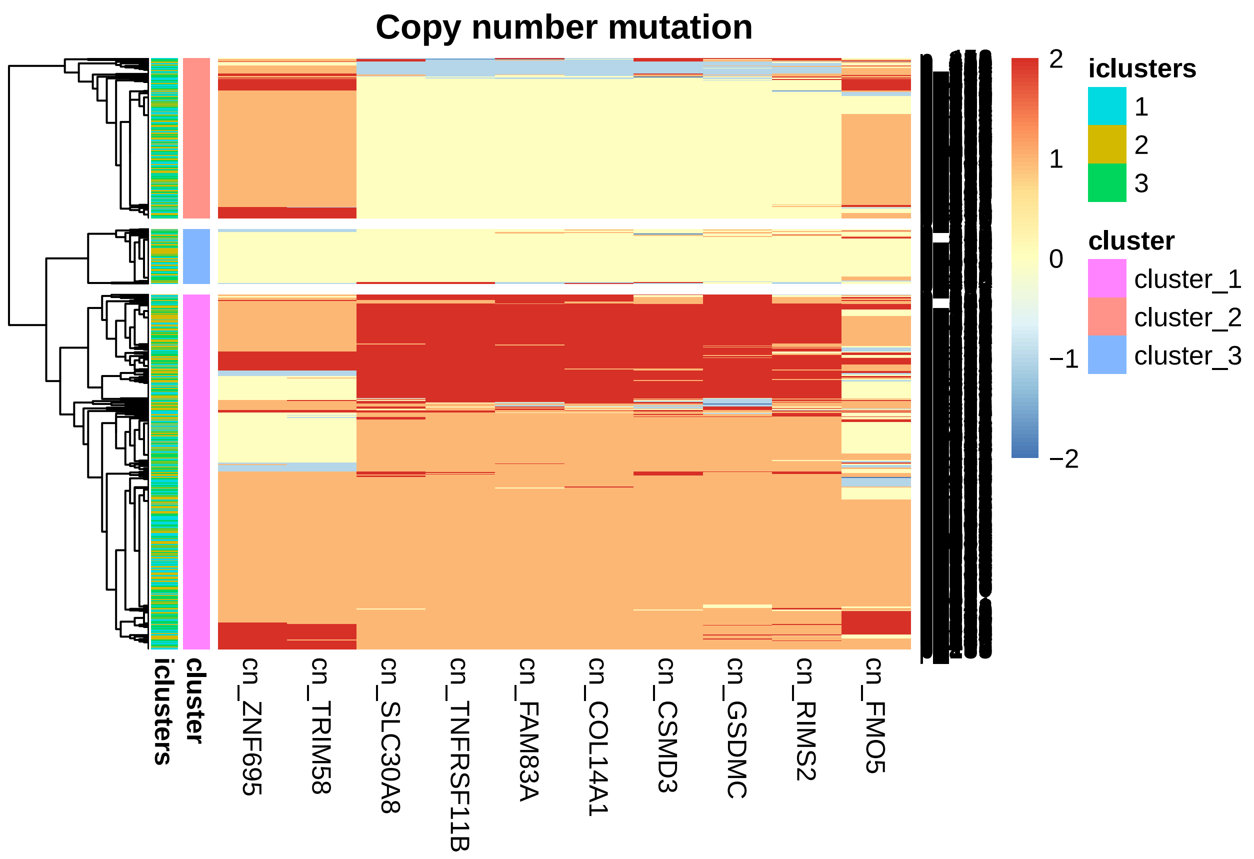
```
# pheatmap
rownames(PE_new) <- paste0("p_", seq(nrow(PE_new)))
# get cluster from hiearchical clutering
PE_norm <- t(apply(PE_new, 1, cal_z_score))
PE_heatmap <- pheatmap(PE_norm, silent = TRUE, cluster_cols = FALSE)
PE_col <- cutree(tree = as.dendrogram(PE_heatmap$tree_row), k = 2)
PE_col <- data.frame(cluster = sapply(PE_col, sw_fun))

PE_col$iclusters <- iclusters
# t_sne
PE_dist <- daisy(PE_norm, metric = "manhattan")  # compute distance using manhattan distance
tsne_PE <- Rtsne(PE_dist, is_distance = TRUE)
tsne_PE_data <- tsne_PE$Y %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(cluster = factor(PE_col$cluster),
         name = rownames(PE_col))
```

pheatmap

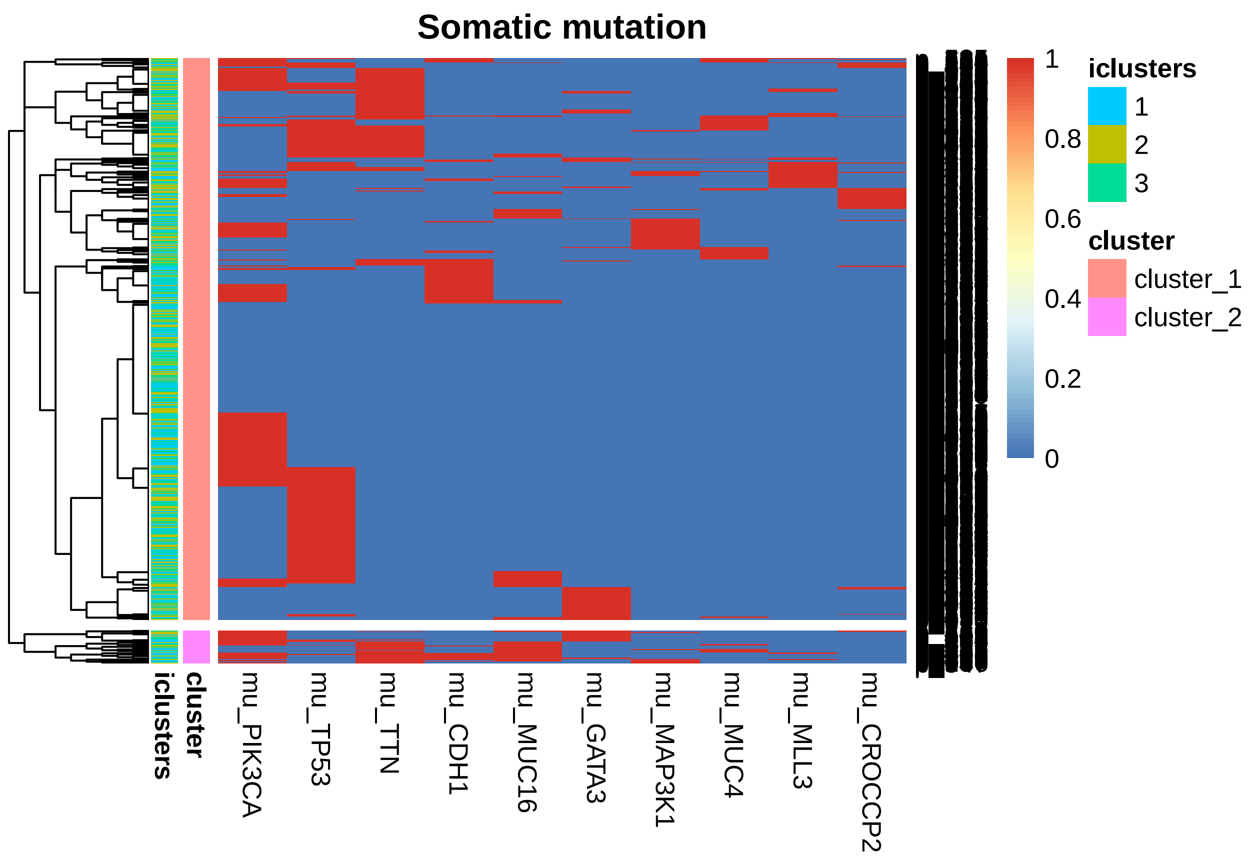
```
par(mfrow=c(1, 4))
# Gene expression
pheatmap(GE_norm, annotation_row = GE_col,
cutree_rows = 3, cluster_cols = FALSE, main='Gene expression')
```



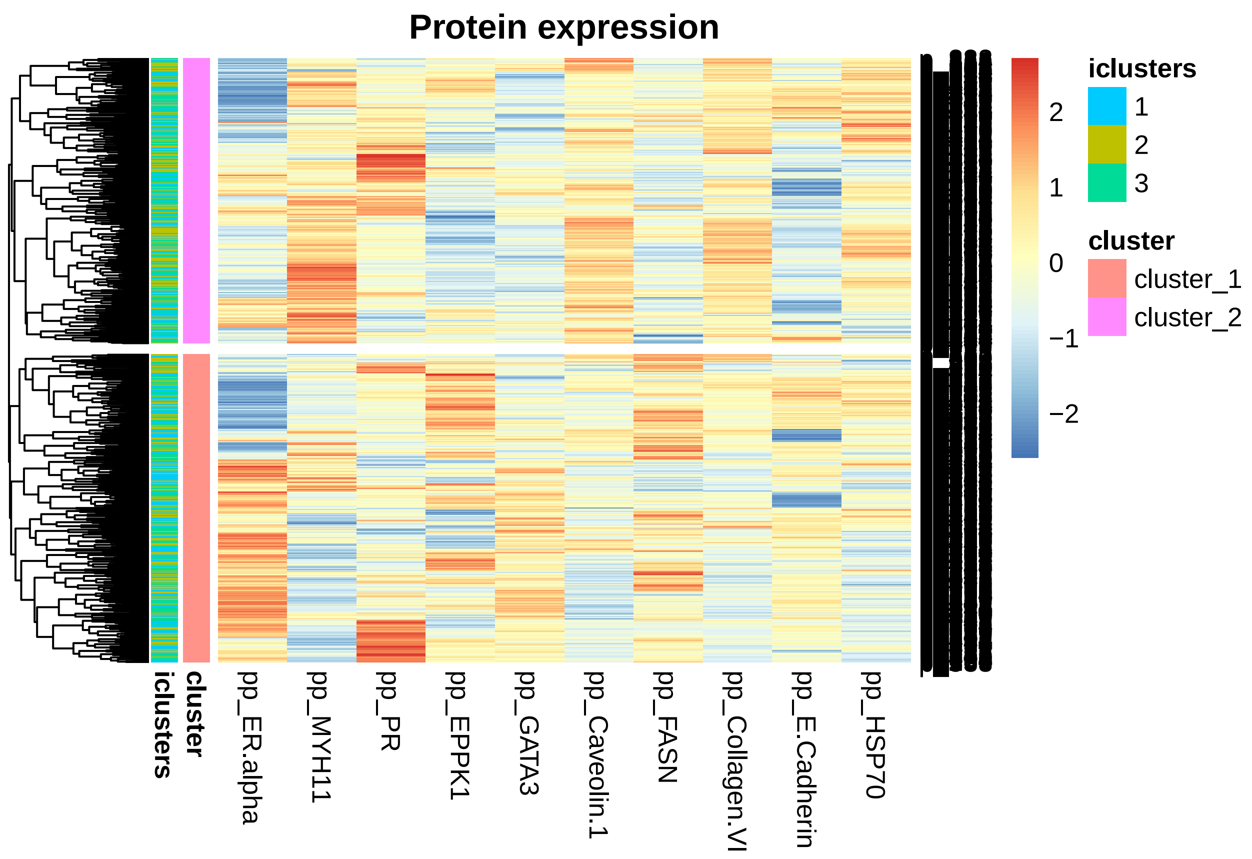
```
# Copy number muatation
pheatmap(CNA_new, annotation_row = CNA_col, clustering_distance_rows="manhattan",
cutree_rows = 3, cluster_cols=FALSE, main='Copy number mutation')
```



```
# Somatic mutation
pheatmap(SM_new, annotation_row = SM_col, clustering_distance_rows="manhattan",
cutree_rows = 2, cluster_cols=FALSE, main='Somatic mutation')
```



```
pheatmap(PE_norm, annotation_row = PE_col,
cutree_rows = 2, cluster_cols=FALSE, main='Protein expression')
```

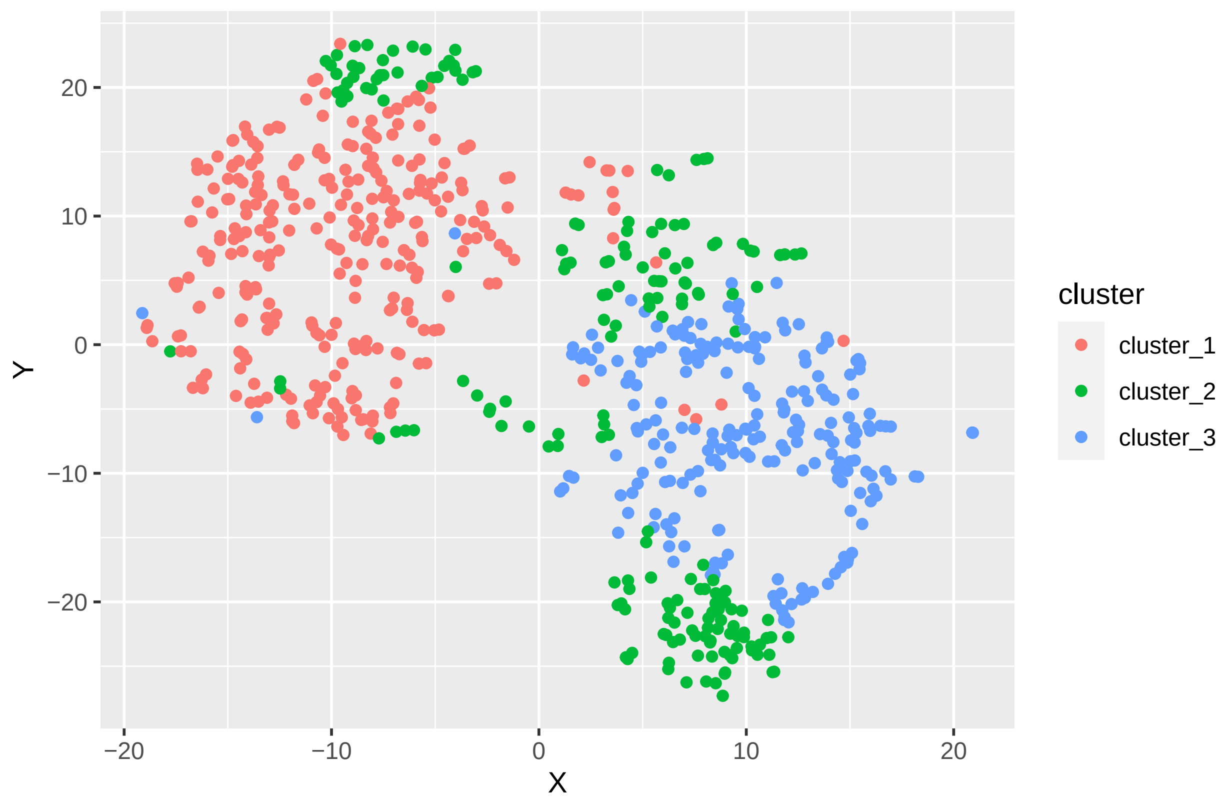


From the pheatmap we can see that there is a huge difference between the group of icluster and group clustered by hierarchical clustering.

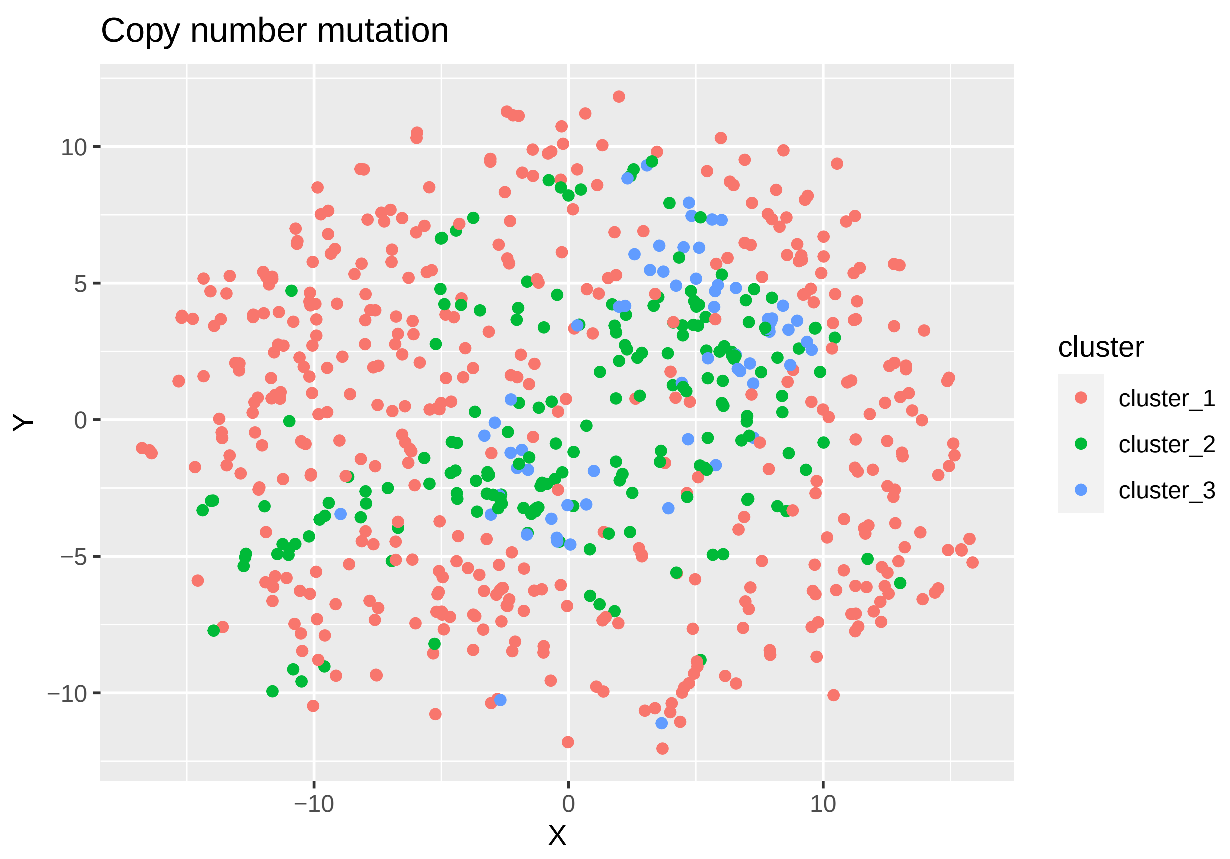
t__sne

```
par(mfrow=c(1, 4))
# Gene expression
ggplot(aes(x = X, y = Y), data = tsne_GE_data) + geom_point(aes(color = cluster)) + ggtitle("Gene expres
```

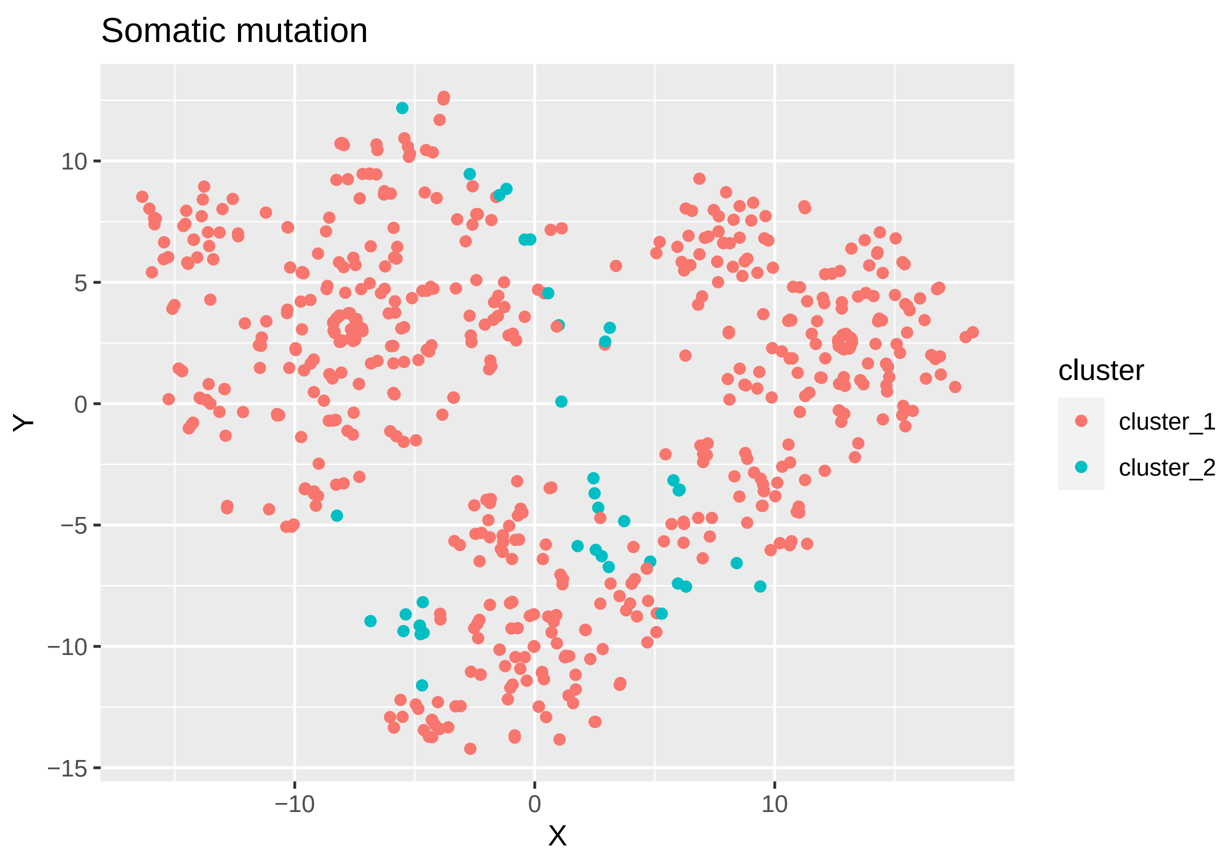
Gene expression



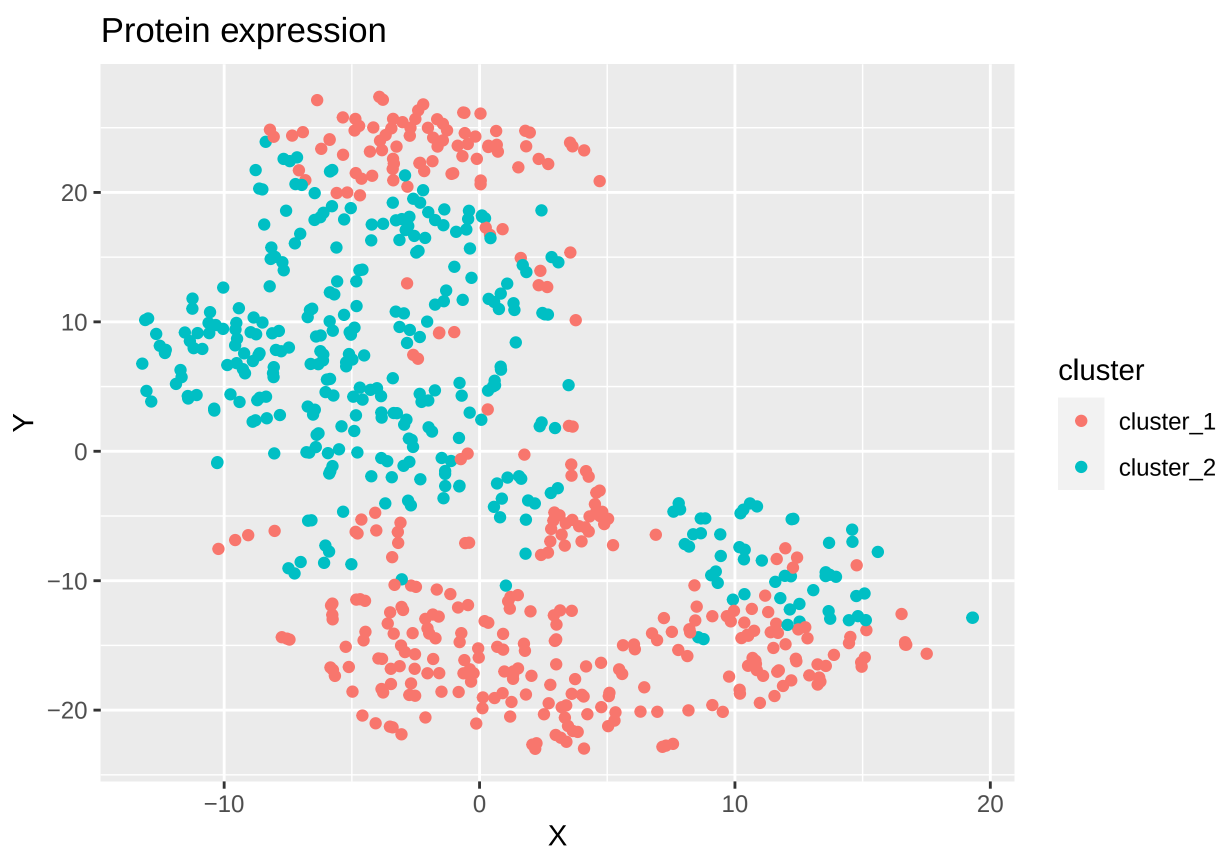
```
# Copy number mutation
ggplot(aes(x = X, y = Y), data = tsne_CNA_data) + geom_point(aes(color = cluster)) + ggtitle("Copy numb
```



```
# Somatic mutation
ggplot(aes(x = X, y = Y), data = tsne_SM_data) + geom_point(aes(color = cluster)) + ggtitle("Somatic mu
```



```
# Protein expression
ggplot(aes(x = X, y = Y), data = tsne_PE_data) + geom_point(aes(color = cluster)) + ggtitle("Protein exp
```



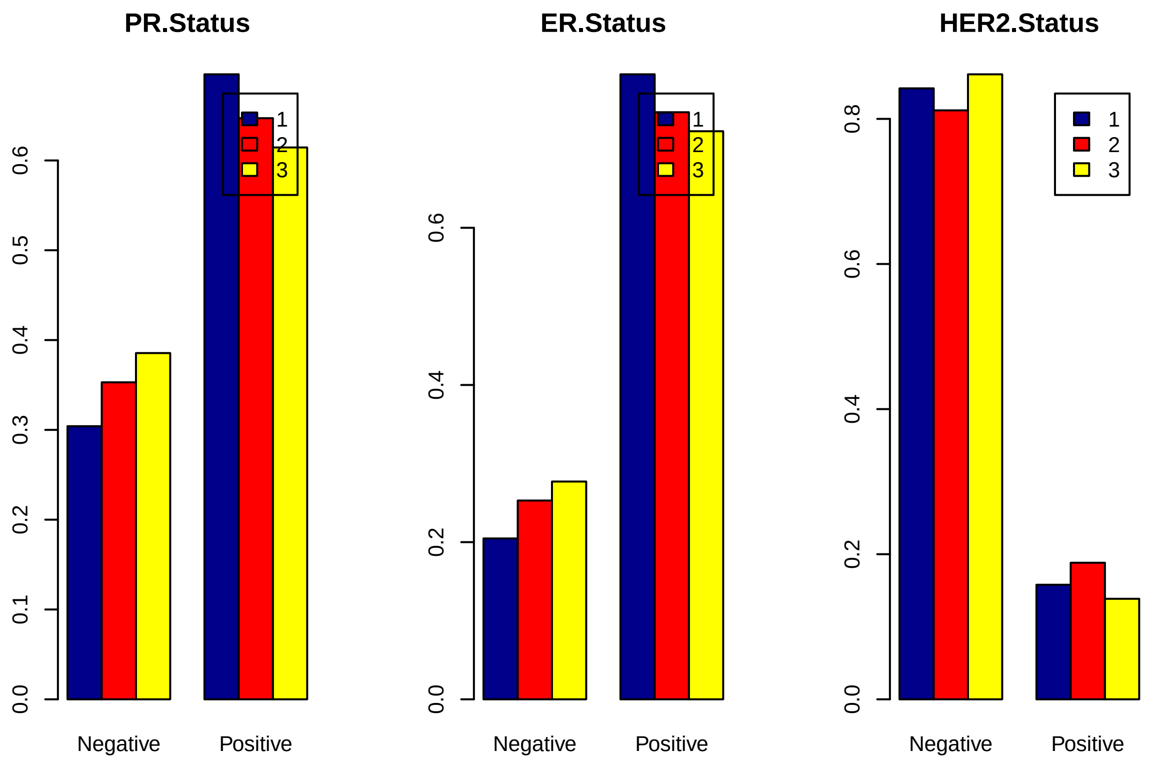
Clustering using icluster

```
# data filtering
status = as.matrix(data[, (length(data)-4) : length(data)])
status[status == 'Performed but Not Available'] <- ""
status[status == 'Indeterminate'] <- ""
status[status == 'Not Performed'] <- ""
status[status == 'Equivocal'] <- ""
status[status == 'Not Available'] <- ""

new_data = as.data.frame(cbind(GE_new, CNA_new, SM_new, PE_new, status))
# add icluster into new_data
iclusters  = output2[[2]]$fit[[306]]$clusters
new_data['icluster'] = as.factor(iclusters)
# add Gene expression clustering
new_data['GE_cluster'] = GE_col['cluster']
# add CNA clustering
new_data['CNA_cluster'] = CNA_col['cluster']
# add SM clustering
new_data['SM_cluster'] = SM_col['cluster']
# add PE clustering
new_data['PE_cluster'] = PE_col['cluster']

# get a new filtered data
new_data1 <- new_data %>% filter(PR.Status != "") %>% filter(ER.Status != "") %>% filter(HER2.Final.Sta
```

Evaluation of the clustering result

icluster

```
par(mfrow=c(1, 3))
# PR status
counts_pr = table(new_data1$icluster, new_data1$PR.Status)
barplot(counts_pr %>% prop.table(margin = 1), main="PR.Status",  # barplot using frequency of counts
        col=c("darkblue","red", "yellow"),
        legend = rownames(counts_pr), beside=TRUE)

# ER status
counts_er = table(new_data1$icluster, new_data1$ER.Status)
barplot(counts_er %>% prop.table(margin = 1), main="ER.Status",   # barplot using frequency of counts
        col=c("darkblue","red", "yellow"),
        legend = rownames(counts_er), beside=TRUE)

# HER2 status
counts_her2 = table(new_data1$icluster, new_data1$HER2.Final.Status)
barplot(counts_her2 %>% prop.table(margin = 1), main="HER2.Status",   # barplot using frequency of coun
        col=c("darkblue","red", "yellow"),
        legend = rownames(counts_her2), beside=TRUE)
```



Chi-squared test

```r
# PR status
print("PR.Status")
```

```
## [1] "PR.Status"
```

```r
chisq.test(counts_pr)
```

```
##
##  Pearson's Chi-squared test
##
## data:  counts_pr
## X-squared = 2.5035, df = 2, p-value = 0.286
```

```r
# ER.Status
print("ER.Status")
```

```
## [1] "ER.Status"
```

```r
chisq.test(counts_er)
```

```
##
##  Pearson's Chi-squared test
##
## data:  counts_er
## X-squared = 2.4886, df = 2, p-value = 0.2881
```

```r
# HER2.Status
print("HER2.Status")
```

```
## [1] "HER2.Status"
```

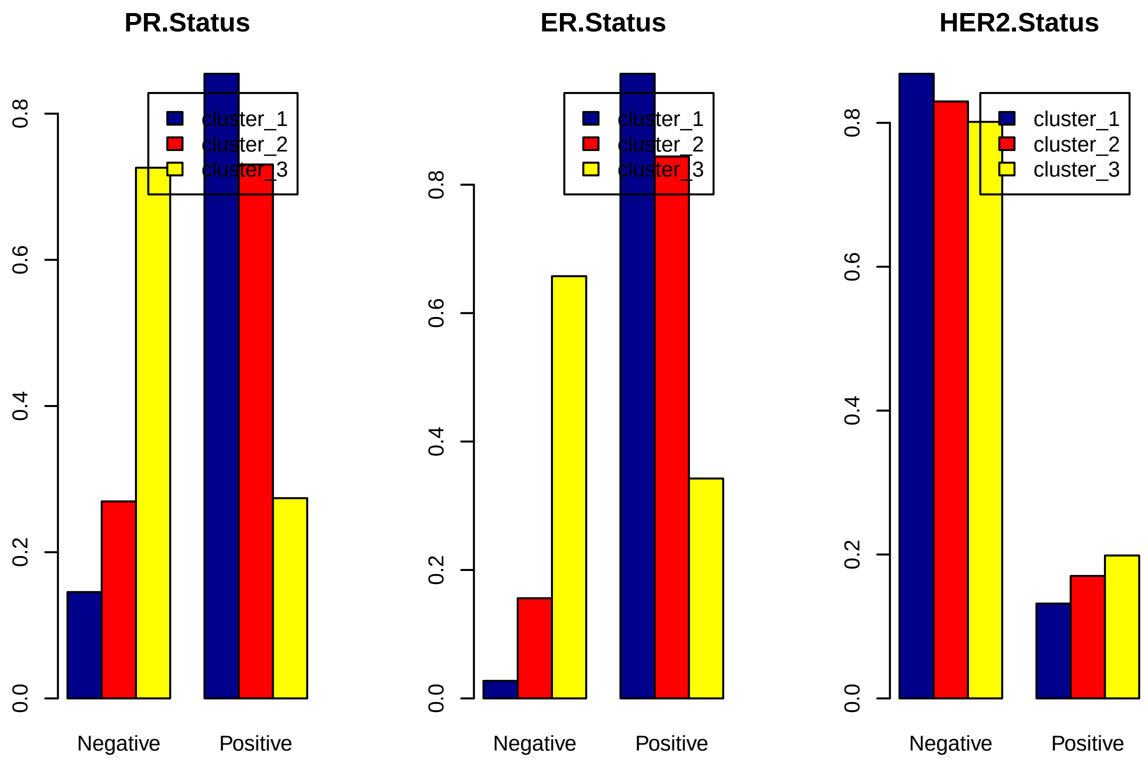
```r
chisq.test(counts_her2)
```

```
##
##  Pearson's Chi-squared test
##
## data:  counts_her2
## X-squared = 1.5571, df = 2, p-value = 0.4591
```

From the barplot, we can see that there is no obvious difference between 3 clusters on PR, ER and HER2 status. In order to validate the observation, Chi-squared test are used to evaluate the difference of frequency between three clusters. Here I set 0.05 as the threshold for p value. We can see that p value of PR status, ER status and HER2 status are all greater than 0.05. We can conclude that the different between different groups is not obvious.

Gene expression

```r
par(mfrow=c(1, 3))
# PR status
counts_pr = table(new_data1$GE_cluster, new_data1$PR.Status)
barplot(counts_pr %>% prop.table(margin = 1), main="PR.Status",
        col=c("darkblue","red", "yellow"),
        legend = rownames(counts_pr), beside=TRUE)

# ER status
counts_er = table(new_data1$GE_cluster, new_data1$ER.Status)
barplot(counts_er %>% prop.table(margin = 1), main="ER.Status",
        col=c("darkblue","red", "yellow"),
        legend = rownames(counts_er), beside=TRUE)

# HER2 status
counts_her2 = table(new_data1$GE_cluster, new_data1$HER2.Final.Status)
barplot(counts_her2 %>% prop.table(margin = 1), main="HER2.Status",
        col=c("darkblue","red", "yellow"),
        legend = rownames(counts_her2), beside=TRUE)
```



Chi-squared test

```r
# PR status
print("PR.Status")
```

```
## [1] "PR.Status"
```

```
chisq.test(counts_pr)
```

```
##
##  Pearson's Chi-squared test
##
## data:  counts_pr
## X-squared = 135.72, df = 2, p-value < 2.2e-16
```

```
# ER.Status
print("ER.Status")
```

```
## [1] "ER.Status"
```

```
chisq.test(counts_er)
```

```
##
##  Pearson's Chi-squared test
##
## data:  counts_er
## X-squared = 196.97, df = 2, p-value < 2.2e-16
```

```
# HER2.Status
print("HER2.Status")
```

```
## [1] "HER2.Status"
```

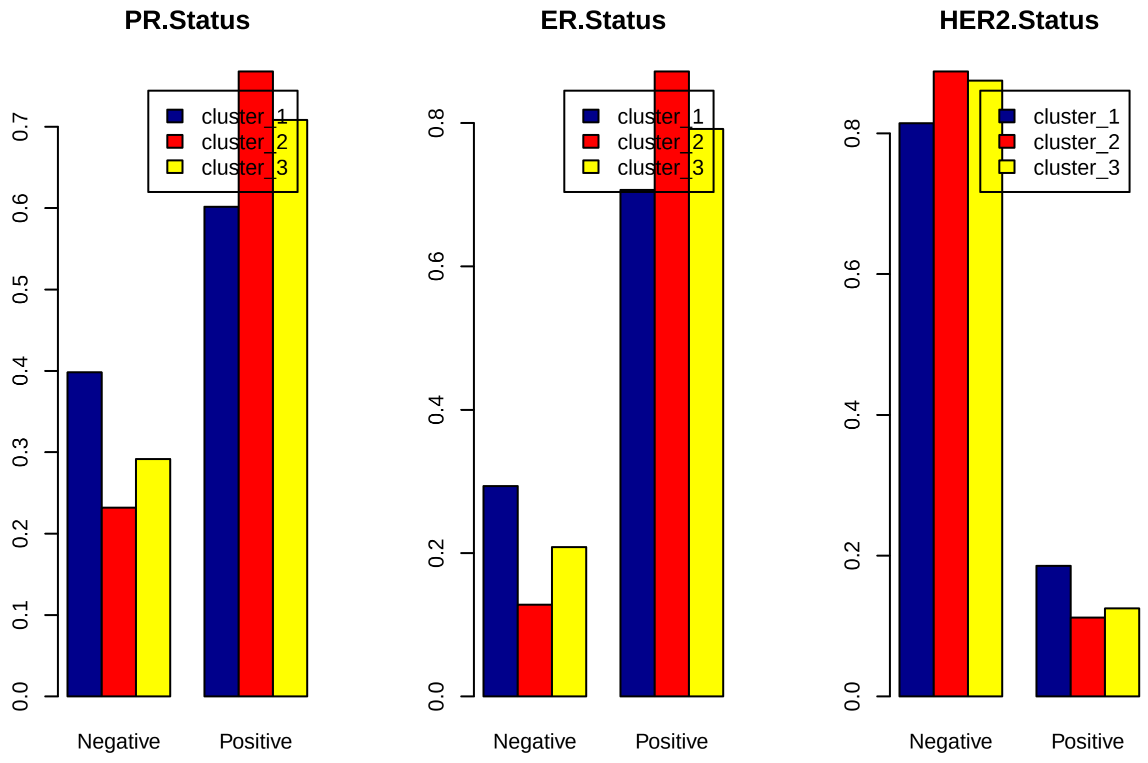
```
chisq.test(counts_her2)
```

```
##
##  Pearson's Chi-squared test
##
## data:  counts_her2
## X-squared = 2.993, df = 2, p-value = 0.2239
```

From the graph and chi squared result we can see that p value of PR status and ER status is less than 0.05. We can conclude that the difference between groups is obvious based on gene expression clustering. However, the p value on HER2 status is greater than 0.05. Therefore, difference on HER2 is not obvious.

CNA clustering

```
par(mfrow=c(1, 3))
# PR status
counts_pr = table(new_data1$CNA_cluster, new_data1$PR.Status)
barplot(counts_pr %>% prop.table(margin = 1), main="PR.Status",
        col=c("darkblue","red", "yellow"),
        legend = rownames(counts_pr), beside=TRUE)

# ER status
counts_er = table(new_data1$CNA_cluster, new_data1$ER.Status)
barplot(counts_er %>% prop.table(margin = 1), main="ER.Status",
        col=c("darkblue","red", "yellow"),
        legend = rownames(counts_er), beside=TRUE)

# HER2 status
counts_her2 = table(new_data1$CNA_cluster, new_data1$HER2.Final.Status)
barplot(counts_her2 %>% prop.table(margin = 1), main="HER2.Status",
        col=c("darkblue","red", "yellow"),
        legend = rownames(counts_her2), beside=TRUE)
```



Chi-squared test

```
# PR status
print("PR.Status")
```

```
## [1] "PR.Status"
```

```
chisq.test(counts_pr)
```

```
##
##  Pearson's Chi-squared test
##
## data:  counts_pr
## X-squared = 11.807, df = 2, p-value = 0.00273
```

```
# ER.Status
print("ER.Statis")
```

```
## [1] "ER.Statis"
```

```
chisq.test(counts_er)
```

```
##
##  Pearson's Chi-squared test
##
## data:  counts_er
## X-squared = 13.847, df = 2, p-value = 0.0009842
```

```
# HER2.Status
print("HER2.Statis")
```

```
## [1] "HER2.Statis"
```

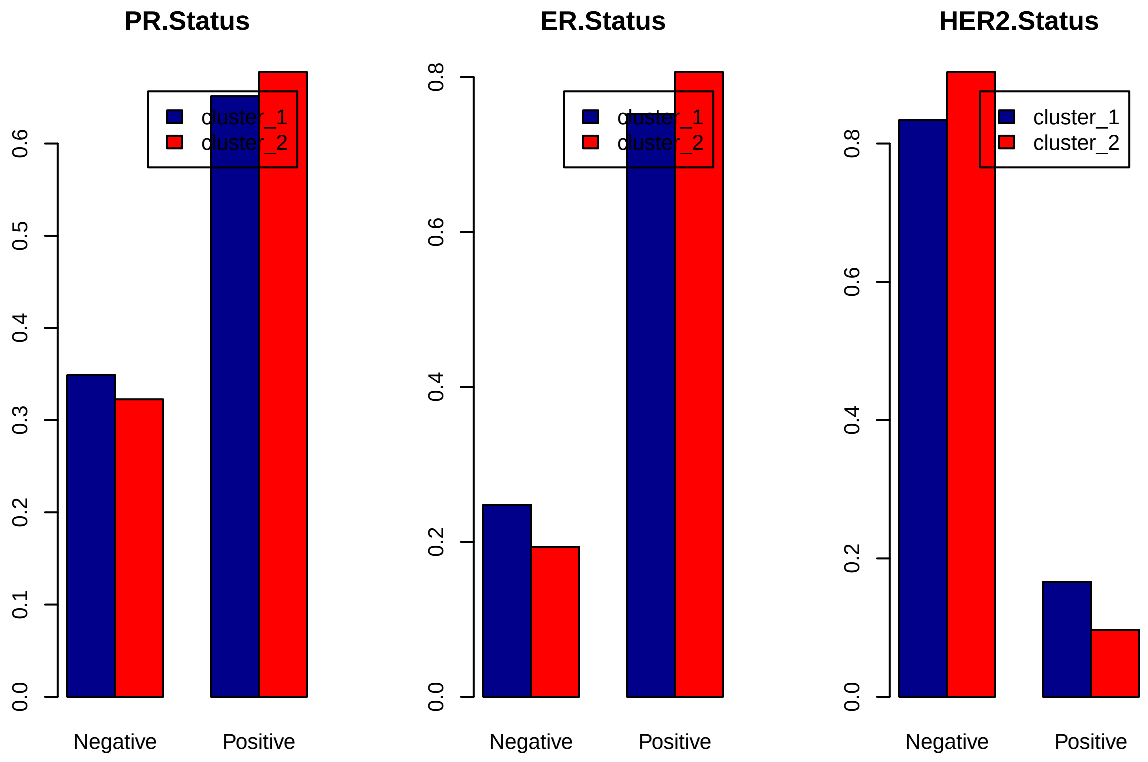
```
chisq.test(counts_her2)
```

```
##
##  Pearson's Chi-squared test
##
## data:  counts_her2
## X-squared = 4.1648, df = 2, p-value = 0.1246
```

From the graph and chi squared result we can see that p value of PR status and ER status is less than 0.05. We can conclude that the difference between groups is obvious based on CNA clustering. However, the p value on HER2 status is greater than 0.05. Therefore, difference on HER2 is not obvious.

SM clustering

```
par(mfrow=c(1, 3))
# PR status
counts_pr = table(new_data1$SM_cluster, new_data1$PR.Status)
barplot(counts_pr %>% prop.table(margin = 1), main="PR.Status",
        col=c("darkblue","red"),
        legend = rownames(counts_pr), beside=TRUE)

# ER status
counts_er = table(new_data1$SM_cluster, new_data1$ER.Status)
barplot(counts_er %>% prop.table(margin = 1), main="ER.Status",
        col=c("darkblue","red"),
        legend = rownames(counts_er), beside=TRUE)

# HER2 status
counts_her2 = table(new_data1$SM_cluster, new_data1$HER2.Final.Status)
barplot(counts_her2 %>% prop.table(margin = 1), main="HER2.Status",
        col=c("darkblue","red"),
        legend = rownames(counts_her2), beside=TRUE)
```



Chi-squared test

```
# PR status
print("PR.Status")
```

```
## [1] "PR.Status"
```

```
chisq.test(counts_pr)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  counts_pr
## X-squared = 0.010355, df = 1, p-value = 0.9189
```

```
# ER.Status
print("ER.Status")
```

```
## [1] "ER.Status"
```

```
chisq.test(counts_er)
```

```
##
```

```
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  counts_er
## X-squared = 0.21766, df = 1, p-value = 0.6408
```

```
# HER2.Status
print("HER2.Status")
```

```
## [1] "HER2.Status"
```

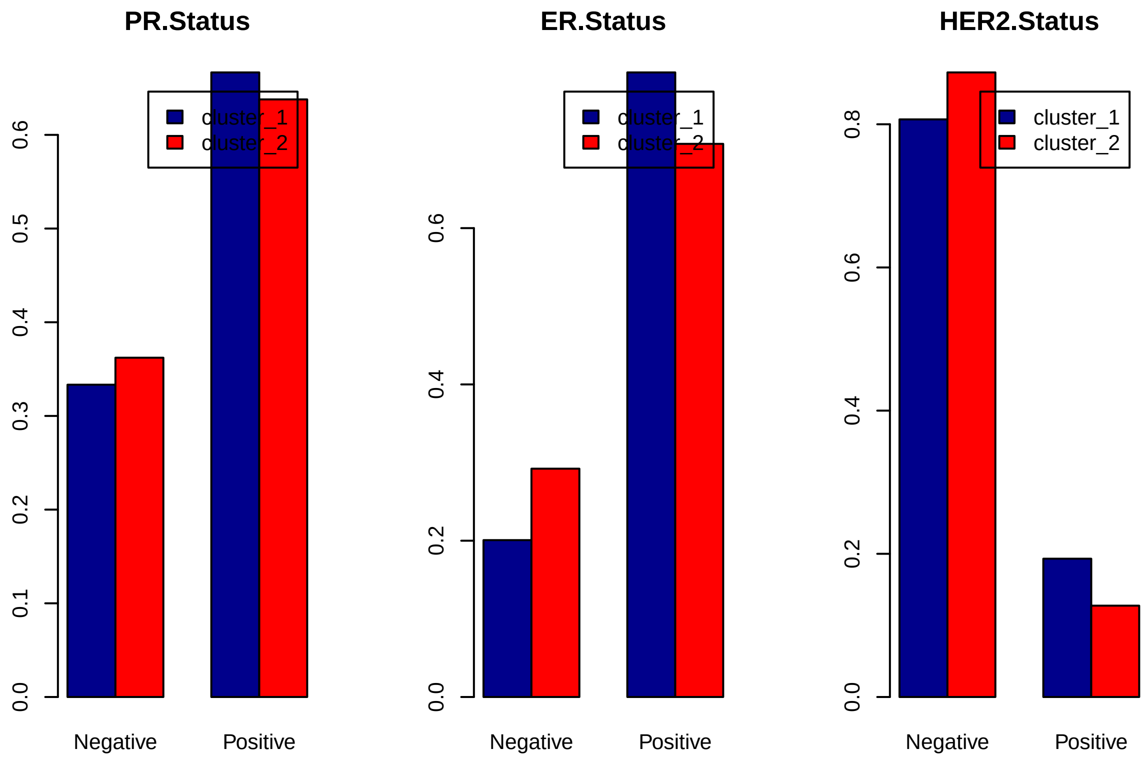
```
chisq.test(counts_her2)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  counts_her2
## X-squared = 0.58076, df = 1, p-value = 0.446
```

From the graph and chi squared result we can see that p value of PR status, ER status and HER2 status are all greater than 0.05. We can conclude that the difference between groups is obvious based on somatic mutation clustering.

PE clustering

```
par(mfrow=c(1, 3))
# PR status
counts_pr = table(new_data1$PE_cluster, new_data1$PR.Status)
barplot(counts_pr %>% prop.table(margin = 1), main="PR.Status",
        col=c("darkblue","red"),
        legend = rownames(counts_pr), beside=TRUE)

# ER status
counts_er = table(new_data1$PE_cluster, new_data1$ER.Status)
barplot(counts_er %>% prop.table(margin = 1), main="ER.Status",
        col=c("darkblue","red"),
        legend = rownames(counts_er), beside=TRUE)

# HER2 status
counts_her2 = table(new_data1$PE_cluster, new_data1$HER2.Final.Status)
barplot(counts_her2 %>% prop.table(margin = 1), main="HER2.Status",
        col=c("darkblue","red"),
        legend = rownames(counts_her2), beside=TRUE)
```



Chi-squared test

```
# PR status
print("PR.Status")
```

```
## [1] "PR.Status"
```

```
chisq.test(counts_pr)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  counts_pr
## X-squared = 0.34491, df = 1, p-value = 0.557
```

```
# ER.Status
print("ER.Statis")
```

```
## [1] "ER.Statis"
```

```
chisq.test(counts_er)
```

```
##
```

```
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  counts_er
## X-squared = 5.24, df = 1, p-value = 0.02207
```

```
# HER2.Status
print("HER2.Statis")
```

```
## [1] "HER2.Statis"
```

```
chisq.test(counts_her2)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  counts_her2
## X-squared = 3.5481, df = 1, p-value = 0.05961
```

From the graph and chi squared result we can see that p value of PR status and HER2 status are all greater than 0.05. We can conclude that the difference between groups is not obvious based on PE clustering. However, the p value on HER2 status is less than 0.05. Therefore, difference on HER2 is obvious.

Conclusion

The unsupervised learning part of this report uses two clustering models, an icluster method clustering and a hierarchical clustering. The metrics used to measure the effectiveness of clustering are PR status, ER status and HER2 status. the quality of clustering is determined by comparing the frequency differences of these metrics. If the above indicators are used as a measure, the quality of clustering by icluster is not satisfactory. icluster is divided into 3 groups, which do not show significant distribution differences in all 3 indicators. Next I used the single omic hierarchical clustering method as an aid to observe the characteristics of icluster's grouping. From the heat map, it can be seen that there is a huge difference between the groupings obtained by hierarchical clustering and those obtained by icluster. From the final results, the grouping results of Gene expression hierarchical clustering showed significant differences in PR status and ER status. copy number mutation showed significant differences in PR status and ER status indicators. somatic mutation showed no significant differences in these three indicators. Protein expression showed significant differences in ER status. Overall, the icluster multi-omics method did not perform well in the above three indicators, and the single omic clustering method produced significant differences in individual indicators. There may be two reasons for the above results, one is that the design of the metrics measuring the effectiveness of icluster clustering does not reflect the ability of icluster clustering. The second conjecture is due to the fact that single omics clustering exhibits unimpressive results in individual metrics, but affects the final presentation because other omics data play the effect of noise.