multiClust Analysis

Shane Crinion

27 March 2019

Gene expression patterns have been used in cancer research to identify significant subgroups and outcome variables such as patient survival [1]. Clustering approachs can identify patterns subgroups by using gene expression and distance metrics. MultiClust uses a clustering algorithm to identify patterns in gene expression levels that associate with a certain clinically relevant feature. The aim of this report is to identify subtype lethality in a transcriptome expression dataset by analysing the disease free survival (DFS) time.

The multiClust R package can generate a number of statistics and visualisations to identify patterns of gene expression by using gene expression and clinical data. This gene expression analysis has a workflow and a number of key objectives which entail gene ranking, gene selection and clustering approaches using multiClust.

Workflow:

The analysis workflow includes the steps:

1. Obtaining and formatting gene expression data from GEO using GEOquery.
2. Loading expression and clinical data from GEO into R in matrix format.
3. Determination of the number of desired probes for gene selection analysis.
4. Selection of a gene selection algorithm.
5. Cluster analysis using hierachical clustering.
6. Obtaining of average gene expression for hierarchael clustering.
7. Perform survival analysis using Kaplan-Meier survival plots.

Objectives:

The analysis will include a number of key objectives:

1. Determine the type of cancer under scrutiny and read up on any published papers based on its original submission to GEO.
2. Identify from this -and previous studies -the likely number of actually known subtypes for this type of cancer (e.g.for breast cancer they break into Luminal A, Luminal B etc.), as this will be useful in clustering the transcriptome data.
3. Load in & implement normalization (if required) of the cohort data.
4. Implement gene ranking for the cohort array.

- experiment with the various options, and select two, justifying this selection

5. Select the optimum number of genes for clustering.

- experiment with the various options, and select two, justifying this selection

6. Set the cluster number

- based on the known/assumed subtypes from the literature.

- determine this value directly from the data itself

7. Perform both hierarchical and k-means clustering

- determine how many samples share common cluster groupings between these two algorithms this could be visually represented using a Venn diagram
- use TreeView to examine the resulting heatmap from the hierarchical processing
  - is there evidence for gene groups correlating with sample groups?

8. Use survival analysis to investigate thoroughly whether different clusters demonstrate significantdifferences in patient outcome.

As per **Objective 1**, the type of cancer is identified as lung cancer by searching "GSE30219" in GEO (https://www.ncbi.nlm.nih.gov/geo/) 1. The experiment performs expression profiling by array using Affymetrix Human Genome U133 Plus 2.0 Array. The associated paper by Rousseaux et al. (2013) detects expression levels of tissue-restricted genes in cancer and control samples to identify "on/off" cancer biomarkers in lung cancer. The study used 293 lung tumour samples to identify 26 genes that have a strong association with poor prognosis could also identify a gene expression pattern that co-related with low immune and signalling functions 2.

As per **Objective 2**, the literature is used to identify the number of known subtypes. According to Kentaro Inamura (2017) the World Health Organisation (WHO) categorises lung cancer into two main groups: small cell lung carcinoma (SCLC) which accounts for 15% of all lung cancers and non-SCLC (NSCLC) which accounts for the other 85%. The clustering of the NSCLCs are subcategorised further into adenocarcinoma, squamous cell carcinoma (SqCC) and large cell carcinoma 3.

Rousseaux et al. (2013) cluster the data by the number of poor prognosis genes that are differentially expressed in each subset. P1 (n=121) have no expression, P2 (n = 125) have one or more ectopically expressed genes and P3 (n = 47) have three or more ectopically expressed genes. Therefore, 3 subtypes are considered in the expression analysis 2.

---

1. Obtaining and formatting gene expression data from GEO using GEOquery.

1.1 Install the MultiClust package from BioConductor.

The supporting documentation can then be found using browseVignettes("multiClust") or alternatively from Bioconductor.

```
#INSTALL BIOCINSTALLER TO USE MULTICLUST
if(!require(BiocInstaller)){
  # enable Bioconductor repositories -> add Bioc-software
  setRepositories()
  install.packages("BiocInstaller")
  library(BiocInstaller)
}

#INSTALL MULTCLUST USING BIOCMANAGER
BiocManager::install('multiClust')
```

1.2 Load the required packages.

```
#load GEO and biobase libraries
library(GEOquery)
library(Biobase)
library(multiClust)
library(preprocessCore)
library(ctc)
```

```
library(gplots)
library(dendextend)
library(graphics)
library(grDevices)
library(amap)
library(survival)
```

- GEOquery is used to get gene expression data directly from R.
- Biobase provides base functions for Bioconductor.
- preprocessCore provides normalising and matrices summarisation functions.
- ctc is used for clustering.
- gplots provides plotting functions in the analysis.
- dendextend is used to visualise and compare hierachael trees
- graphics provides base graphics.
- grDevices graphics devices and support for base and grid graphics
- amap is use for the clustering step.
- survival is used to generate the Kaplan Meier graphs.

2. Loading expression and clinical data from GEO into R in matrix format.

2.1 Obtain gene expression data: As per** Objective 3**, the data is loaded into R using the getGEO function. This extracts the gene expression data from each sample in the study.

```
# Obtain GSE series matrix file from GEO website using getGEO function
gse <- getGEO(GEO="GSE30219")
```

```
## Warning: 62 parsing failures.
##   row     col           expected    actual         file
## 54614 SPOT_ID 1/0/T/F/TRUE/FALSE --Control literal data
## 54615 SPOT_ID 1/0/T/F/TRUE/FALSE --Control literal data
## 54616 SPOT_ID 1/0/T/F/TRUE/FALSE --Control literal data
## 54617 SPOT_ID 1/0/T/F/TRUE/FALSE --Control literal data
## 54618 SPOT_ID 1/0/T/F/TRUE/FALSE --Control literal data
## ..... ....... .................. ......... ............
## See problems(...) for more details.
```

```
# Save the gene expression matrix as an object
data.gse <- exprs(gse[[1]])

# Save the patient clinical data to an object
#This should contain the other data
pheno <- pData(phenoData(gse[[1]]))
```

2.2 Format the expression and clinical data into matrices:

MultiClust analysis requires 2 text files containing the gene expression dataset and patient clinical parameters. These files are also generated from the getGEO command and then stored in matrice format using WriteMatrixToFile.

The gene expression data must be in matrix format to perform the multiClust analysis:

```
# Write the gene expression and clinical data to text files
# This is the 1st required file (gene probe expression)
WriteMatrixToFile(tmpMatrix=data.gse,
                  tmpFileName="GSE30219.expression.txt",
                  blnRowNames =TRUE,
```

```
                  blnColNames=TRUE) #contains GSM values

WriteMatrixToFile(tmpMatrix=pheno,
                  tmpFileName="GSE30219.clinical.txt",
                  blnRowNames=TRUE,
                  blnColNames=TRUE)# contains LC sample with status
```

The above commands writes two .txt files and creates 2 matrices from the GSE data derived from the command above. GSE30219.expression.txt contains gene/probe to sample information and is assigned to R variable data.gse in the previous step. GSE30219.clinical.txt is stored as R matrix pheno and contains comprehensive data for each sample on their diagnosis, characteristics, collection protocol, etc.

2.3 Identify if normalisation is required

```
head(pheno$data_processing)
```

```
## [1] The data was normalized by Robust Multi-Array average (RMA) algorithm (Genespring software; Agil
## [2] The data was normalized by Robust Multi-Array average (RMA) algorithm (Genespring software; Agil
## [3] The data was normalized by Robust Multi-Array average (RMA) algorithm (Genespring software; Agil
## [4] The data was normalized by Robust Multi-Array average (RMA) algorithm (Genespring software; Agil
## [5] The data was normalized by Robust Multi-Array average (RMA) algorithm (Genespring software; Agil
## [6] The data was normalized by Robust Multi-Array average (RMA) algorithm (Genespring software; Agil
## 2 Levels: The data was normalized by Robust Multi-Array average (RMA) algorithm (Genespring software
```

As per the second part of **Objective 3**, the gene expression data is inspected to identify if normalisatin is required.The data does not require normalisation as data_processing column states:

> *"The data was normalized by Robust Multi-Array average (RMA) algorithm (Genespring software)"*

Therefore no normalization of the data was implemented.

2.4 Load survival data

The survival data file, GSE30219-DFS-Clinical-Outcome.txt is used to identify tumour subtypes associated with increased lethality. The file contains disease free survival measurements (column 1) to current relapse status (column 2) in binary format (1=RELAPSE, 0=NO RELAPSE) which is the required format.

The data can be accessed here or can also be extracted manually from clinical using columns disease free survival in months:ch1 and relapse (event=1; no event=0):ch1.

```
# Obtain clinical outcome file
clin_file <- system.file("extdata", "GSE30219-DFS-Clinical-Outcome.txt",
    package="multiClust")

# Load in survival information file
clinical <- read.delim2(file=clin_file, header=TRUE)

# Display first few rows of the file
clinical[1:11, 1:2]
```

```
##    DFS_time_months DFS_event
## 1              121         0
## 2               21         0
## 3               86         0
## 4               10         1
## 5               81         1
## 6               68         0
## 7               52         0
## 8               NA        NA
```

```
## 9                70         0
## 10              115         0
## 11               13         1
```

2.5 Load the expression data

```
# Obtain gene expression matrix
exp_file <- system.file("extdata", "GSE30219.expression.txt", package= "multiClust")

# Load the gene expression matrix
data.exprs <- input_file(input=exp_file)
```

```
## [1] "The gene expression matrix has been loaded"
```

```
# View the matrix and confirm that gene probes are assigneed as row names
data.exprs[1:4,1:4]
```

```
##           GSM748053 GSM748054 GSM748055 GSM748056
## 1007_s_at 10.805665 10.381745 11.119014 10.169954
## 1053_at   6.9102187 7.9216137   8.17258  7.364354
## 117_at    6.0589886 5.9645586 5.9444494  6.061103
## 121_at    7.1477785 7.4169154  7.683858 7.2066803
```

The system.file command is used to safely interact with a locally contained file on R. input_file interacts with the gene expression matrix of probe/gene data to sample number (eg. "1007_s_at" corresponds to gene DDR1 and each column represents a different participant).

3. Determination of the number of desired probes for gene selection analysis

As per **Objective 5**, two options are used to select desired number of probes. Optimization of the number of probes per gene is performed to achieve the best measurement of expression per gene. The number of probes selected is dependant on the gene expression set and multiple probes per gene can provide the most accurate gene expression measurement [4].

number_probes is used to specify the number of probes to use in the gene selection analysis. The data.exprs contains the gene expression matrix for the lung cancer samples. This data will be used to distinguish the subset associated with lethality and consideration in future biopsies. There are four potential methods of selecting desired probes or genes: fixed, percent, poly and adaptive. The approaches used for this experiment are percent and poly methods:

3.1. Percent method

Percent method selects a user-specified percentages of samples to use for cluster and survival analysis. This also informs the analyst of the number of probes used.

```
# Choosing 50% of the total selected gene probes in a dataset
# Obtain gene expression matrix
exp_file <- system.file("extdata", "GSE30219.expression.txt",
    package="multiClust")

gene_num <- number_probes(input=exp_file,
    data.exp=data.exprs, Fixed=NULL,
    Percent=50, Poly=NULL, Adaptive=NULL)
```

```
## [1] "The percent gene probe number is: 27338"
```

Using 50% dataset selection limits the selection to 27,338 probes. This number of probes would indicate that there is still more than 1 probe per gene which indicates a thorough analysis.

3.2. Poly method

Poly fits three second degree polynomial functions and uses the dataset's standard deviation and mean to calculate the optimum number of probes.

```
# Example 3: Choosing the polynomial selected gene probes in a dataset
# Obtain gene expression matrix
exp_file <- system.file("extdata", "GSE30219.expression.txt",
    package="multiClust")

gene_num <- number_probes(input=exp_file,
    data.exp=data.exprs, Fixed=NULL,
    Percent=NULL, Poly=TRUE, Adaptive=NULL)
```

```
## [1] "The poly gene probe number is: 2366"
```

```
head(gene_num)
```

```
## [1] 2366
```

As the poly method considers expression data derived parameters, it may be the most accurate optimiser for probe selection number. The poly probe selection will be used for the analysis due to its expression dataset specific probe selection number.

Two other methods, fixed and adaptive, are available for prone selection. Fixed method was not used due to lack a lacked of specificity deemed insufficient for accurate analysis. Adaptive selection provides Gaussian mixture modeulling however was not used due to computational restrictions that made this method impossible.

4. Selection of a gene selection algorithm

In line with **Objective 4**, the probe_ranking function is used implement gene ranking for the cohort array. The available probe ranking methods include "CV_Rank", "CV_Guided", "SD_Rank" and "Poly". "Poly" and "SD_Rank" gene selection agorithms are used with Poly and Percent probe number selection respectively. The probe_ranking command will also use generate a .txt file containing the top genes to be used for the analysis.

Two approaches are tested for the gene selection algorithm which as "SD_Rank" and "Poly". As the probe number selection approach is "Poly", 2366 probes will be selected due to results in part 3.

Poly Ranking The Poly gene selection algorithm is explored with the Poly probe number selection below:

```
# Call probe_ranking function
# Select for 300 probes
# Choose genes using the Poly method
ranked.exprs <- probe_ranking(input=exp_file,
    probe_number=2366,
    probe_num_selection="Poly_Probe_Num",
    data.exp=data.exprs,
    method="Poly")
```

```
## [1] "The selected Poly Gene Expression text file has been written"
```

Inspect the data using head() and dim(). The filtering is evident by comparing the probe numbers in this command vs. those from data.exprs.

```
# Display the first few columns and rows of the matrix
head(ranked.exprs[1:3,1:3])
```

```
##             GSM748053 GSM748054 GSM748055
```

```
## 209125_at    4.400830  4.349534  3.661923
## 209988_s_at  3.078285  3.079797  3.467936
## 218835_at   13.541261 12.944545  9.725079
```

```
# Display dimensions to inspect probe number selected
dim(ranked.exprs)
```

```
## [1] 2366  307
```

The poly ranking method, described in **3.2**, uses the same algorithm for probe number selection and gene selection algorithm. The Poly approach uses dataset derived parameters to extract to select genes and will be used going forward for this specificity.

SD Rank:

```
# Call probe_ranking function
# Select for 300 probes
# Choose genes using the SD_Rank method
ranked.exprs <- probe_ranking(input=exp_file,
    probe_number=2366,
    probe_num_selection="Poly_Probe_Num",
    data.exp=data.exprs,
    method="SD_Rank")
```

```
## [1] "The selected SD_Rank Gene Expression text file has been written"
```

The results are inspected using head()

```
head(ranked.exprs[1:3, 1:3])
```

```
##             GSM748053 GSM748054 GSM748055
## 209125_at   2.2074293 2.1561330  1.468522
## 209988_s_at 0.8848845 0.8863963  1.274535
## 223678_s_at 9.1963140 8.4441530  3.763430
```

The SD_Rank gene expression is used going forward to identify the variation in gene expression that is cluster specific.

5. Cluster analysis using hierachical clustering

5.1 Divide the samples into clusters using the Fixed approach

number_clusters is used in to cluster sample subtypes by their gene expression patterns as per **Objective 6**. There are two possible options, "Fixed" and "Gap Statistic", of which the Fixed is used. The alternative clustering option, gap_statistic is not used due to computational limitations. The gap_statistic would expectedly provide more accurate results as it would determine the cluster number to optimally divide samples into.

```
# Call the number_clusters function
# data.exp is the original expression matrix object outputted from
# the input_file function
# User specifies that samples will be separated into 3 clusters
# with the "Fixed" argument
cluster_num <- number_clusters(data.exp=data.exprs, Fixed=3,gap_statistic=NULL)
```

```
## [1] "The fixed cluster number is: 3"
```

The above will perform the analysis to create 3 distinct clusters using gene expression data. The literature indicates that there are 3 subtypes classified by the number of genes of poor prognosis genes found in the

patient. **Objective 6** also states to extract this info from the data; hierachical clustering should indicate 1 group with no expression variation and 2 groups linked by increased expression of key poor prognosis genes.

5.2 Perform Hierarchical Clustering of Genes/Probes and Samples

cluster_analysis is used to perform hierachical and Kmeans clustering, in line with **Objective 7**. The Euclidean distance (straight line) option is selected from available genetic distance clustering models. linkage_type specifies Ward's method which selects the optimum value of an objective function.

Hierarchical clustering builds clusters incrementally to create a dendogram. The method does this by assigning each sample to a cluster and merging each of the clusters that are most similar.

```
# Call the cluster_analysis function
hclust_analysis <- cluster_analysis(sel.exp=ranked.exprs,
    cluster_type="HClust",
    distance="euclidean", linkage_type="ward.D2",
    gene_distance="correlation",
    num_clusters=3, data_name="GSE30219 Lung Cancer",
    probe_rank="SD_Rank", probe_num_selection="Poly_Probe_Num",
    cluster_num_selection="
    Fixed_Clust_Num")
```

```
## [1] "Your HClust Sample Dendrogram has been outputted"
## [1] "Your atr, gtr, and cdt files have been outputted for viewing in Java TreeView"
## [1] "A CSV file has been produced containing your sample and cluster assignment information"
```

The hierarchical clustering result are inspected using head(), the CSV file and the generated dendrogram and heatmap.

```
# Display the first few columns and rows of the object
head(hclust_analysis)
```

```
## GSM1465989 GSM1465990 GSM1465991 GSM1465992 GSM1465993 GSM1465994
##          3          3          1          2          2          2
```

The results show clustering <= 3 for each sample in line with generation of 3 clusters generated by hierachael clustering.

Dendrogram:

The dendrogram shows 3 distinct clusters from the GSE30219 lung cancer samples using hierarchical clustering. The clustering was performed using Euclidean and Ward.D2 linkage. The gene selection was performed using "Poly" for the top 300 genes.

Heatmap:

The heatmap indicates gene expression values in lung cancer vs control patients. TreeView (http://jtreeview.sourceforge.net/) is a Java application that is used to interact with the ATR, GTR and CDT files generated from the clustering.

TreeView interacts with the files generated from hierachichal clustering. The heat map is used to identify if gene expression variation corresponds with sample clusters.

5.3 Perform Kmeans Clustering of Genes/Probes and Samples

The k-means algorithm iterates between an assign (to the closest match) and update (to the mean value) step. k-means analysis assumes that the data is spherical.

```
# Call the cluster_analysis function
 kmeans_analysis <- cluster_analysis(sel.exp=ranked.exprs,
    cluster_type="Kmeans",
    distance=NULL, linkage_type=NULL,
```

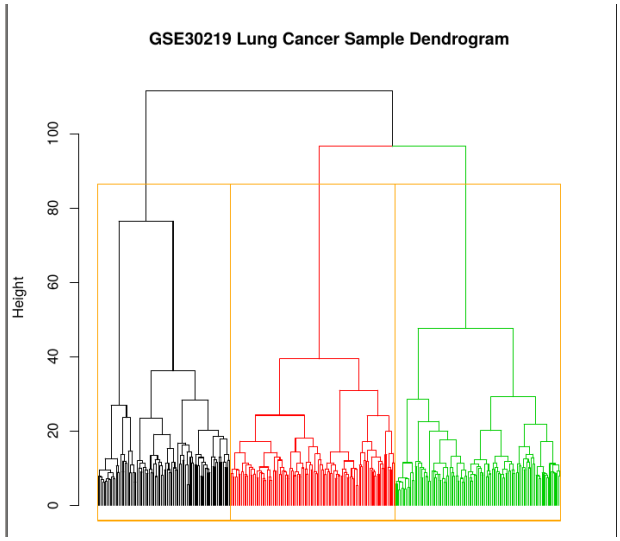


Figure 1: GSE30219 Dendrogram for lung cancer patients

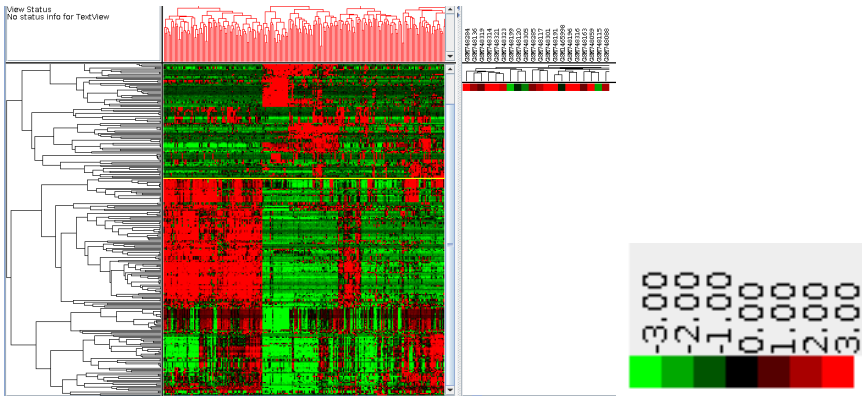


Figure 2: GSE30219 Heatmap indicating differential expression in lung cancer patients

```
    gene_distance=NULL, num_clusters=3,
    data_name="GSE30219 Lung", probe_rank="SD_Rank",
    probe_num_selection="Poly_Probe_Num",
    cluster_num_selection="Fixed_Clust_Num")
```

```
## [1] "A CSV file has been produced containing your sample and cluster assignment information"
```

The above generates a CSV file ranking genes by their gene expression.

```
 # Display the first few rows and columns of the object
 head(kmeans_analysis)
```

```
## GSM748053 GSM748054 GSM748055 GSM748056 GSM748057 GSM748058
##         1         1         1         2         1         1
```

The CSV file lists the sample name and how many clusters share common cluster grouping. The CSV files can be read into R to identify the differential expression and overlap between both methods.

6. Obtaining the Average Expression for Each Gene/Probe in Each Cluster

avg_probe_exp will generate the average expression of each gene for each cluster. The ranked.exprs using the poly probe ranking method is used to obtain gene expression levels. The samp_cluster used is specified to either HClust or kmeans depending on the clustering approach being analysed.

```
# Call the avg_probe_exp function
avg_matrix <- avg_probe_exp(sel.exp=ranked.exprs,
    samp_cluster=hclust_analysis,
    data_name="GSE30219 Lung Cancer", cluster_type="HClust", distance="euclidean",
    linkage_type="ward.D2", probe_rank="SD_Rank",
    probe_num_selection="Poly_Probe_Num",
    cluster_num_selection="Fixed_Clust_Num")
```

```
## [1] "Your matrix containing the average gene probe expression in each cluster is finished"
```

View the results using head()

```
# Display the first few rows and columns of the matrix
head(avg_matrix)
```

```
##             Cluster 1 Cluster 2 Cluster 3
## 209125_at    1.829711  8.989074  2.642929
## 209988_s_at  6.038510  1.574828  3.013194
## 223678_s_at  3.016441  5.415421  8.398579
## 218835_at    5.469980  7.957935 10.609592
## 201820_at    3.445349  9.684303  3.349310
## 213796_at    2.048491  6.919938  1.661265
```

Using the kmeans_analysis approach,

```
# Call the avg_probe_exp function
avg_matrix <- avg_probe_exp(sel.exp=ranked.exprs,
    samp_cluster=kmeans_analysis,
    data_name="GSE30219 Lung Cancer", cluster_type="Kmeans", distance=NULL,
    linkage_type=NULL, probe_rank="SD_Rank",
    probe_num_selection="Poly_Probe_Num",
    cluster_num_selection="Fixed_Clust_Num")
```

```
## [1] "Your matrix containing the average gene probe expression in each cluster is finished"
```

```
head(avg_matrix)
```

```
##             Cluster 1 Cluster 2 Cluster 3
## 209125_at    2.642711  8.711454  1.619808
## 209988_s_at  3.314865  1.561273  6.411187
## 223678_s_at  8.264785  5.220412  2.398227
## 218835_at   10.543429  7.785838  4.734300
## 201820_at    3.417439  9.511165  3.179800
## 213796_at    1.668850  6.759497  2.020406
```

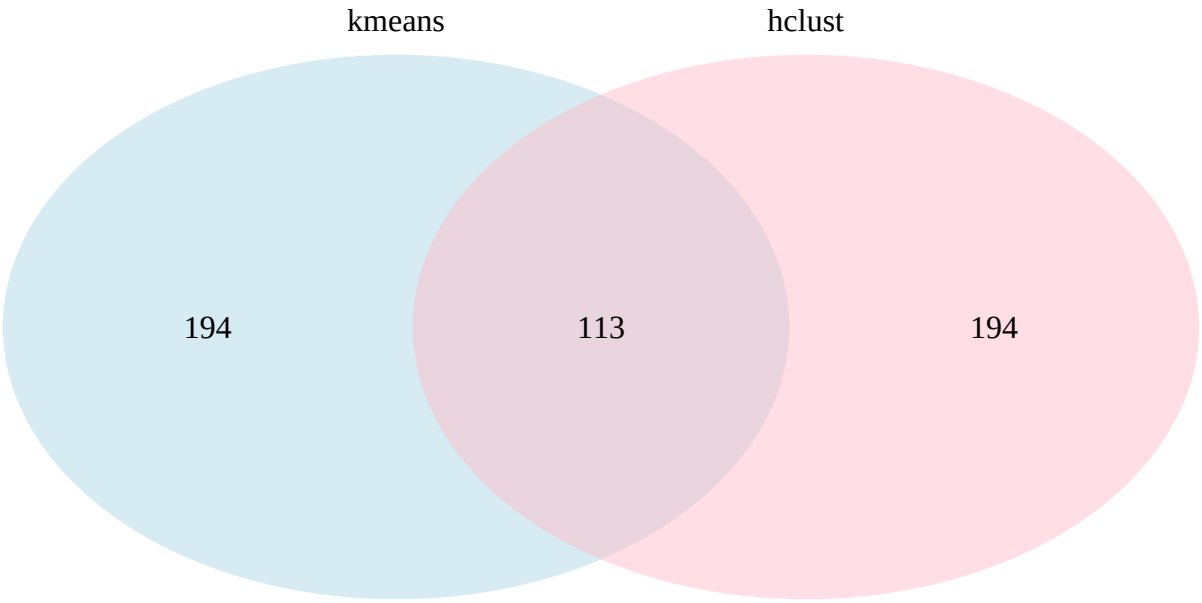
The results for the hierachical and kmeans clustering by generating a Venn diagram:

```
library(VennDiagram)
#To generate a Venn diagram
#read in each gene/probe list as CSV files
kmeans_csv <- read.csv('Kmeans.csv',header = FALSE)
head(kmeans_csv)
hclust_csv <- read.csv('HClust.csv', header = FALSE)
head(hclust_csv)

#Perform an element wise comparison between the 2 columns
kmeans_csv == hclust_csv

# Number of samples that are placed in same or different cluster.
same_cluster = length(which((kmeans_csv == hclust_csv)[,2]))
different_cluster = length(which((kmeans_csv != hclust_csv)[,2]))
total_samples_in_all_cluster = length(kmeans_csv[,2])
```

```
# Plot it in a venn.
grid.newpage()
draw.pairwise.venn(total_samples_in_all_cluster, total_samples_in_all_cluster, same_cluster, category =
```




```
## (polygon[GRID.polygon.11], polygon[GRID.polygon.12], polygon[GRID.polygon.13], polygon[GRID.polygon.
```

The Venn diagram indicates that here are only 113 overlapping genes between the two methods, indicating high variability in the resulting genes using each method.

7. Perform survival analysis using Kaplan-Meier survival plots.

The survival analysis is performed using the results from the hierarchical and kmeans methods in line with **Objective 7**. The surival analyis is performed using Kaplan-Meier plots.

HClust approach:

```
# Obtain clinical outcome file
clin_file <- system.file("extdata", "GSE30219-DFS-Clinical-Outcome.txt",
    package="multiClust")

# Call the avg_probe_exp function
surv <- surv_analysis(samp_cluster=hclust_analysis, clinical=clin_file,
    survival_type="DFS", data_name="GSE30219 Lung Cancer",
    cluster_type="HClust", distance="euclidean",
    linkage_type="ward.D2", probe_rank="SD_Rank",
    probe_num_selection="Poly_Probe_Num",
    cluster_num_selection="Fixed_Clust_Num")
```

```
## [1] "Your Kaplan Meier Survival Plot has been finished"
```

```
surv
```

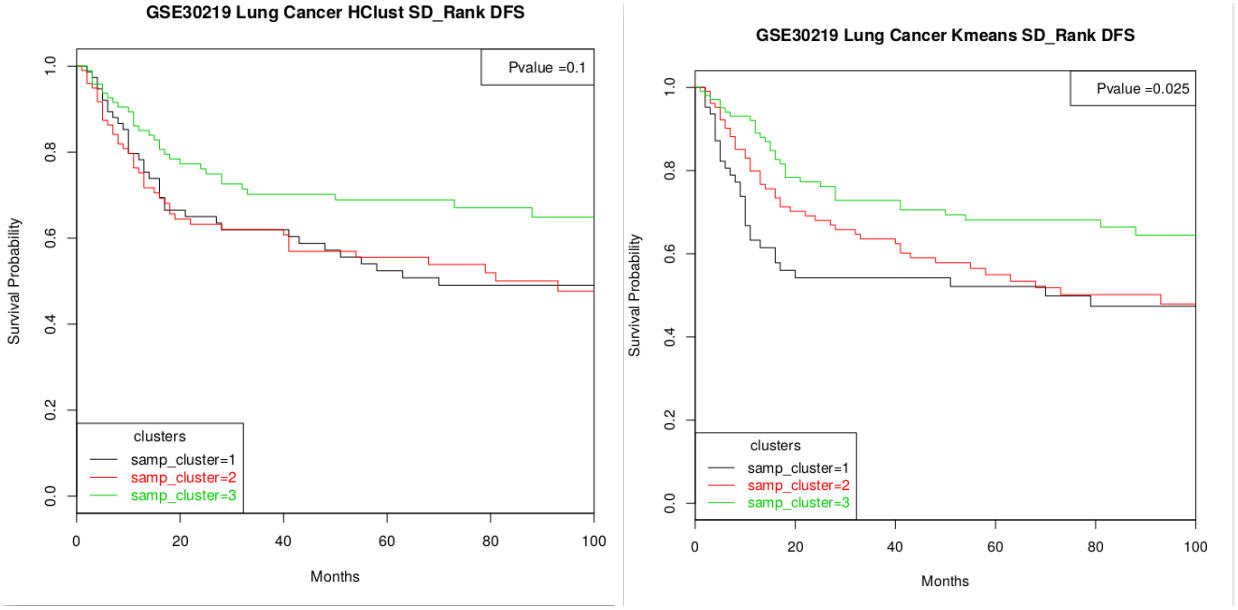


Figure 3: GSE30219 Disease free survival (DFS) for Lung cancer patients

```
##    pvalue
## 0.1024732
```

Kmeans approach:

```
# Call the avg_probe_exp function
surv <- surv_analysis(samp_cluster=kmeans_analysis, clinical=clin_file,
    survival_type="DFS", data_name="GSE30219 Lung Cancer",
    cluster_type="Kmeans", distance=NULL,
    linkage_type=NULL, probe_rank="SD_Rank",
    probe_num_selection="Poly_Probe_Num",
    cluster_num_selection="Fixed_Clust_Num")
```

```
## [1] "Your Kaplan Meier Survival Plot has been finished"
```

```
surv
```

```
##     pvalue
## 0.02501829
```

This above commands generate the following survival plots:

The Kaplan Meier plot portays patient survival time probability over time. The clinical outcome of DFS is measured. The survival analysis using survival package on R specifies whether there is significant correlation between a subgroup and clinical outcome.

Conclusion:

The multiClust apporach uses gene ranking, gene selection and clustering to generate clinically related groups from gene expression data. The GSE30219 contains lung cancer samples that were clustered using the number of "poor prognosis" genes per sample transcriptome and resulted in 3 distinct clusters.

The use of SD ranking was selected as it was reported as most effect in combination with kmeans clustering 5. This kmeans approach resulted in significant ($p = 0.025$) association of cluster 3 with prolonged DFS. The

hierarchical approach used Euclidean distance also however results were insignificant (p = 0.1). Hierarchical clustering provided a useful dendrogram and heatmap to identify neighboring clusters and co-expressed genes associated with poor prognosis. Gene expression analysis indicated genes expression associated with each cluster eg. 218835_at reports surfactant protein A2 (SFTPA2) which is reported to have significantly lower expression in NSCLC tissue (PMID:25514367) and in this study has higher differential in cluster 1 for kmeans analysis. The survival analysis indicate that the cluster 3 demonstrates significant influence on patient outcome and is the subset containing no expression of poor prognosis genes. multiClust proves a reliable approach for cluster analysis given the statistically signficant subtype identification. This subtype classification can be applied to tailor treatment and improve clinical outcome.

References

1. Edgar R, Domrachev M, Lash AE. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Res. 2002 Jan 1;30(1):207-10

2. Rousseaux S, Debernardi A, Jacquiau B, et al. Ectopic activation of germline and placental genes identifies aggressive metastasis-prone lung cancers. Sci Transl Med. 2013;5(186):186ra66. doi:10.1126/scitranslmed.3005723

3. Inamura K. Lung Cancer: Understanding Its Molecular Pathology and the 2015 WHO Classification. Front Oncol. 2017;7:193. Published 2017 Aug 28. doi:10.3389/fonc.2017.00193

4. Chou CC, Chen CH, Lee TT, Peck K. Optimization of probe length and the number of probes per gene for optimal microarray analysis of gene expression. Nucleic Acids Res. 2004;32(12):e99. Published 2004 Jul 8. doi:10.1093/nar/gnh099

5. Lawlor N, Fabbri A, Guan P, George J, Karuturi RK. multiClust: An R-package for Identifying Biologically Relevant Clusters in Cancer Transcriptome Profiles. Cancer Inform. 2016;15:103–114. Published 2016 Jun 12. doi:10.4137/CIN.S38000