Example application of Singscore on sample classification

Ruqian

2018-06-17

Introduction

In this document, I explain the steps I used to derive the gene signatures for the three iClusters of 183 liver cancer patient samples clustered by (Ally et al., n.d.). They used datasets across five different platforms in forming the iClusters, while I only look at the mRNA expression dataset in forming the iCluster gene signatures. Using scores from `singscore` method, I trained a multinomial classification model that can be used for predicting iCluster labels for new samples.

In the (Ally et al., n.d.) study, they classified new sample cohort dataset with the classfier they built and produced survival plot for the different iCluster groups in the new samples cohort (refer to Figure 3 at (Ally et al., n.d.). They identified samples in iCluster1 had lowest survival rate.

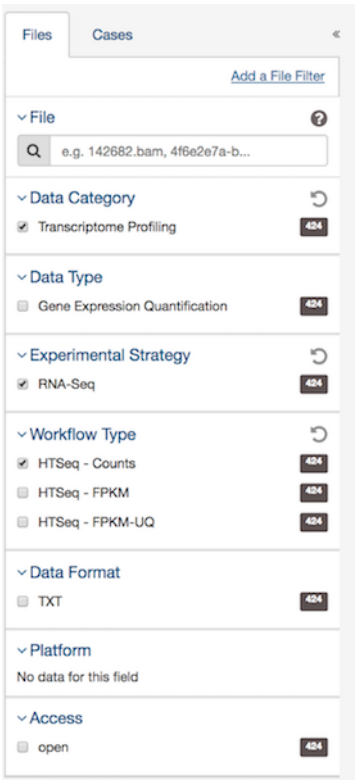
I built an iCluster classifier using only the mRNA expression dataset and `singscore`. I scored the 183 liver cancer samples against the iCluster gene signatures using `singscore` and obtained three scores for each sample. I then used the obtained scores to train a classfier for new sample prediction.

I scored the (FUDAN) dataset (Roessler et al. 2010) GSE14520 with the three gene signatures and predicted the samples' cluster labels with the classification model trained using scores of the formed iCluster samples. I also did the survival analysis. The result showed that iCluster1 had lowest survival rate, which was concordance with the article (Ally et al., n.d.).

Download count-data

We first download the dataset and then load the data in R. After that, we used some annotation files in identifying the samples' iCluster labels for Differential gene expression analysis.

The raw count data were downloaded from the Genomic Data Commons Data Portal https://portal.gdc.cancer.gov/ from project TCGA-LIHC. 424 samples were downloaded using the follow-



ing filter rules.

The study by (Ally et al., n.d.) had clustered the liver cancer patient samples into three groups.The sample IDs of patients within each cluster can be found in the TableS1(Ally et al., n.d.).

Load the raw read count data

Before we load the raw read count data, we need to find the samples that we are interested in, which are the clustered samples from our downloaded files.

Now, we read in the TableS1 for samples' iCluster labels.

```
# load the libraries we need for the analysis

library(readxl)
library(edgeR)
```

```
## Warning: package 'edgeR' was built under R version 3.4.3
```

```
## Warning: package 'limma' was built under R version 3.4.3
```

```
library(limma)
library(singscore)
library(survival)
library(RColorBrewer)
library(ggplot2)


mmc1 <- read_excel("~/Documents/davisLab/mmc1.xlsx", skip = 3)
# have a look at some columns of the data
head(mmc1[,c(1,2,22,84)])
```

```
## # A tibble: 6 x 4
```

```
##   UUID              Barcode     `iCluster clusters (k=3~ `HCC subtypes`
##   <chr>             <chr>       <chr>                    <chr>
## 1 a06fe860-8d5f-4e~ TCGA-ES-A2~ iCluster:2               NA
## 2 ef575b18-0cd6-40~ TCGA-DD-A3~ NA                       No specific subt~
## 3 d4fcbd8f-0b83-4b~ TCGA-DD-A7~ iCluster:3               No specific subt~
## 4 1a0498cb-07d9-41~ TCGA-DD-A1~ iCluster:1               Fibrolamellar ca~
## 5 2c1ac52f-293b-4c~ TCGA-G3-A3~ iCluster:1               Steatohepatitic
## 6 ae09a90d-e285-44~ TCGA-DD-A4~ iCluster:1               No specific subt~
```

We analyse samples that have been clustered, so we remoe the "NA" rows.

```
identified_clusters <- mmc1[!mmc1$`iCluster clusters (k=3, Ronglai Shen)`=="NA",]

# So we will be looking at 183 samples
dim(identified_clusters)
```

```
## [1] 183  98
```

Then we construct the list of files of samples we are intested in.

```
# get the sample IDs of 183 samples
clustered_samples <- identified_clusters$Barcode

# From the sample sheet downloaded from GDC, we get the file names mapped to the
# SampleID
gdc_sample_sheet.2018.03.28 <-
  read.delim("~/Documents/davisLab/HCC/gdc_sample_sheet.2018-03-28.tsv",
             quote="")

all_files <-
  gdc_sample_sheet.2018.03.28[gdc_sample_sheet.2018.03.28$Sample.ID
                              %in% clustered_samples,]

dim(all_files)
```

```
## [1] 183   8
```

Load the raw count data from `all_files`. I used `edgeR`'s function for loading all samples' read count data at once.

```
files_path <-
  paste0("/Users/ruqianlyu/Documents/davisLab/HCC/gdc_download_20180328_001539/",
                    all_files$File.ID, "/", all_files$File.Name)

# Use the readDGE function for edgeR for reading all samples' count files.
initial_dge_object <- edgeR::readDGE(files_path)
```

```
## Meta tags detected: __no_feature, __ambiguous, __too_low_aQual, __not_aligned, __alignment_not_unique
```

```
# Here is the dimension of the raw gene expression dataset
dim(initial_dge_object)
```

```
## [1] 60487   183
```

```
# Use shorter file names
files_names <- substring(colnames(initial_dge_object), 107,
                         nchar(colnames(initial_dge_object)))

# head(files_names)
```

```r
# rename the column
colnames(initial_dge_object) <- files_names
# rename the rows
rownames(initial_dge_object) = sub("\\.\\d+", "", rownames(initial_dge_object))
# Map the file_names to Sample.ID so that we can identify the cluster of the
# counts
y <- as.data.frame(colnames(initial_dge_object))
colnames(y) <- "File.Name"

k <- as.data.frame(paste0(y$File.Name,".gz"))

colnames(k) <- "File.Name"

head(k)
```

```
##                                             File.Name
## 1 0fc6f38a-62da-4c2f-8a72-5c34b77656e5.htseq.counts.gz
## 2 f32c1def-c5c6-4076-966e-ae5f7233060a.htseq.counts.gz
## 3 687e7d1d-99eb-4bf9-9fa3-49e324ef32c3.htseq.counts.gz
## 4 f6ae6ac1-3e00-4021-a6e7-fbb0d5f12836.htseq.counts.gz
## 5 5fe28ffa-63af-4a8d-8512-b0742b4cded4.htseq.counts.gz
## 6 554f6de3-63c7-47b1-a75a-dcfc73f54e96.htseq.counts.gz
```

```r
# Using the sample sheet document to align the file name with Sample.ID
f <- merge(k, gdc_sample_sheet.2018.03.28, by = "File.Name", sort = FALSE)


# Use the mmc1 file to annotate the Samples with cluster labels
g <- merge(f, mmc1, by.x = "Sample.ID", by.y = "Barcode", sort = FALSE)

head(g[,c(1:2,29)])
```

```
##          Sample.ID                                             File.Name
## 1 TCGA-BD-A3EP-01A 0fc6f38a-62da-4c2f-8a72-5c34b77656e5.htseq.counts.gz
## 2 TCGA-DD-A4NO-01A f32c1def-c5c6-4076-966e-ae5f7233060a.htseq.counts.gz
## 3 TCGA-G3-A3CK-01A 687e7d1d-99eb-4bf9-9fa3-49e324ef32c3.htseq.counts.gz
## 4 TCGA-DD-A4NS-01A f6ae6ac1-3e00-4021-a6e7-fbb0d5f12836.htseq.counts.gz
## 5 TCGA-CC-A7II-01A 5fe28ffa-63af-4a8d-8512-b0742b4cded4.htseq.counts.gz
## 6 TCGA-PD-A5DF-01A 554f6de3-63c7-47b1-a75a-dcfc73f54e96.htseq.counts.gz
##   iCluster clusters (k=3, Ronglai Shen)
## 1                            iCluster:2
## 2                            iCluster:2
## 3                            iCluster:2
## 4                            iCluster:1
## 5                            iCluster:1
## 6                            iCluster:3
```

```r
# The factors that might be of interest
race <- as.factor(g$race)
gender <- as.factor(g$gender)

# group stores the cluster label for each sample
group <- as.factor(g$`iCluster clusters (k=3, Ronglai Shen)`)

#remove the ":" from cluster label
```

```
temp <- sapply(group, function(x){sub(":","",x)})
group <- temp

initial_dge_object$samples$group <- group
initial_dge_object$samples$race <- race
initial_dge_object$samples$gender <- gender

# Here we have the initial_dge_object which contains the count matrxi and sample
# information for 183 samples and 60487 ESEMBL transcript IDs
dim(initial_dge_object)
```

```
## [1] 60487   183
```

Check gene types with bioMart

bioMart is an R pacakge that provides the interface with BioMart databases like Ensembl, COSMIC, Uniprot, HGNC, Gramene, Wormbase and dbSNP. Thus I used bioMart for annotating and filtering the genes in our dataset. I only kept the protein coding genes for the following analysis.

```
library(biomaRt)
ensembl=useMart("ensembl")
#listDatasets(ensembl)
ensembl = useDataset("hsapiens_gene_ensembl",mart=ensembl)
#listFilters(ensembl)
attrs = listAttributes(ensembl)

# mapToType = getBM(
#   attributes = c('transcript_biotype', 'entrezgene','ensembl_gene_id',
# 'ensembl_gene_id_version','hgnc_symbol'),
#   filters = 'entrezgene',
#   values = genes$ENTREZID,
#   mart = ensembl
# )
# save(mapToType,file = "mapToType.RData")
```

Load the saved annotation in mapToType.RData file

```
load(file = "./mapToType.RData")

head(mapToType)
```

```
##     transcript_biotype entrezgene ensembl_gene_id ensembl_gene_id_version
## 1 processed_transcript      10046 ENSG00000013619      ENSG00000013619.13
## 2       protein_coding      10046 ENSG00000013619      ENSG00000013619.13
## 3      retained_intron      10046 ENSG00000013619      ENSG00000013619.13
## 4       protein_coding      10048 ENSG00000010017      ENSG00000010017.12
## 5 processed_transcript      10048 ENSG00000010017      ENSG00000010017.12
## 6       protein_coding      10061 ENSG00000033050       ENSG00000033050.8
##   hgnc_symbol
## 1      MAMLD1
## 2      MAMLD1
## 3      MAMLD1
## 4      RANBP9
## 5      RANBP9
```

```
## 6       ABCF2
```

```
# Find the mappings for transcripts in our intial dge object
mappedTrans = mapToType[mapToType$ensembl_gene_id
                        %in% rownames(initial_dge_object$counts),]

# Find the proten coding genes
mappedTrans = mappedTrans[mappedTrans$transcript_biotype == "protein_coding",]
# remove genes without symbol mapped
mappedTrans = mappedTrans[!mappedTrans$hgnc_symbol == "",]


dim(mappedTrans)
```

```
## [1] 19235     5
```

```
length(unique(mappedTrans$ensembl_gene_id))
```

```
## [1] 19067
```

```
# remove the duplicated mapping by ensembl_gene_id

dup <- mappedTrans$ensembl_gene_id[duplicated(mappedTrans$ensembl_gene_id)]
length(dup)
```

```
## [1] 168
```

```
# The first 5 duplicated ones are
# mappedTrans[mappedTrans$hgnc_symbol %in% dup,][1:5,]

# takes care of the duplication by keeping the first occurrence of each gene ID
# match returns a vector of the positions of (first) matches of its first argument in its
# second
mat <- match(unique(mappedTrans$ensembl_gene_id), mappedTrans$ensembl_gene_id)
mappedTrans <- mappedTrans[mat,]

mat <- match(unique(mappedTrans$hgnc_symbol), mappedTrans$hgnc_symbol)
mappedTrans <- mappedTrans[mat,]

dim(mappedTrans)
```

```
## [1] 19066     5
```

```
# Use the `ENSEMBL` column remained in the `genes` to subset the initial dge object
dge_object <- initial_dge_object[mappedTrans$ensembl_gene_id,]
dge_object$genes <- mappedTrans
rownames(dge_object) = mappedTrans$hgnc_symbol
# Cleaned dge_object
dim(dge_object)
```

```
## [1] 19066   183
```

Data pre-processing for deriving DEGs for three iClusters

Normalisation and filtering

```
# raw counts are converted to CPM and log-CPM values using the CPM function provided by edgeR

cpmVal <- cpm(dge_object)
lcpm <- cpm(dge_object, log=TRUE)

# removing lowly expressed genes, 0 across all samples
table(rowSums(dge_object$counts==0)==183)
```

```
##
## FALSE  TRUE
## 18871   195
```

```
# keep genes expressed in at least 55 samples, 55 is the number of samples in the smallest group
keep.exprs <- rowSums(cpmVal>1)>=55

sum(keep.exprs)
```

```
## [1] 12988
```

```
dge_object <- dge_object[keep.exprs,, keep.lib.sizes=FALSE]
dim(dge_object)
```

```
## [1] 12988   183
```

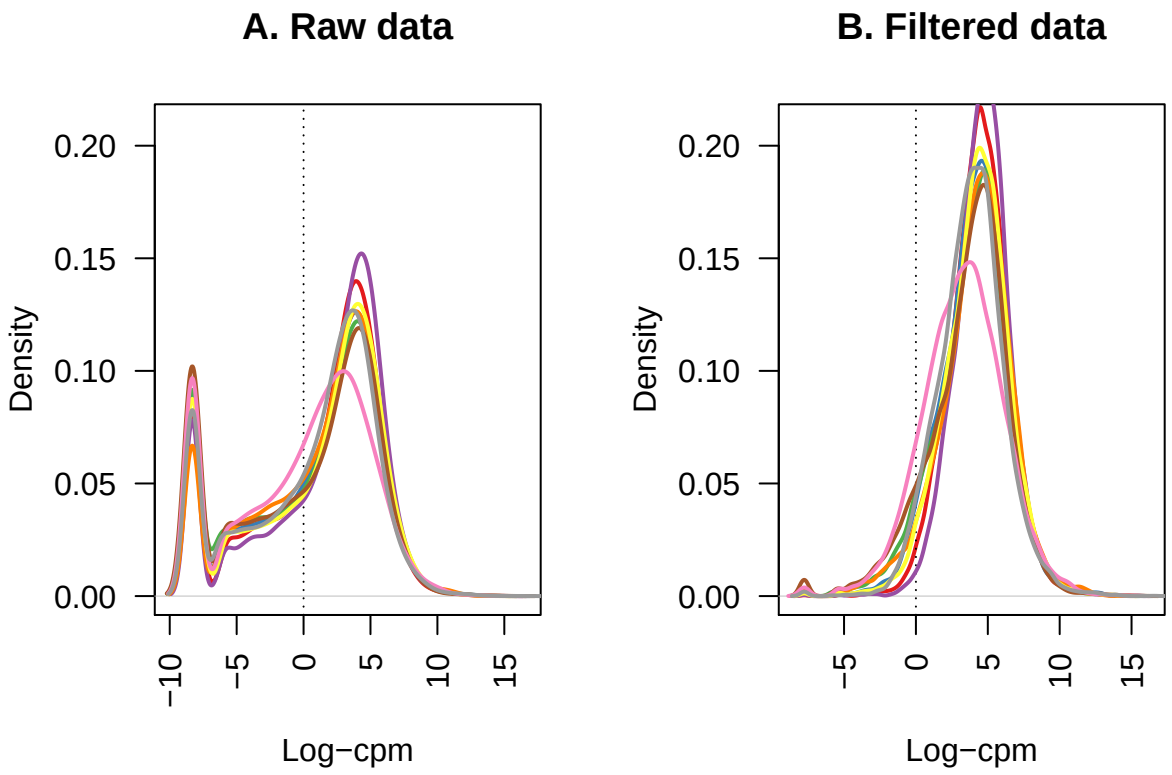
Plot comparison of before and after filtering

```
nsamples <- ncol(dge_object)
col <- brewer.pal(nsamples, name = "Set1")
```

```
## Warning in brewer.pal(nsamples, name = "Set1"): n too large, allowed maximum for palette Set1 is 9
## Returning the palette you asked for with that many colors
```

```
par(mfrow=c(1,2))
# Before filtering
plot(density(lcpm[,1]), col=col[1], lwd=2, ylim=c(0,0.21),
     las=2, main="", xlab="")
title(main="A. Raw data", xlab="Log-cpm")
abline(v=0, lty=3)
for (i in 2:nsamples){
 den <- density(lcpm[,i])
 lines(den$x, den$y, col=col[i], lwd=2)
}

# legend("topright", samplenames, text.col=col, bty="n")
# After filtering
lcpm <- cpm(dge_object, log=TRUE)
plot(density(lcpm[,1]), col=col[1], lwd=2, ylim=c(0,0.21), las=2,
main="", xlab="")
title(main="B. Filtered data", xlab="Log-cpm")
abline(v=0, lty=3)
for (i in 2:nsamples){
   den <- density(lcpm[,i])
   lines(den$x, den$y, col=col[i], lwd=2)
```

```
}
```




```
# legend("topright", samplenames, text.col=col, bty="n")
```

TMM normalisation

Normalisation by the method of trimmed mean of M-values (TMM)

```
dge_object <- calcNormFactors(dge_object, method = "TMM")
head(dge_object$samples$norm.factors)
```

```
## [1] 1.1894254 0.9491540 0.9250655 1.3645886 1.2291164 1.1252687
```

Differential gene expression analysis

After the gene expression dataset has been filtered and cleand, we can derive DEGs with functions from `limma`. I have referred to the 'RNASeq-1-2-3' (Law et al. 2016) article for carrying out the DGE analysis for the iCluster samples.

```
# Make the design matrix by group
design <- model.matrix(~0+group)
colnames(design) <- gsub("group", "", colnames(design))

# Make the contrast matrix in the way of comparing each one of the cluster
# with the rest.
contr.matrix2 <- makeContrasts(
   c1vsc2_3 = iCluster1-(iCluster2+iCluster3)/2,
   c2vsc1_3 = iCluster2-(iCluster1+iCluster3)/2,
   c3vsc1_2 = iCluster3-(iCluster1+iCluster2)/2,
```

```
   levels = colnames(design))

# Use the voom function to tranform our RNA-seq data for linear modelling
v <- voom(dge_object, design, plot=FALSE)

# Fit liear model to our transformed data
vfit <- lmFit(v, design)
vfit <- contrasts.fit(vfit, contrasts=contr.matrix2)
# empirical Bayes moderation is carried out by the eBayes function
efit <- eBayes(vfit)

summary(decideTests(efit))
```

```
##        c1vsc2_3 c2vsc1_3 c3vsc1_2
## Down       3118      999     1708
## NotSig     6961    10548     9653
## Up         2909     1441     1627
```

```
tfit <- treat(efit, lfc=log2(1.27))
dt <- decideTests(tfit)
summary(dt)
```

```
##        c1vsc2_3 c2vsc1_3 c3vsc1_2
## Down        860       14      374
## NotSig    10953    12867    12542
## Up         1175      107       72
```

```
new.c1.vs.c23 <- topTreat(tfit, coef=1, n=Inf, p.value = 0.01)
new.c2.vs.c13 <- topTreat(tfit, coef=2, n=Inf, p.value = 0.01)
new.c3.vs.c12 <- topTreat(tfit, coef=3, n=Inf, p.value = 0.01)

dim(new.c1.vs.c23)
```

```
## [1] 1571   10
```

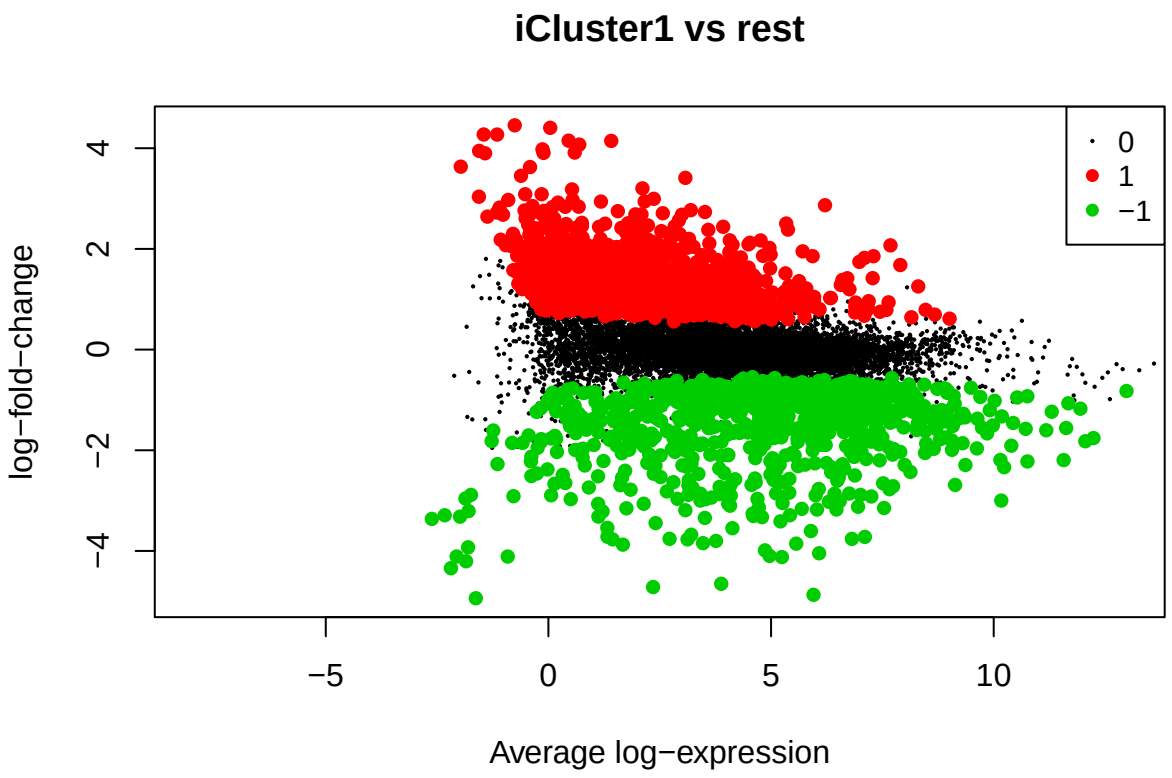
```
dim(new.c2.vs.c13)
```

```
## [1] 61 10
```

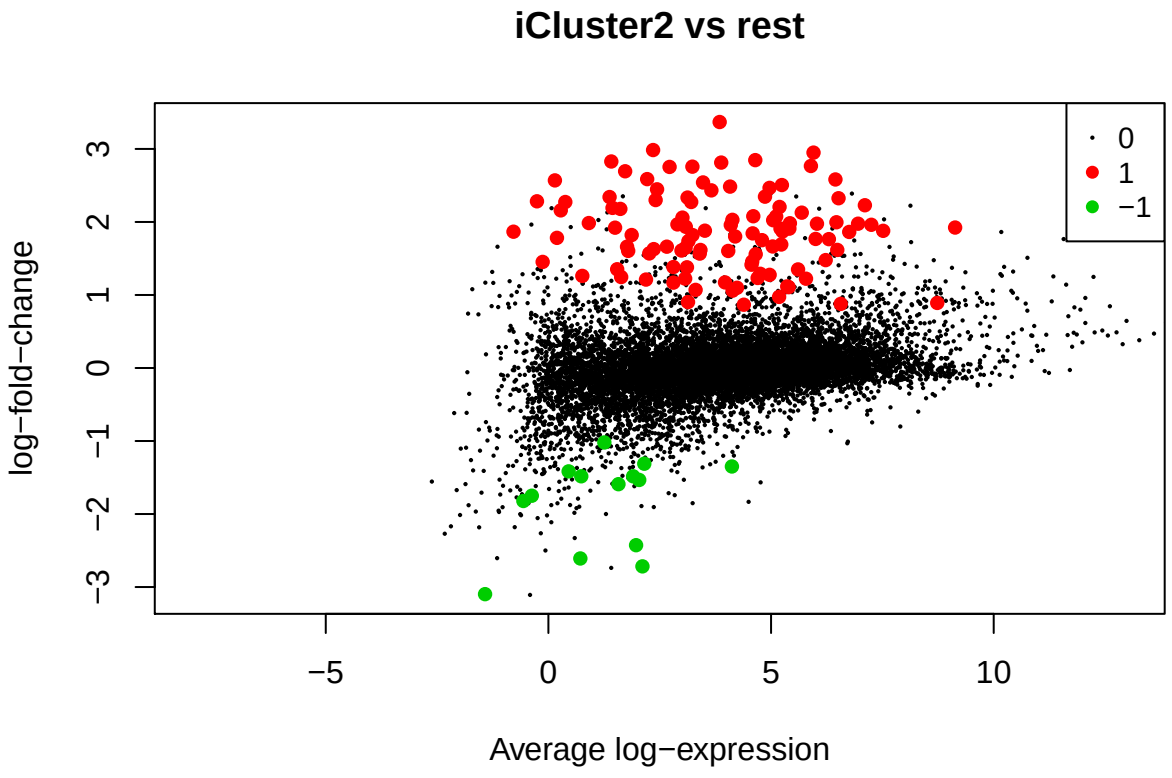
```
dim(new.c3.vs.c12)
```

```
## [1] 221  10
```

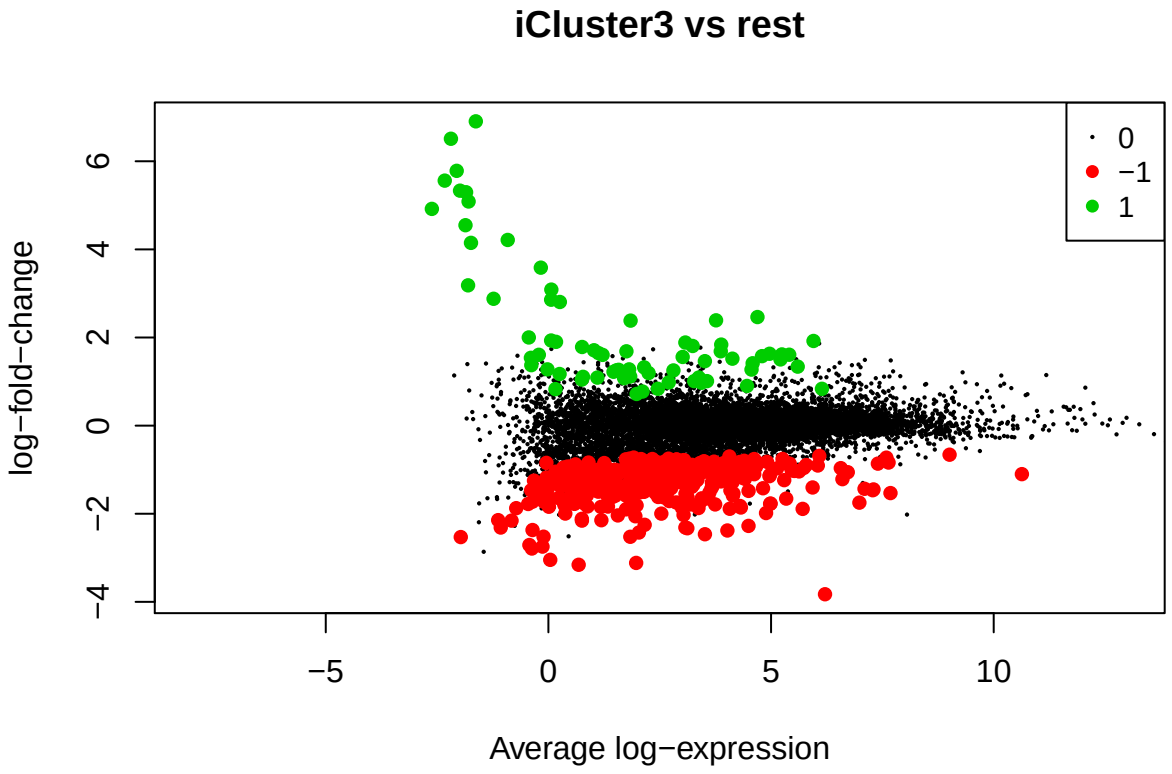
```
# Generate the MD plot
plotMD(tfit, column=1, status=dt[,1], main='iCluster1 vs rest',xlim=c(-8,13))
```



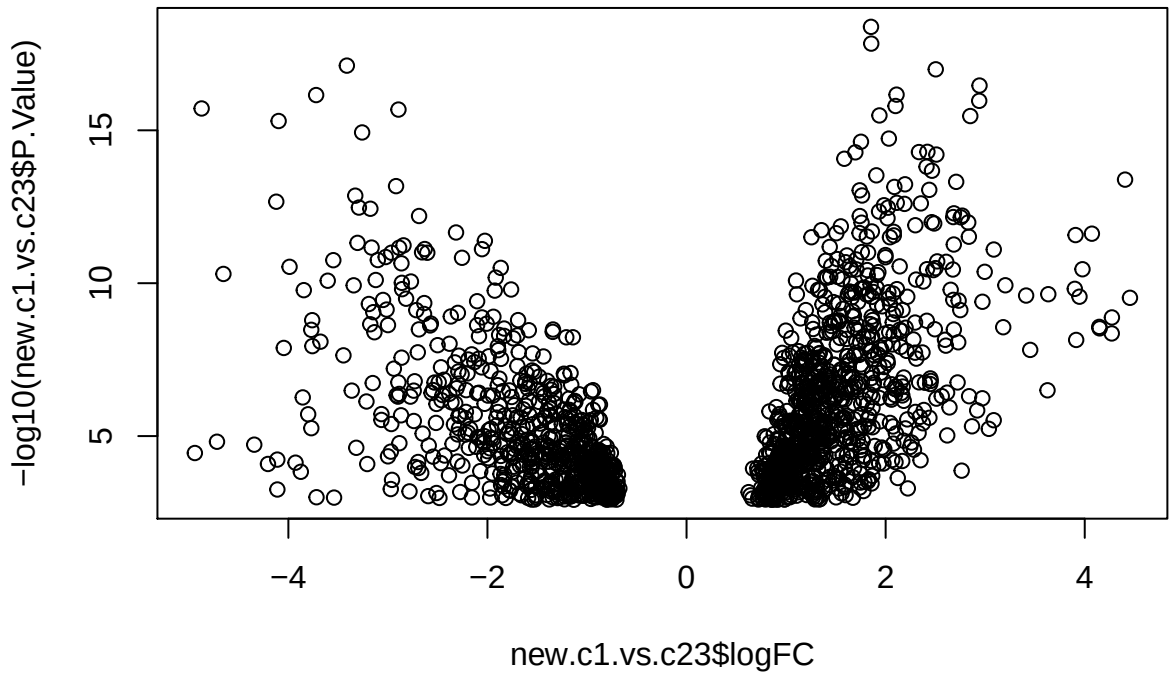

```
plotMD(tfit, column=2, status=dt[,2], main='iCluster2 vs rest',xlim=c(-8,13))
```




```
plotMD(tfit, column=3, status=dt[,3], main='iCluster3 vs rest',xlim=c(-8,13))
```




```
# From the DEGs for c1.vs.c23, we select the more signicant and with large logFC
plot(new.c1.vs.c23$logFC, -log10(new.c1.vs.c23$P.Value))
```




```
new.c1.vs.c23 <- new.c1.vs.c23[-log10(new.c1.vs.c23$P.Value)> 7
                               & (abs(new.c1.vs.c23$logFC)>1.55),]
dim(new.c1.vs.c23)
```

```
## [1] 294  10
```

```
#write the final iCluster gene signatures to files

write.table(new.c1.vs.c23, file = "new.c1.vs.c23.txt")
write.table(new.c2.vs.c13, file = "new.c2.vs.c13.txt")
write.table(new.c3.vs.c12, file = "new.c3.vs.c12.txt")
```

Obtain the up-regulated gene sets and down-regulated gene sets for each gene signauture

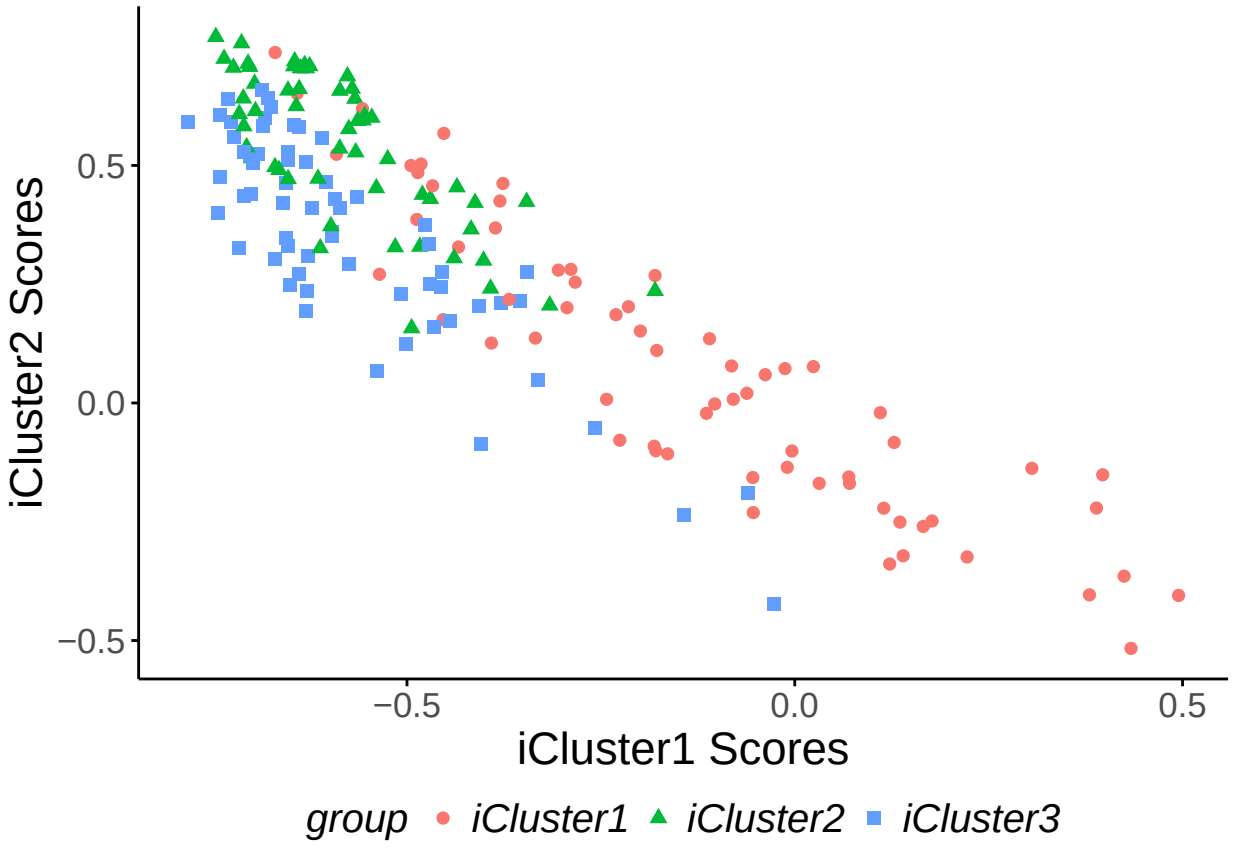
```
up_ic1 <- new.c1.vs.c23[new.c1.vs.c23$logFC>0,]
dn_ic1 <- new.c1.vs.c23[new.c1.vs.c23$logFC<0,]

up_ic2 <- new.c2.vs.c13[new.c2.vs.c13$logFC>0,]
dn_ic2 <- new.c2.vs.c13[new.c2.vs.c13$logFC<0,]

up_ic3 <- new.c3.vs.c12[new.c3.vs.c12$logFC>0,]
dn_ic3 <- new.c3.vs.c12[new.c3.vs.c12$logFC<0,]

up_ic1_gs <- up_ic1$hgnc_symbol
dn_ic1_gs <- dn_ic1$hgnc_symbol

up_ic2_gs <- up_ic2$hgnc_symbol
dn_ic2_gs <- dn_ic2$hgnc_symbol

up_ic3_gs <- up_ic3$hgnc_symbol
dn_ic3_gs <- dn_ic3$hgnc_symbol
```

Score the iCluster dataset with `singscore`

The 183 liver cancer samples with known iClusters labels were scored against the three gene signatures obtained above. The scoring results were used to train a multinomial classification model.

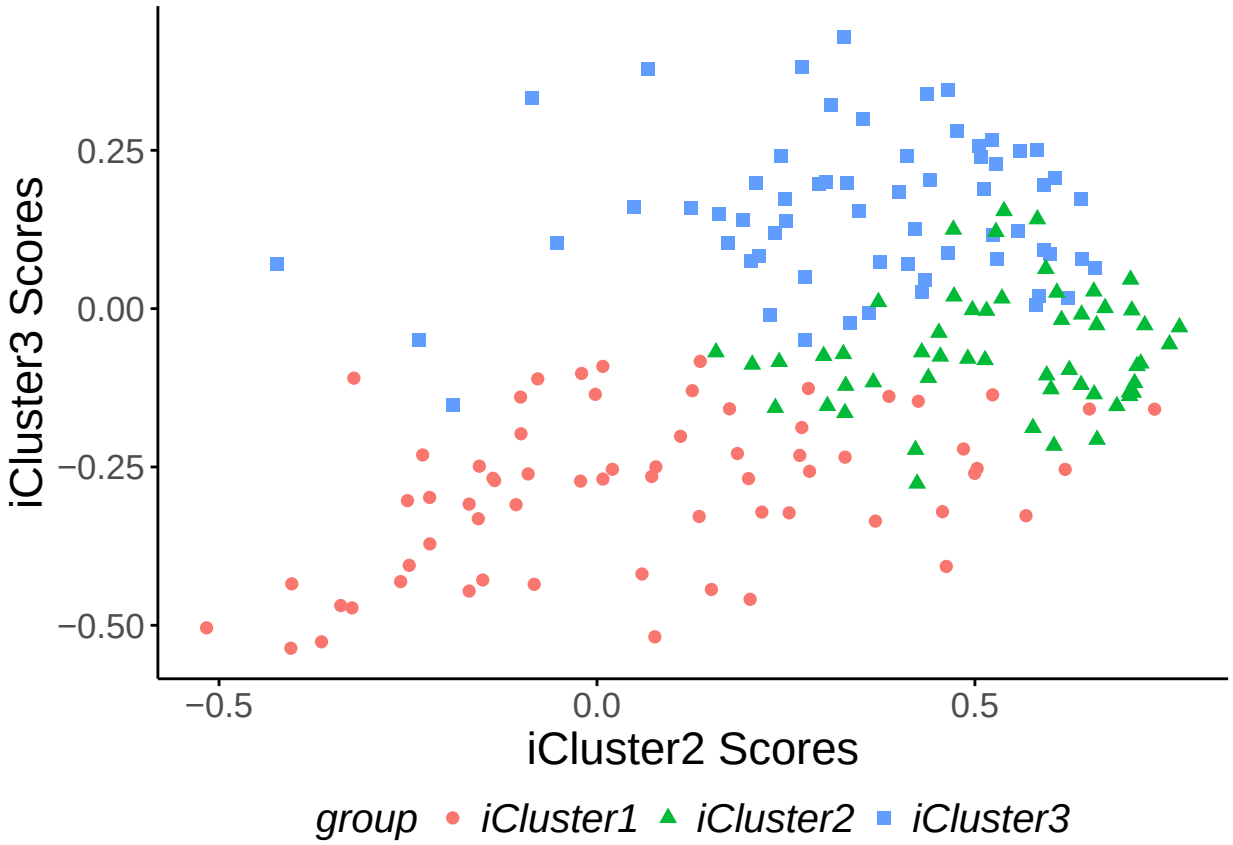
```
rankedData = rankGenes(dge_object$counts)
#rownames(rankedData) = dge_object$genes$hgnc_symbol

#head(rankedData)
# Score the data set using three gene signatures respectively
scoredf1 = simpleScore(rankedData, upSet = up_ic1_gs, downSet = dn_ic1_gs,
                       centerScore = TRUE)
scoredf2 = simpleScore(rankedData, upSet = up_ic2_gs, downSet = dn_ic2_gs,
                       centerScore = TRUE)
scoredf3 = simpleScore(rankedData, upSet = up_ic3_gs, downSet = dn_ic3_gs,
                       centerScore = TRUE)

cluster1Samples = g$File.Name[g$`iCluster clusters (k=3, Ronglai Shen)`
                              == "iCluster:1"]
cluster2Samples = g$File.Name[g$`iCluster clusters (k=3, Ronglai Shen)`
                              == "iCluster:2"]
cluster3Samples = g$File.Name[g$`iCluster clusters (k=3, Ronglai Shen)`
                              == "iCluster:3"]
```

```
scoredf1_pdf = data.frame(files = rownames(scoredf1),scoredf1$TotalScore,
                          group = dge_object$samples$group)

scoredf2_pdf = data.frame(files = rownames(scoredf2),scoredf2$TotalScore,
                          group = dge_object$samples$group)


scoredf3_pdf = data.frame(files = rownames(scoredf3),scoredf3$TotalScore,
                          group = dge_object$samples$group)

all = merge(merge(scoredf1_pdf, scoredf2_pdf, by.x = "files", by.y = "files"),
            scoredf3_pdf, by = "files")
rownames(all) = all$files
all = all[,c(2,4,6,7)]
textSize = 1.5


# The 2D landscape scatter plot for all samples


ggplot(data = all)+
  geom_point(aes(x = `scoredf1.TotalScore`,y=`scoredf2.TotalScore`,
                 colour = group, shape = group),size = 2)+
  xlab("iCluster1 Scores")+
  ylab("iCluster2 Scores")+
  theme_minimal() +
    theme(
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      axis.title = element_text(size = rel(textSize)),
      axis.text.x = element_text(angle = 0, size = rel(textSize)),
      axis.text.y = element_text(angle = 0, size = rel(textSize)),
      strip.background = element_rect(colour = "#f0f0f0",
                                      fill = "#f0f0f0"),
      strip.text = element_text(size = rel(textSize)),
      axis.line = element_line(colour = "black"),
      axis.ticks = element_line(),
      legend.position = "bottom",
      legend.direction = "horizontal",
      legend.margin = margin(unit(0, "cm")),
      legend.text = element_text(face = "italic", size = 15),
      legend.title = element_text(face = "italic", size = 15),
      plot.title = element_text(
        face = "bold",
        size = rel(textSize),
        hjust = 0.5
      )
    )
```



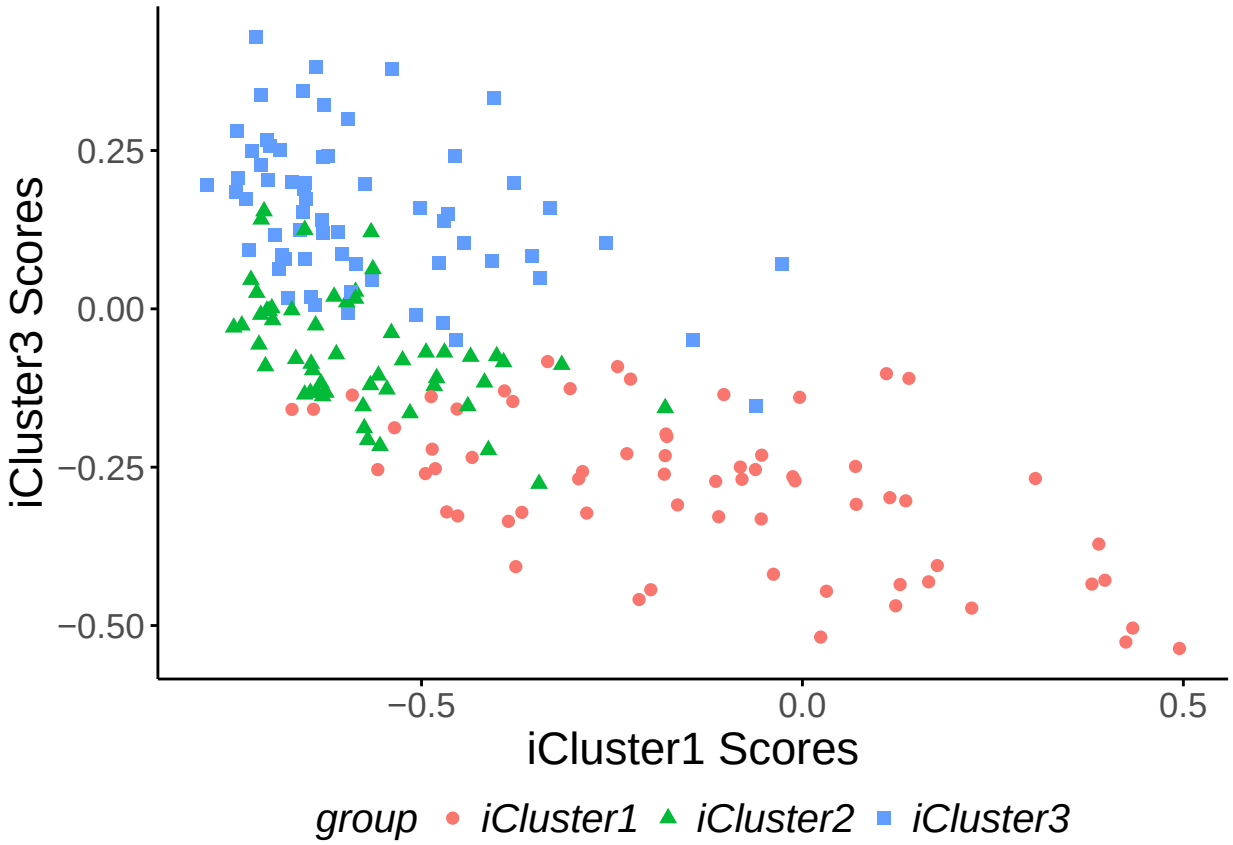

```
ggplot(data = all)+
  geom_point(aes(x = `scoredf2.TotalScore`,y=`scoredf3.TotalScore`,
                 colour = group, shape = group),size = 2)+
  xlab("iCluster2 Scores")+
  ylab("iCluster3 Scores")+
  theme_minimal() +
    theme(
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      axis.title = element_text(size = rel(textSize)),
      axis.text.x = element_text(angle = 0, size = rel(textSize)),
      axis.text.y = element_text(angle = 0, size = rel(textSize)),
      strip.background = element_rect(colour = "#f0f0f0",
                                      fill = "#f0f0f0"),
      strip.text = element_text(size = rel(textSize)),
      axis.line = element_line(colour = "black"),
      axis.ticks = element_line(),
      legend.position = "bottom",
      legend.direction = "horizontal",
      legend.margin = margin(unit(0, "cm")),
      legend.text = element_text(face = "italic", size = 15),
      legend.title = element_text(face = "italic", size = 15),
      plot.title = element_text(
        face = "bold",
        size = rel(textSize),
        hjust = 0.5
      )
```

```
  )
```




```
ggplot(data = all)+
  geom_point(aes(x = `scoredf1.TotalScore`,y=`scoredf3.TotalScore`,
                 colour = group, shape = group),size = 2)+
  xlab("iCluster1 Scores")+
  ylab("iCluster3 Scores")+
  theme_minimal() +
    theme(
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      axis.title = element_text(size = rel(textSize)),
      axis.text.x = element_text(angle = 0, size = rel(textSize)),
      axis.text.y = element_text(angle = 0, size = rel(textSize)),
      strip.background = element_rect(colour = "#f0f0f0",
                                      fill = "#f0f0f0"),
      strip.text = element_text(size = rel(textSize)),
      axis.line = element_line(colour = "black"),
      axis.ticks = element_line(),
      legend.position = "bottom",
      legend.direction = "horizontal",
      legend.margin = margin(unit(0, "cm")),
      legend.text = element_text(face = "italic", size = 15),
      legend.title = element_text(face = "italic", size = 15),
      plot.title = element_text(
        face = "bold",
        size = rel(textSize),
```

```
        hjust = 0.5
      )
    )
```



We can also generate a 3D plot for all samples' scores against 3 gene signatures.

```
library(plotly)
all$group = as.factor(all$group)

p <- plot_ly(all, x = ~scoredf1.TotalScore, y = ~scoredf2.TotalScore,
             z = ~scoredf1.TotalScore, color = ~group,
             colors = c('red', 'blue', 'green')) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'iCGS1'),
                      yaxis = list(title = 'iCGS2'),
                      zaxis = list(title = 'iCGS3')))
```

Build the Classification model

Train a multinomial classification model with all samples's scores and the response variable as the iCluster labels.

```
# Train a multinorm classification model with all samples's scores
all$group = as.character(all$group)


library(nnet)
```

```
regressionLm = multinom(group ~ ., data = all, family = binomial(link="logit") )
```

```
## # weights:  15 (8 variable)
## initial  value 201.046049
## iter  10 value 61.965319
## iter  20 value 59.762476
## iter  30 value 59.539105
## iter  40 value 59.518693
## iter  50 value 59.514911
## iter  60 value 59.514666
## final  value 59.514657
## converged
```

```
summary(regressionLm)
```

```
## Call:
## multinom(formula = group ~ ., data = all, family = binomial(link = "logit"))
##
## Coefficients:
##           (Intercept) scoredf1.TotalScore scoredf2.TotalScore
## iCluster2    1.154555           0.3006629             6.62307
## iCluster3    3.384369          -1.3603609            -1.71346
##           scoredf3.TotalScore
## iCluster2             22.4011
## iCluster3             45.4204
##
## Std. Errors:
##           (Intercept) scoredf1.TotalScore scoredf2.TotalScore
## iCluster2    1.402704            3.892323            3.087184
## iCluster3    1.577414            4.829124            3.949955
##           scoredf3.TotalScore
## iCluster2            5.780252
## iCluster3            7.589460
##
## Residual Deviance: 119.0293
## AIC: 135.0293
```

```
predictions = predict(regressionLm, all, type="class")

# The mis-classfication error:
sum(!predictions==all$group)/length(all$group)
```

```
## [1] 0.1420765
```

New FUDAN dataset

In the next step, new microarray gene expression dataset of liver cancer samples were downloaded and scored. We use the multinomial classification model we trained before to classify the new sample cohort into iClusters and perform the survival analysis to check whether iCluster1's survival rate is the lowest.

Download the new gene expression data from GEO.

```
library(GEOquery)
gse = getGEO(GEO = "GSE14520")
gse[[1]]
# I saved the downloaded data into RData object
save(gse, file = "gse.RData")
```

Get the FUDAN sample cohort gene expression intensity dataset

The gene set from GEO omin, GSE14520, microarray gene expression dataset

```
load("./gse.RData")
newHCC = gse[[1]]
exprs(newHCC)[1:5,1:5]
```

```
##           GSM362958 GSM362959 GSM362960 GSM362961 GSM362962
## 1007_s_at     6.876     7.648     7.915     6.662     7.124
## 1053_at       4.651     4.283     4.250     4.105     3.928
## 117_at        6.775     3.796     3.380     4.483     3.639
## 121_at        5.578     6.213     5.579     6.590     6.151
## 1255_g_at     3.195     3.269     3.467     3.547     3.328
```

```
summary(exprs(newHCC)[,1:5])
```

```
##    GSM362958         GSM362959         GSM362960         GSM362961
##  Min.   : 2.609   Min.   : 2.694   Min.   : 2.771   Min.   : 2.783
##  1st Qu.: 3.639   1st Qu.: 3.695   1st Qu.: 3.747   1st Qu.: 3.850
##  Median : 4.431   Median : 4.424   Median : 4.456   Median : 4.505
##  Mean   : 5.216   Mean   : 5.135   Mean   : 5.154   Mean   : 5.068
##  3rd Qu.: 6.388   3rd Qu.: 6.112   3rd Qu.: 6.056   3rd Qu.: 5.745
##  Max.   :13.790   Max.   :13.827   Max.   :13.763   Max.   :13.969
##    GSM362962
##  Min.   : 2.763
##  1st Qu.: 3.710
##  Median : 4.450
##  Mean   : 5.187
##  3rd Qu.: 6.174
##  Max.   :13.797
```

Get the gene expression matrxi for all new samples

```
dim(pData(newHCC))
```

```
## [1] 445  46
```

```
dim(fData(newHCC))
```

```
## [1] 22268    16
```

```
geneMatrix = exprs(newHCC)

# find probes with Symbol
features = fData(newHCC)
pheoNew = pData(newHCC)
```

```
tumourSample = pheoNew$geo_accession[pheoNew$`Tissue:ch1`=="Liver Tumor Tissue"]
tumourSample = c(tumourSample,
                 pheoNew$geo_accession[pheoNew$`tissue:ch1`=="Liver Tumor Tissue"])
tumourSample = tumourSample[!is.na(tumourSample)]
keep = features[features$`Gene Symbol` %in% mapToType$hgnc_symbol[mapToType$transcript_biotype=="protei

# keep = keep[!keep$Protein_Product=="",]
#
geneMatrix = geneMatrix[keep$ID,]
rownames(geneMatrix) = keep$`Gene Symbol`

geneMatrix = geneMatrix[, tumourSample]

dim(geneMatrix)
```

```
## [1] 18135   225
```

Scoring FUDAN samples using singscore against the three iCluster gene signatures

```
rankedNew = rankGenes(geneMatrix)
scoredNewDf1 = simpleScore(rankedNew, upSet = up_ic1_gs, downSet = dn_ic1_gs)
```

```
## Warning in singscoring(rankData, upSet = upSet, downSet = downSet,
## subSamples = subSamples, : 62 genes missing: QSOX1, SLC45A4, LINGO1, HTRA3,
## WTIP, PAPLN, DMKN, RAB34, CERCAM, ANO9, CDCA7, SPECC1, LRFN1, SEL1L3,
## WNK2, FAM19A5, FNDC1, PMEPA1, C6orf132, GAREM2, SSC5D, ZNF853, ZNF469,
## METRNL, MXRA8, P3H3, YBX3, VCAN, DACT3, EBF4, PLPP2, FNDC10, SRRM3, LPAR2,
## MELTF, HID1, ADGRL1, PPP2R2C, TMEM119, CPXM1, FAM155B, EVC2, ADAP1, FOXS1,
## RAP1GAP2, EPCAM, SULF2, LPAR1, CTHRC1, MISP, GGT5, SAMD11, B3GNT7, PLEKHH2,
## C4orf48, PLEKHG4, GLIS3, HAPLN3, PIMREG, WIPF3, SPIRE1, PODN
```

```
## Warning in singscoring(rankData, downSet, NULL, subSamples, FALSE,
## dispersionFun): 31 genes missing: ACSM5, CYP8B1, ABCG8, ACSM2A, DMGDH,
## ETNPPL, ACSM2B, UGT2B10, ASPDH, CMBL, GLYATL1, FAM184A, NUGGC, ACSS3, MLIP,
## GBP7, ADH4, A1BG, AGXT2, SLC47A1, RTP3, AGMO, FGGY, ACKR2, SPDYC, LDHD,
## ENPP7, CES3, IYD, IL27, ARID3C
```

```
scoredNewDf2 = simpleScore(rankedNew, upSet = up_ic2_gs, downSet = dn_ic2_gs)
```

```
## Warning in singscoring(rankData, upSet = upSet, downSet = downSet,
## subSamples = subSamples, : 22 genes missing: ASPDH, RIPPLY1, SLC25A47,
## CYP8B1, LRCOL1, ACSM5, ACKR2, ETNPPL, RTP3, GLYATL1, SAA2-SAA4, AGXT2,
## SAA2, NUGGC, SLC9B2, A1BG, CCL14, ENPP7, ACSM2A, ABCG8, DEPDC7, SLC13A5
```

```
scoredNewDf3 = simpleScore(rankedNew, upSet = up_ic3_gs, downSet = dn_ic3_gs)
```

```
## Warning in singscoring(rankData, upSet = upSet, downSet = downSet,
## subSamples = subSamples, : 19 genes missing: LYPD8, CNTNAP4, COX7B2,
## FAM133A, CSMD1, GPR158, SLC44A5, DCAF4L2, FAM184A, CHRM3, GBP7, GSTA2,
## FGGY, ABCC11, ACSM5, DNAH11, HMGN5, EME1, ABCG8
```

```
## Warning in singscoring(rankData, downSet, NULL, subSamples, FALSE,
## dispersionFun): 51 genes missing: SULF2, IGF2, GGT5, QSOX1, ZNF853,
## PDZD4, WTIP, HHIPL1, RAB34, SH3RF3, PRICKLE2, CCDC3, RIMKLB, P3H3, GPBAR1,
## KIAA1211L, YBX3, IFITM10, MXRA8, METRNL, SSC5D, DACT3, FAM109B, OSBPL5,
```

```
## DLL1, EBF4, APLNR, PODN, ADAMTS15, GALNT16, LPAR1, CERCAM, INMT, JCAD,
## C11orf96, FOXS1, ZNF469, KNDC1, SORCS2, ZNF521, HTRA3, SPECC1, LINGO1,
## TDRP, ACKR1, CCDC80, SLC44A2, HDAC7, KIF26A, KIRREL1, TMEM119
```

```
newData = data.frame(GS1 = scoredNewDf1$TotalScore,
                     GS2 = scoredNewDf2$TotalScore,
                     GS3 = scoredNewDf3$TotalScore)
rownames(newData) = rownames(scoredNewDf1)
colnames(newData) = c("scoredf1.TotalScore", "scoredf2.TotalScore",
                      "scoredf3.TotalScore")

newPredictions =  predict(regressionLm, newData, type="class")
# group1 = rownames(scoredNewDf1[order(scoredNewDf1$TotalScore, decreasing = TRUE),][1:35,])
# group2 = rownames(scoredNewDf1[order(scoredNewDf1$TotalScore, decreasing = TRUE),][36:100,])

newDf = data.frame(pred = newPredictions, "GSM" = rownames(newData))
```

Survival Analysis of iClustered FUDAN samples

With the new samples' predicted iCluster labels, we do survival analysis

```
library(survival)

sur.os = read.delim("~/Documents/davisLab/HCC/GSE14520_Extra_Supplement.txt",
                    quote="", stringsAsFactors=FALSE)

surdf = merge(newDf, sur.os, by.x = "GSM", by.y = "Affy_GSM")
newData$GSM = rownames(newData)
surdf = merge(surdf, newData, by.x = "GSM", by.y = "GSM")

mydata<-surdf
mydata = na.omit(mydata)
## Step (1)
## Create the Survival Object

mySurv<-Surv(time=mydata$Survival.months, event = mydata$Survival.status)
#class(mySurv)
head(mySurv)
```

```
## [1] 28.2   9.5  66.1+ 67.4+ 66.6+ 66.1+
```

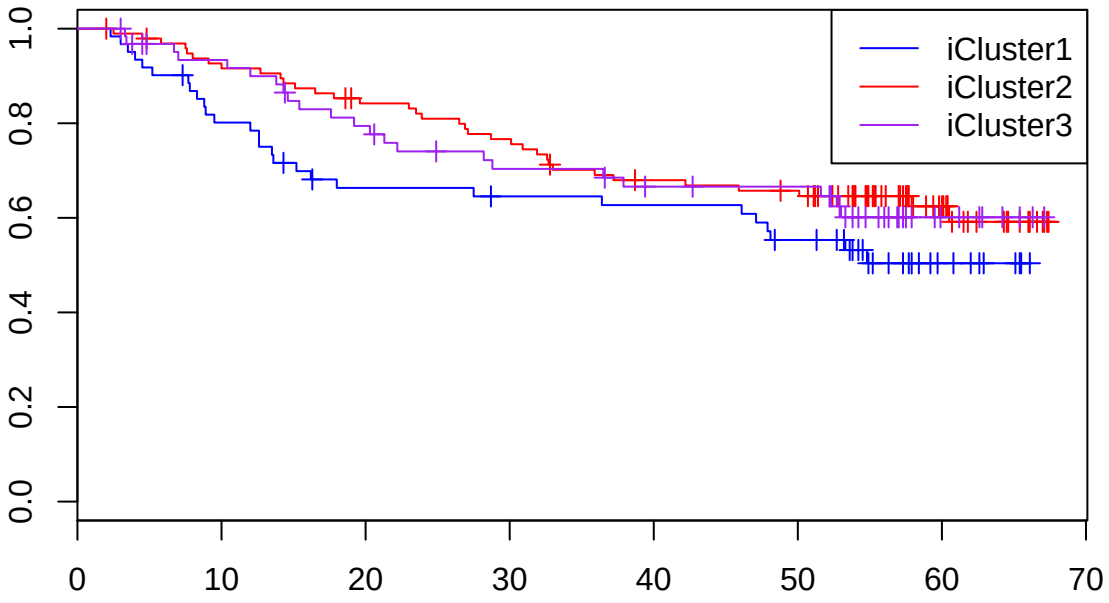
```
mydata$AgeGroup = "Young"
mydata$AgeGroup[mydata$Age> median(mydata$Age,na.rm = TRUE)] = "Old"
mydata$AgeGroup = as.factor(mydata$AgeGroup)

## specify predictor variable in =the formula
myfit<-survfit(mySurv~mydata$pred)
table(mydata$pred)
```

```
##
## iCluster1 iCluster2 iCluster3
##        61        97        63
```

```
survdiff(mySurv~pred+mydata$Main.Tumor.Size.....5.cm.,
         data = mydata)
```

```
## Call:
## survdiff(formula = mySurv ~ pred + mydata$Main.Tumor.Size......5.cm.,
##     data = mydata)
##
##                                                          N Observed
## pred=iCluster1, mydata$Main.Tumor.Size......5.cm.=large 26       18
## pred=iCluster1, mydata$Main.Tumor.Size......5.cm.=small 35       10
## pred=iCluster2, mydata$Main.Tumor.Size......5.cm.=.      1        0
## pred=iCluster2, mydata$Main.Tumor.Size......5.cm.=large 24        9
## pred=iCluster2, mydata$Main.Tumor.Size......5.cm.=small 72       26
## pred=iCluster3, mydata$Main.Tumor.Size......5.cm.=large 30       11
## pred=iCluster3, mydata$Main.Tumor.Size......5.cm.=small 33       11
##                                                         Expected (O-E)^2/E
## pred=iCluster1, mydata$Main.Tumor.Size......5.cm.=large    6.408   20.9701
## pred=iCluster1, mydata$Main.Tumor.Size......5.cm.=small   14.929    1.6276
## pred=iCluster2, mydata$Main.Tumor.Size......5.cm.=.        0.565    0.5654
## pred=iCluster2, mydata$Main.Tumor.Size......5.cm.=large    8.646    0.0145
## pred=iCluster2, mydata$Main.Tumor.Size......5.cm.=small   31.096    0.8351
## pred=iCluster3, mydata$Main.Tumor.Size......5.cm.=large   10.031    0.0936
## pred=iCluster3, mydata$Main.Tumor.Size......5.cm.=small   13.324    0.4053
##                                                         (O-E)^2/V
## pred=iCluster1, mydata$Main.Tumor.Size......5.cm.=large   22.8332
## pred=iCluster1, mydata$Main.Tumor.Size......5.cm.=small    1.9779
## pred=iCluster2, mydata$Main.Tumor.Size......5.cm.=.        0.5702
## pred=iCluster2, mydata$Main.Tumor.Size......5.cm.=large    0.0161
## pred=iCluster2, mydata$Main.Tumor.Size......5.cm.=small    1.3206
## pred=iCluster3, mydata$Main.Tumor.Size......5.cm.=large    0.1064
## pred=iCluster3, mydata$Main.Tumor.Size......5.cm.=small    0.4812
##
##  Chisq= 24.7  on 6 degrees of freedom, p= 0.000384
```

```
###  plot the inverse of a survival function

plot(myfit, col=c("blue","red","purple"), mark=3) ## mark.time=T marked at
## each censoring time
legend("topright", c("iCluster1","iCluster2",'iCluster3'),
       col=c("blue","red","purple"), lty=1)
```



supplementary

This section we did a survival analysis for the origial iClustered samples from information enclosed in the clinical.tsv downloaded from GDC.

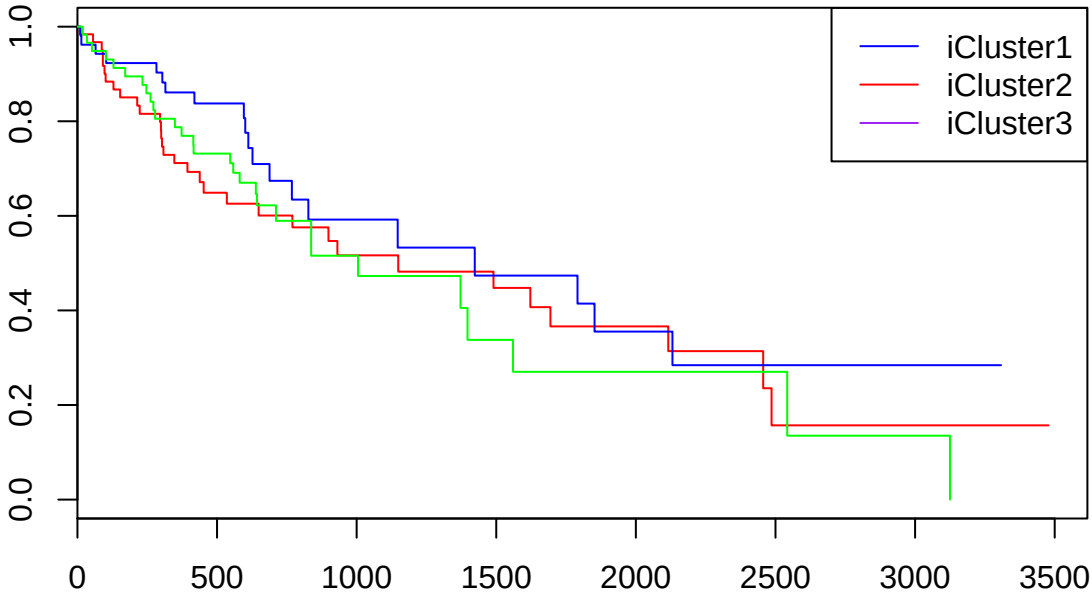
```
clinical <- read.delim("~/Documents/davisLab/HCC/clinical.cart.2018-05-16/clinical.tsv", quote="", stri

time = rep("0",dim(clinical)[1])
time = sapply(1:dim(clinical)[1], function(x){
  if(clinical$days_to_death[x]=="--"){
    time[x] = clinical$days_to_last_follow_up[x]}
  else{
    time[x] = clinical$days_to_death[x]}})

clinical$time =as.numeric(time)
```

```
## Warning: NAs introduced by coercion
```

```
dim(merge(clinical, g, by.x = 'submitter_id', by.y = "Case.ID"))
```

```
## [1] 183 133
```

```
icluster_clinical = merge(clinical, g, by.x = 'submitter_id', by.y = "Case.ID")
icluster_clinical$vital_status[icluster_clinical$vital_status=="dead"] = 1
icluster_clinical$vital_status[icluster_clinical$vital_status=="alive"] = 0
icluster_clinical$vital_status = as.numeric(icluster_clinical$vital_status)
tcgaSurv = Surv(time=icluster_clinical$time, event = icluster_clinical$vital_status)

tcgaSurv.fit <- survfit(tcgaSurv~icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`)
plot(tcgaSurv.fit, col=c("red","blue","green"))
legend("topright", c("iCluster1","iCluster2",'iCluster3'), col=c("blue","red","purple"), lty=1)
```




```
survdiff(tcgaSurv~icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`)
```

```
## Call:
## survdiff(formula = tcgaSurv ~ icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`)
##
##                                                                    N
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:1 65
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:2 55
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:3 63
##                                                                 Observed
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:1       32
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:2       20
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:3       29
##                                                                 Expected
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:1     29.9
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:2     25.4
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:3     25.7
##                                                                 (O-E)^2/E
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:1     0.150
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:2     1.139
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:3     0.412
##                                                                 (O-E)^2/V
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:1     0.241
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:2     1.666
## icluster_clinical$`iCluster clusters (k=3, Ronglai Shen)`=iCluster:3     0.615
##
##  Chisq= 1.7  on 2 degrees of freedom, p= 0.425
```

References

Ally, Adrian, Miruna Balasundaram, Rebecca Carlsen, Eric Chuah, Amanda Clarke, Noreen Dhalla, Robert A. Holt, et al. n.d. "Comprehensive and Integrative Genomic Characterization of Hepatocellular Carcinoma." *Cell* 169 (7). Elsevier: 1327–1341.e23. doi:10.1016/j.cell.2017.05.046.

Law, CW, M Alhamdoosh, S Su, GK Smyth, and ME Ritchie. 2016. "RNA-Seq Analysis Is Easy as

1-2-3 with Limma, Glimma and edgeR [Version 2; Referees: 3 Approved]." *F1000Research* 5 (1408). doi:10.12688/f1000research.9005.2.

Roessler, Stephanie, Hu-Liang Jia, Anuradha Budhu, Marshonna Forgues, Qing-Hai Ye, Ju-Seog Lee, Snorri S. Thorgeirsson, et al. 2010. "A Unique Metastasis Gene Signature Enables Prediction of Tumor Relapse in Early-Stage Hepatocellular Carcinoma Patients." *Cancer Research* 70 (24). American Association for Cancer Research: 10202–12. doi:10.1158/0008-5472.CAN-10-2607.