# iClusterPlus: integrative clustering of multiple genomic data sets

Qianxing Mo[1] and Ronglai Shen[2]

February 26, 2023

[1]Department of Biostatistics & Bioinformatics
H. Lee Moffitt Cancer Center & Research Institute
qianxing.mo@moffitt.org

[2]Deparment of Epidemiology and Biostatistics
Memorial Sloan-Kettering Cancer Center
shenr@mskcc.org

## Contents

## 1 Introduction

iClusterPlus is developed for integrative clustering analysis of multi-type genomic data and is an enhanced version of iCluster proposed and developed by Shen, Olshen and Ladanyi (2009). Multi-type genomic data arise from the experiments where biological samples (e.g., tumor samples) are analyzed by multiple techniques, for instance, array comparative genomic hybridization (aCGH), gene expression microarray, RNA-seq and DNA-seq, and so on. Examples of these data can be obtained from the Cancer Genome Atlas (TCGA) (http://cancergenome.nih.gov/).

Originally, Shen, Olshen and Ladanyi (2009) proposed a latent variable regression with a lasso constraint (Tibshirani, 1996) for joint modeling of multiple omics data types to identify common latent variables that can be used to cluster patient samples into biologically and clinically relevant disease sub-types. In a followup work, Shen, Wang and Mo (2012) further incorporated elasticnet (Zou and Hastie, 2005) and fused lasso (Tibshirani, 2005) into the integration framework. This document discusses the iClusterPlus method that extends the framework to allow integration of binary, categorical, count, and continuous data types (Mo et al. 2013). In iClusterPlus model, binary observations such as somatic mutation are modeled as Binomial processes; categorical observations such as copy number states (gain, normal, loss) are realizations of Multinomial random variables; counts are modeled as Poisson random processes; and continuous measures are modeled by Gaussian distributions. We simultaneously regress the observations (somatic mutation, DNA copy number, DNA methylation, mRNA expression) under their proper distributional assumptions to a common set of latent variables that represent a set of underlying oncogenic processes. To identify the genomic features (e.g., ERBB2 amplification and over-expression) that make important contributions to the latent onco- genic process, we use a penalized regression approach by applying the lasso (L1-norm) penalty (Tibshirani, 1996) in generalized linear models. This is achieved by directly incorporating the Fortran source code written by Friedman, Hastie and Tibshirani (2009) that is available in the glmnet package (`http://cran.r-project.org/web/packages/glmnet/index.html`). Finally, we show how to use iClusterBayes method, a newly developed Bayesian integrative clustering method to analyze multi-omics data (Mo et al., 2018).

In this document, we use the TCGA glioblastoma data set as an example to demonstrate how to use iClusterPlus to perform an integrative clustering analysis of somatic mutation, DNA copy number and gene expression data.

## 2 Data and Pre-processing

A description of the TCGA glioblastoma data set can be found in TCGA (2008). The GBM datasets were downloaded from the Cancer Genome Atlas public data portal, and from the cBio Cancer Genomics Portal (`http://cbioportal.org/`) at the Memorial Sloan-Kettering Cancer Center. Somatic mutation data are available in a total of 91 matched tumor?normal pairs and in 601 selected genes. DNA copy number alterations (CNAs) were measured on three microarray platforms (Agilent 244K, SNP6, Illumina 550K) and analyzed with multiple analytical algorithms. Level 3 normalized and segmented data were used. In our data preprocessing step, we reduce multi-sample array CGH data to 1K-5K non-redundant regions for subsequent clustering analysis. For mRNA expression data, unified gene expression data across three microarray platforms (Affymetrix Human Exon 1.0 ST GeneChips, Affymetrix HT-HG-U133A GeneChips, and custom designed Agilent 244K array) as described in (Verhaak et al. 2010) were used. A set of 1,740 most variable genes were used for the analysis. The "triplet" data set (mutation, copy number, expression) were available on 84 samples for integrative analysis.
The somatic mutation data should be stored in binary matrix (1: mutation, 0: no mutation) with the rows and columns corresponding to the samples and genes, respectively. We

recommend filtering out genes with too few mutations for clustering (e.g., less than 5%).

```
> library(iClusterPlus)
> library(GenomicRanges)
> library(gplots)
> library(lattice)
> data(gbm)
> dim(gbm.mut)

[1]  84 306

> mut.rate=apply(gbm.mut,2,mean)
> gbm.mut2 = gbm.mut[,which(mut.rate>0.02)]
> gbm.mut2[1:10,1:8]

             A2M ABCC4 ADAMTSL3 ASXL1 BAI3 BCAR1 BCL11A BCL11B
TCGA.02.0001   0     0        0     0    0     0      0      0
TCGA.02.0003   0     0        0     0    0     0      0      0
TCGA.02.0006   0     0        0     0    0     0      0      0
TCGA.02.0007   0     0        0     0    0     0      0      0
TCGA.02.0009   0     0        0     0    0     0      0      0
TCGA.02.0010   0     0        1     1    1     0      0      1
TCGA.02.0011   0     0        0     0    0     0      0      0
TCGA.02.0014   0     0        0     0    0     0      1      0
TCGA.02.0021   0     0        0     0    0     0      0      0
TCGA.02.0024   1     0        0     0    0     0      0      0
```

For gene expression data, we recommend using the top variable genes for integrative clustering analysis, which can be obtained by variance filtering. For example, we use the top 1740 genes for our iCluster analysis.

```
> dim(gbm.exp)

[1]   84 1740

> # the rows and columns corresponding to the samples and genes respectively
> gbm.exp[1:3,1:8]

                FSTL1     MMP2   BBOX1    GCSH     EDN1    CXCR4    SALL1
TCGA.02.0001 -0.66392 -0.27716 0.79896 0.09005  0.46557  0.30278  0.76869
TCGA.02.0003 -0.28438  1.00445 0.19157 0.92115  1.08181 -0.03790  0.00452
TCGA.02.0006  0.98890  0.19374 0.93830 0.49317 -0.22644  1.43145 -0.38401
                 MMP7
TCGA.02.0001  0.55745
TCGA.02.0003 -0.04971
TCGA.02.0006  1.58288
```

It is a challenge to incorporate raw or normalized copy number data for iCluster analysis considering the high dimensionality and spatial correlation of the data. Based on our experience, we think it is more feasible to use the segmentation results produced by the DNAcopy package (`http://www.bioconductor.org/packages/release/bioc/html/DNAcopy.html`).

```
> dim(gbm.seg)

[1] 16295     6

> gbm.seg[1:3,]  #gbm.cn is the segmentation results produced by DNAcopy

        sample chromosome     start       end num.mark seg.mean
1 TCGA.02.0001          1 150823073 150848509        5  -2.1537
2 TCGA.02.0001          1 150852858 167483267     1814   0.1907
3 TCGA.02.0001          1 167493768 167507882        2  -3.4343
```

We reduce the GBM copy number regions to  5K by removing the redundant regions using function CNregions.

```
> data(variation.hg18.v10.nov.2010)
> gbm.cn=CNregions(seg=gbm.seg,epsilon=0,adaptive=FALSE,rmCNV=TRUE,
+   cnv=variation.hg18.v10.nov.2010[,3:5],
+   frac.overlap=0.5, rmSmallseg=TRUE,nProbes=5)

Removing CNV...

> dim(gbm.cn)

[1]   84 5512

> gbm.cn[1:3,1:5]

             chr1.554268-554287 chr1.554287-736483 chr1.736483-746956
TCGA.02.0001             0.2077             0.2077             0.2077
TCGA.02.0003            -0.0096            -0.0096            -0.0096
TCGA.02.0006             0.0027             0.0027             0.0027
             chr1.746956-757922 chr1.757922-769590
TCGA.02.0001             0.2077             0.2077
TCGA.02.0003            -0.0096            -0.0096
TCGA.02.0006             0.0027             0.0027
```

Here seg is the DNAcopy segmentation output. In the first step, we define a set of non-redundant regions with parameter epsilon that denotes the maximum distance (Euclidean) between adjacent probes tolerated for defining a non-redundant region. epsilon=0 is equivalent as taking the union of all unique break points across the n samples. Default epsilon=0.005. We then take the medoid signature as the representative copy number profile for that region, an approach similar to van Wieringen and van de Wiel (2002). The degree of dimension reduction is proportional to the number of samples included. We recommend setting an epsilon such that the reduced dimension is less than 10K. When sample size is large, an adaptive dimension reduction is more effective. Instead of setting absolute threshold epsilon, setting adaptive=T will reduce dimension proportional to upper quantile of the distances. default is False. rmCNV=T remove germline CNV. When set to True, one must supply a list of germline CNVs as cnv=cnv. The additional argument rmSmallseg removes small segments likely to be noise. Default is False. When set of True, nProbes, the segment length threshold below which rmSmallseg will exclude should be specified.

Sort gbm.cn to make sure all the samples are in the same order.

```
> gbm.cn=gbm.cn[order(rownames(gbm.cn)),]
> # check if all the samples are in the same order for the three data sets
> all(rownames(gbm.cn)==rownames(gbm.exp))

[1] TRUE

> all(rownames(gbm.cn)==rownames(gbm.mut2))

[1] TRUE
```

# 3 Integrative clustering analysis

Given multiple genomic data types (e.g., mutation, copy number, gene expression, etc) measured in the same set of samples, and specified sparsity parameter values, iClusterPlus fits a regularized latent variable model based clustering that generates an integrated cluster assignment based on joint inference across data types. Below is a one-liner to run iClusterPlus given the desired number of eigen-features k (number of clusters is k+1) and the values of parameter set lambda (which determines how many genomic features will have nonzero weights on the fused eigen-feature). Normally, we have to optimize k and lambda through model tuning which we discuss in the next section.

```
> fit.single=iClusterPlus(dt1=gbm.mut2,dt2=gbm.cn,dt3=gbm.exp,
+   type=c("binomial","gaussian","gaussian"),
+   lambda=c(0.04,0.61,0.90),K=2,maxiter=10)
```

# 4 Model tuning using tune.iClusterPlus

Using parallel computing, tune.iClusterPlus samples a series of lambda values from the parameter space based on the Uniform design (Fang and Wang, 1995) to search for the best model. The number of points to sample (n.lambda) depends on the number of data types and can take the following values: If we know the number of clusters in the samples, we may just choose the corresponding k (the number of latent variables) for the cluster analysis. We use the rule that the number of clusters equals k+1. If we don't know the cluster number, we can test the k from 1 to N (a reasonable number of clusters). In this example, we let the k from 1 to 5.

| Number of data types | n.lambda |
|---|---|
| 1 | any |
| 2 | 8, 13, 21, 34, 55, 89, 144, 233, 377, 610 |
| 3 | 35, 101, 135, 185, 266, 418, 597, 828, 1010 |
| 4 | 307, 562, 701, 1019, 2129, 3001, 4001, 5003, 6007 |

```
> set.seed(123)
> date()

[1] "Sat Feb 25 11:57:51 2023"

> for(k in 1:5){
+   cv.fit = tune.iClusterPlus(cpus=12,dt1=gbm.mut2,dt2=gbm.cn,dt3=gbm.exp,
+     type=c("binomial","gaussian","gaussian"),K=k,n.lambda=185,
+     scale.lambda=c(1,1,1),maxiter=20)
+   save(cv.fit, file=paste("cv.fit.k",k,".Rdata",sep=""))
+ }

185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation
185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation
185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation
185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation
185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation

> date()

[1] "Sat Feb 25 19:15:39 2023"
```

Here for demonstration purpose, we specify n.lambda=185 sampling points (a modest value for this setting) which may result in some variability. As a result, either k=2 can be chosen as the best k based on % explained variation (percentEV). Use set.seed() to make sure you can reproduce your results from independent runs. For general practice, set n.lambda=NULL to use the default value.

## 5 Model selection

Now we illustrate how to analyze and interpret the output of tune.iClusterPlus. The first step is to examine model selection criteria. For each k, we use Bayesian information criteria (BIC) to select the best sparse model with the optimal combination of penalty parameters. To select the best k, we compute the deviance ratio which is the ratio of the log-likelihood(fitted) - log-likelihood(null model) divided by the log-likelihood (full model) - log-likelihood(null model). The deviance ratio can be interpreted as the percentEV. We choose the k where the curve of percentEV levels off. Below, we show how to do this in R.

```
> output=alist()
> files=grep("cv.fit",dir())
> for(i in 1:length(files)){
+   load(dir()[files[i]])
+   output[[i]]=cv.fit
+ }
> nLambda = nrow(output[[1]]$lambda)
> nK = length(output)
> BIC = getBIC(output)
> devR = getDevR(output)
```

Now we get the ID for the lambda vector at which the BIC is minimum. Then we obtain the deviance ratio of the lambda vector at which the BIC is minimum.

```
> minBICid = apply(BIC,2,which.min)
> devRatMinBIC = rep(NA,nK)
> for(i in 1:nK){
+   devRatMinBIC[i] = devR[minBICid[i],i]
+ }
```

See Figure 1 for the plot of number of clusters vs. percent of explained variation.
The optimal k (number of latent variables) is where the curve of %Explained variation levels off. By examining Figure 1, we choose to present the three-cluster results. Get cluster membership (note number of clusters is k+1):

```
> clusters=getClusters(output)
> rownames(clusters)=rownames(gbm.exp)
> colnames(clusters)=paste("K=",2:(length(output)+1),sep="")
> #write.table(clusters, file="clusterMembership.txt",sep='\t',quote=F)
> k=2
> best.cluster=clusters[,k]
> best.fit=output[[k]]$fit[[which.min(BIC[,k])]]
```

If the data are noisy, we may not see a curve like 1. For noisy data, in general, when k increases, the deviance ratio also increases. In this case, we suggest the user make heat maps of the data sets, and then decide the optimal number of clusters based on the features' pattern.

```
> plot(1:(nK+1),c(0,devRatMinBIC),type="b",xlab="Number of clusters (K+1)",
+      ylab="%Explained Variation")
```

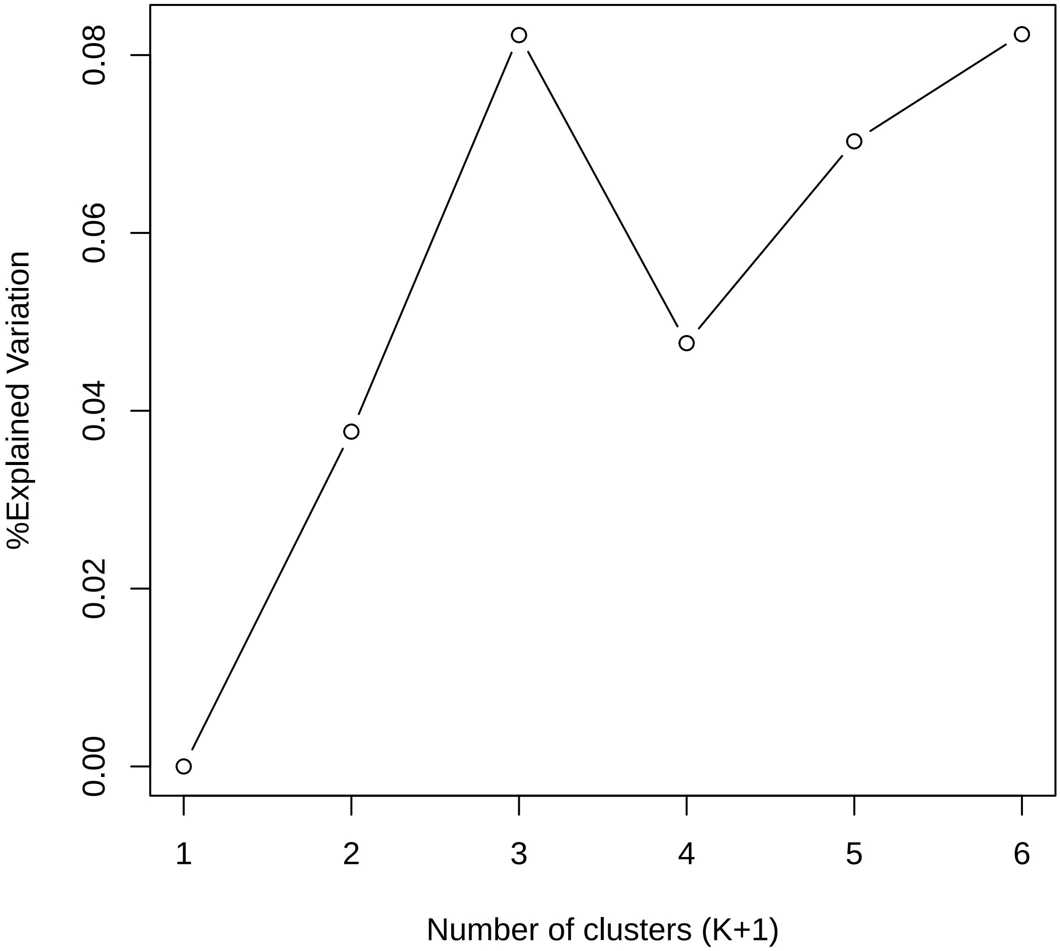

Figure 1: Number of clusters vs. percent of explained variation.

## 6 Generate heatmap

We provide a function to plot heatmaps. If necessary, the users may modify plotHeatmap function to fit their own needs.

```
> chr=unlist(strsplit(colnames(gbm.cn),"\\."))
> chr=chr[seq(1,length(chr),by=2)]
> chr=gsub("chr","",chr)
> chr=as.numeric(chr)
> #truncate the values for a better image plot
> cn.image=gbm.cn
> cn.image[cn.image>1.5]=1.5
> cn.image[cn.image< -1.5]= -1.5
> exp.image=gbm.exp
> exp.image[exp.image>2.5]=2.5
> exp.image[exp.image< -2.5]= -2.5
```

See Figure 2 for the heatmap plot for the three-cluster result.

## 7 Feature selection

Select the top features based on lasso coefficient estimates for the 3-cluster solution.

```
> features = alist()
> features[[1]] = colnames(gbm.mut2)
> features[[2]] = colnames(gbm.cn)
> features[[3]] = colnames(gbm.exp)
> sigfeatures=alist()
> for(i in 1:3){
+   rowsum=apply(abs(best.fit$beta[[i]]),1, sum)
+   upper=quantile(rowsum,prob=0.75)
+   sigfeatures[[i]]=(features[[i]])[which(rowsum>upper)]
+ }
> names(sigfeatures)=c("mutation","copy number","expression")
> #print a few examples of selected features
> head(sigfeatures[[1]])

character(0)

> head(sigfeatures[[2]])

[1] "chr1.202692537-202801982" "chr4.52383858-52395837"
[3] "chr4.52395837-52522387"   "chr4.52522387-52989049"
[5] "chr4.52989049-53002654"   "chr4.53002654-53517879"

> head(sigfeatures[[3]])
> bw.col = colorpanel(2,low="white",high="black")
> col.scheme = alist()
> col.scheme[[1]] = bw.col
> col.scheme[[2]] = bluered(256)
> col.scheme[[3]] = bluered(256)
> plotHeatmap(fit=best.fit,datasets=list(gbm.mut2,cn.image,exp.image),
+             type=c("binomial","gaussian","gaussian"), col.scheme = col.scheme,
+             row.order=c(F,F,T),chr=chr,plot.chr=c(F,T,F),sparse=c(T,F,T),cap=c(F,T,F))
```

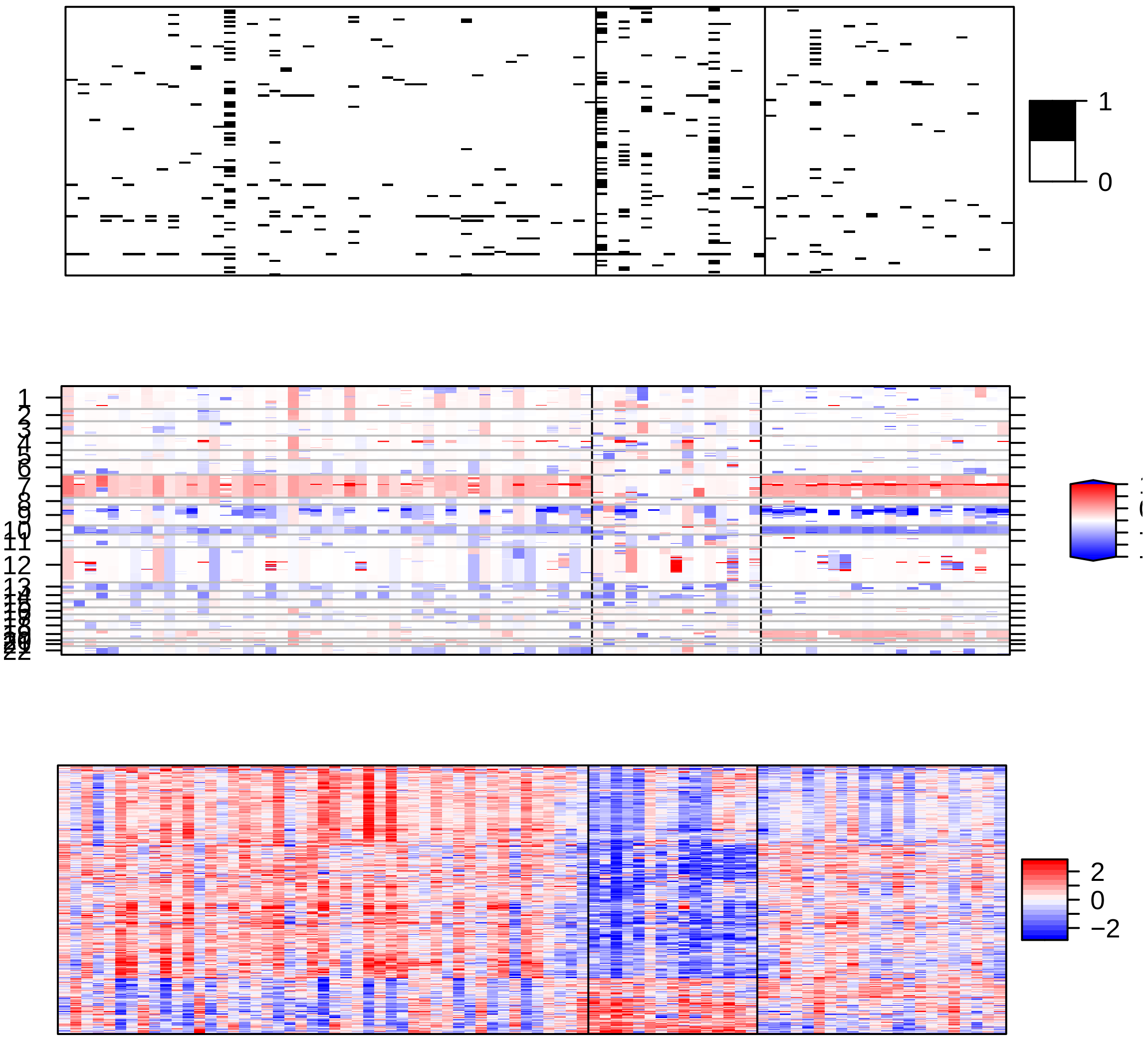

Figure 2: Heatmap of mutation (top panel), DNA copy number (middle panel), and mRNA expression (bottom panel) for the three-cluster solution. Rows are genomic features and columns are samples.

```
[1] "FSTL1"    "CXCR4"    "MMP7"     "ZEB1"     "KIAA1199" "SERPINF1"
```

We notice that no gene is selected from the mutation data, which indicates that the selected lambda value is too large and it is not in the same scale as those for the copy number and gene expression data. To solve this problem, we need to set the scale.lambda (an argument of tune.iClusterPlus) to a value between 0 and 1. For example, after several tries, we find that letting scale.lambda equal to 0.05 allows a more effective variable selection of the mutation data and the three-cluster solution is still one of the best solutions. The following is a repeat of the above analysis with a new scale.lambda for the mutation data.

```
> set.seed(123)
> date()

[1] "Sat Feb 25 19:15:43 2023"

> for(k in 1:5){
+   cv2.fit = tune.iClusterPlus(cpus=12,dt1=gbm.mut2,dt2=gbm.cn,dt3=gbm.exp,
+     type=c("binomial","gaussian","gaussian"),K=k,n.lambda=185,
+     scale.lambda=c(0.05,1,1),maxiter=20)
+   save(cv2.fit, file=paste("cv2.fit.k",k,".Rdata",sep=""))
+ }

185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation
185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation
185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation
185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation
185  points of lambdas are used to tune parameters.
Begin parallel computation
End parallel computation

> date()

[1] "Sun Feb 26 04:14:06 2023"

> output2=alist()
> files=grep("cv2.fit",dir())
> for(i in 1:length(files)){
+   load(dir()[files[i]])
+   output2[[i]]=cv2.fit
+ }
> nLambda = nrow(output2[[1]]$lambda)
> nK = length(output2)
> BIC = getBIC(output2)
> devR = getDevR(output2)

> minBICid = apply(BIC,2,which.min)
> devRatMinBIC = rep(NA,nK)
> for(i in 1:nK){
+   devRatMinBIC[i] = devR[minBICid[i],i]
+ }
```

See Figure 3 for the plot of number of clusters vs. percent of explained variation.

The optimal k (number of latent variables) is where the curve of %Explained variation levels off. By examining Figure 3, we choose to present the three-cluster results. Get cluster membership (note number of clusters is k+1):

```
> clusters=getClusters(output2)
> rownames(clusters)=rownames(gbm.exp)
> colnames(clusters)=paste("K=",2:(length(output2)+1),sep="")
> #write.table(clusters, file="clusterMembership.txt",sep='\t',quote=F)
> k=2
> best.cluster=clusters[,k]
> best.fit=output2[[k]]$fit[[which.min(BIC[,k])]]
```

See Figure 4 for the heatmap plot for the three-cluster result.

Select the top features based on lasso coefficient estimates for the 3-cluster solution.

```
> features = alist()
> features[[1]] = colnames(gbm.mut2)
> features[[2]] = colnames(gbm.cn)
> features[[3]] = colnames(gbm.exp)
> sigfeatures=alist()
> for(i in 1:3){
+   rowsum=apply(abs(best.fit$beta[[i]]),1, sum)
+   upper=quantile(rowsum,prob=0.75)
+   sigfeatures[[i]]=(features[[i]])[which(rowsum>upper)]
+ }
> names(sigfeatures)=c("mutation","copy number","expression")
> #print a few examples of selected features
> head(sigfeatures[[1]])

[1] "A2M"    "BRCA2"  "CAST"   "CENTG1" "CES3"   "CHAT"
```

```
> plot(1:(nK+1),c(0,devRatMinBIC),type="b",xlab="Number of clusters (K+1)",
+       ylab="%Explained Variation")
```

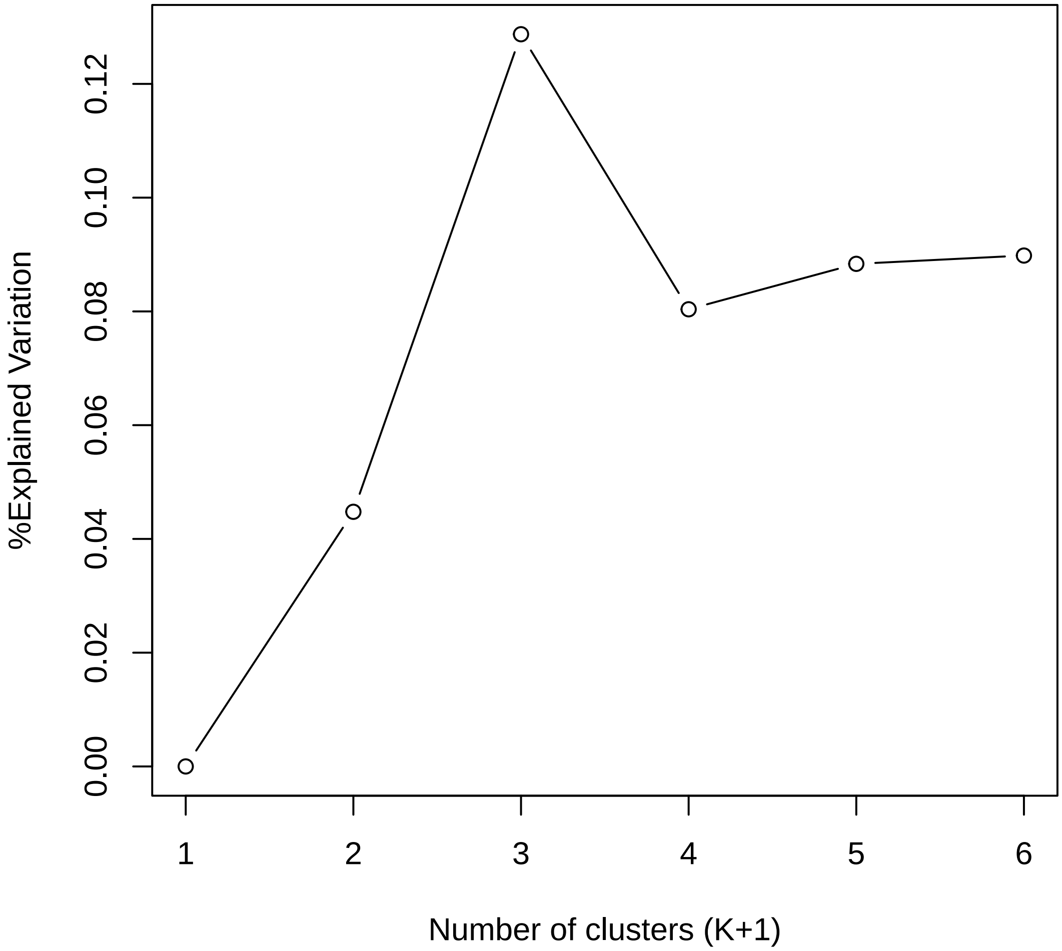

Figure 3: Number of clusters vs. percent of explained variation.

```
> plotHeatmap(fit=best.fit,datasets=list(gbm.mut2,cn.image,exp.image),
+               type=c("binomial","gaussian","gaussian"), col.scheme = col.scheme,
+               row.order=c(F,F,T),chr=chr,plot.chr=c(F,T,F),sparse=c(T,F,T),cap=c(F,T,F))
```

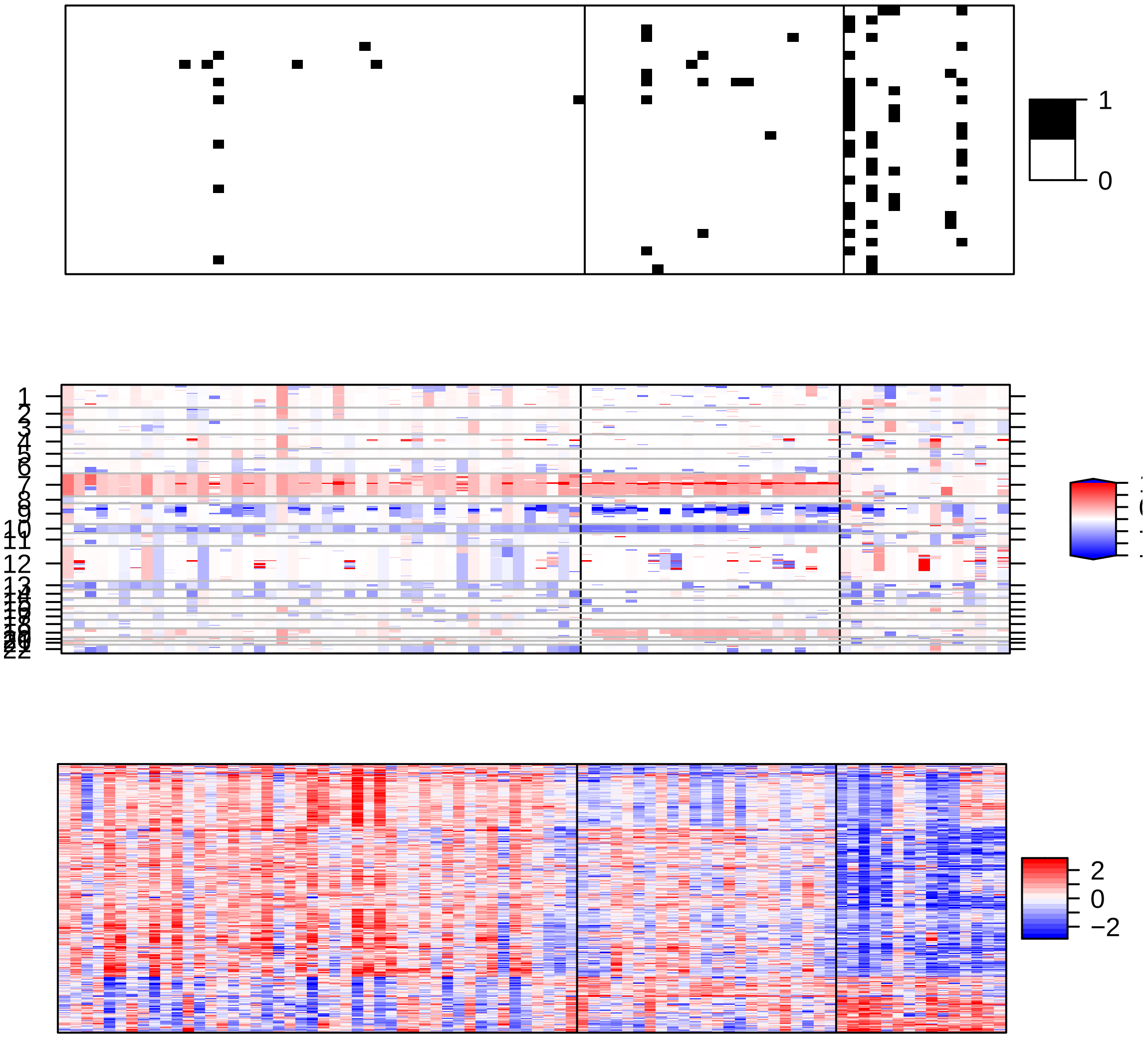

Figure 4: Heatmap of mutation (top panel), DNA copy number (middle panel), and mRNA expression (bottom panel) for the three-cluster solution. Rows are genomic features and columns are samples.

```
> head(sigfeatures[[2]])

[1] "chr1.202692537-202801982" "chr4.52383858-52395837"
[3] "chr4.52395837-52522387"   "chr4.52522387-52989049"
[5] "chr4.52989049-53002654"   "chr4.53002654-53517879"

> head(sigfeatures[[3]])

[1] "FSTL1"    "CXCR4"    "MMP7"     "ZEB1"     "KIAA1199" "SERPINF1"
```

Finally, we show how to analyze the GBM data set using iClusterBayes, a newly developed Bayesian integrative clustering mehtod (Mo et al., 2018). This method does not need to perform a grid search for the optimal lasso parameter, and thus it needs much less computing resource when the data sets are relatively large. In addition, it calculates the posterior probability of being a driver for each genomic feature, which can be used as a criterion for seleting genomic features that drive the integrative clustering.

```
> set.seed(123)
> date()

[1] "Sun Feb 26 04:14:09 2023"

> bayfit = tune.iClusterBayes(cpus=6,dt1=gbm.mut2,dt2=gbm.cn,dt3=gbm.exp,
+     type=c("binomial","gaussian","gaussian"),K=1:6,n.burnin=18000,
+     n.draw=12000,prior.gamma=c(0.5,0.5,0.5),sdev=0.05,thin=3)

Begin parallel computation
End parallel computation

> date()

[1] "Sun Feb 26 05:02:51 2023"

> save.image(file="gbmBayfit.RData")
> #load("gbmBayfit.RData")
```

BIC or deviance ratio may be used as a criterion to select an optimal value for k, the parameter that defines the number of clusters.

```
> allBIC = NULL
> devratio = NULL
> nK = length(bayfit$fit)
> for(i in 1:nK){
+   allBIC = c(allBIC,bayfit$fit[[i]]$BIC)
+   devratio = c(devratio,bayfit$fit[[i]]$dev.ratio)
+ }
```

```
> par(mar=c(4.0,4.0,0.5,0.5),mfrow=c(1,2))
> plot(1:nK, allBIC,type="b",xlab="k",ylab="BIC",pch=c(1,1,19,1,1,1))
> plot(1:nK,devratio,type="b",xlab="k",ylab="Deviance ratio",pch=c(1,1,19,1,1,1))
```

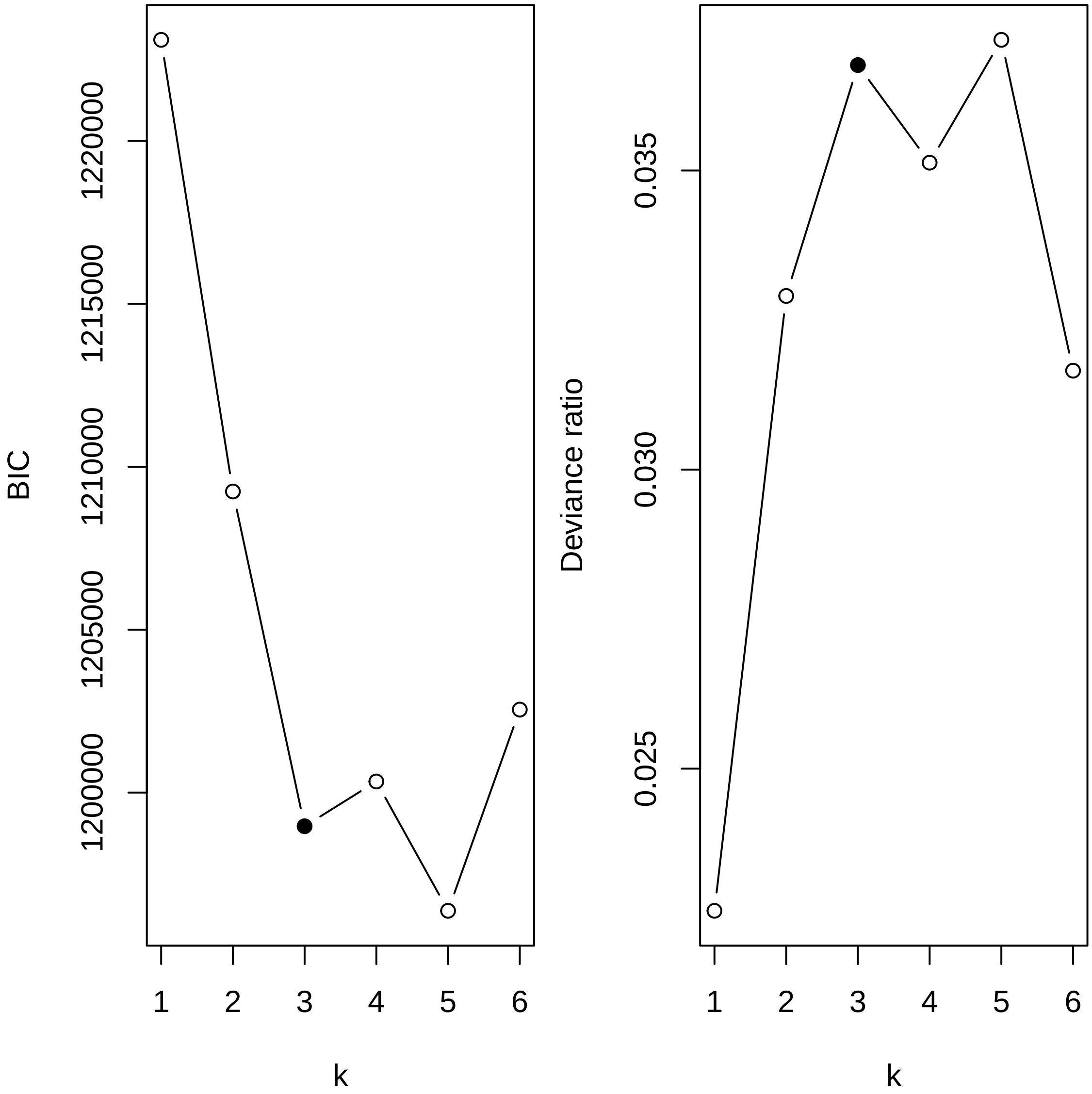

Figure 5: BIC and percent of explained variation vs. K.

From the BIC and deviance ratio plots, we see that k=3 is an optimal solution. However, if the data are noisy, the deviance ratio may keep increasing and the BIC may keep decreasing when k increases. In this case, we suggest the user make heat maps of the data sets, and then decide the optimal number of clusters based on the features' pattern on the heat maps.

Plot the posterior probability of genomic features when k=3.
 See Figure 7 for the heatmap plot for the four-cluster result.

Let's compare the three clusters (labeled as C1, C2, C3) given by iClusterPlus with the four clusters (labeled as B1,B2,B3,B4) given by iClusterBayes.

```
> best.cluster.Bayes = bayfit$fit[[3]]$clusters
> #best.cluster is the clusters generated by iClusterPlus; See the above code
> table(best.cluster,best.cluster.Bayes)
```

```
            best.cluster.Bayes
best.cluster  1  2  3  4
           1 20  2 21  3
           2  1  0  2 20
           3  0 14  1  0
```

We basically find that C1 corresponds to B1, C2 corresponds to B4, and C3 corresponds to B2 and B3, respectively. In other words, the clusters genered by the two methods are quite similar except for that iClusterBayes further divides the cluster C3 into 2 sub-clusters (B2 and B3).

```
> sessionInfo()
```

```
R Under development (unstable) (2023-01-14 r83615)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Catalina 10.15.7

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3/Resources/lib/libRlapack.dylib;  LAPACK

Random number generation:
 RNG:     L'Ecuyer-CMRG
 Normal:  Inversion
 Sample:  Rejection

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal
```

```
> par(mfrow=c(3,1))
> k=3
> plot(bayfit$fit[[k]]$beta.pp[[1]],xlab="Genes",ylab="Posterior probability",
+      main="Mutation")
> plot(bayfit$fit[[k]]$beta.pp[[2]],xlab="Genomic region",ylab="Posterior probability",
+      main="Copy Number")
> plot(bayfit$fit[[k]]$beta.pp[[3]],xlab="Genes",ylab="Posterior probability",
+      main="Expression")
```

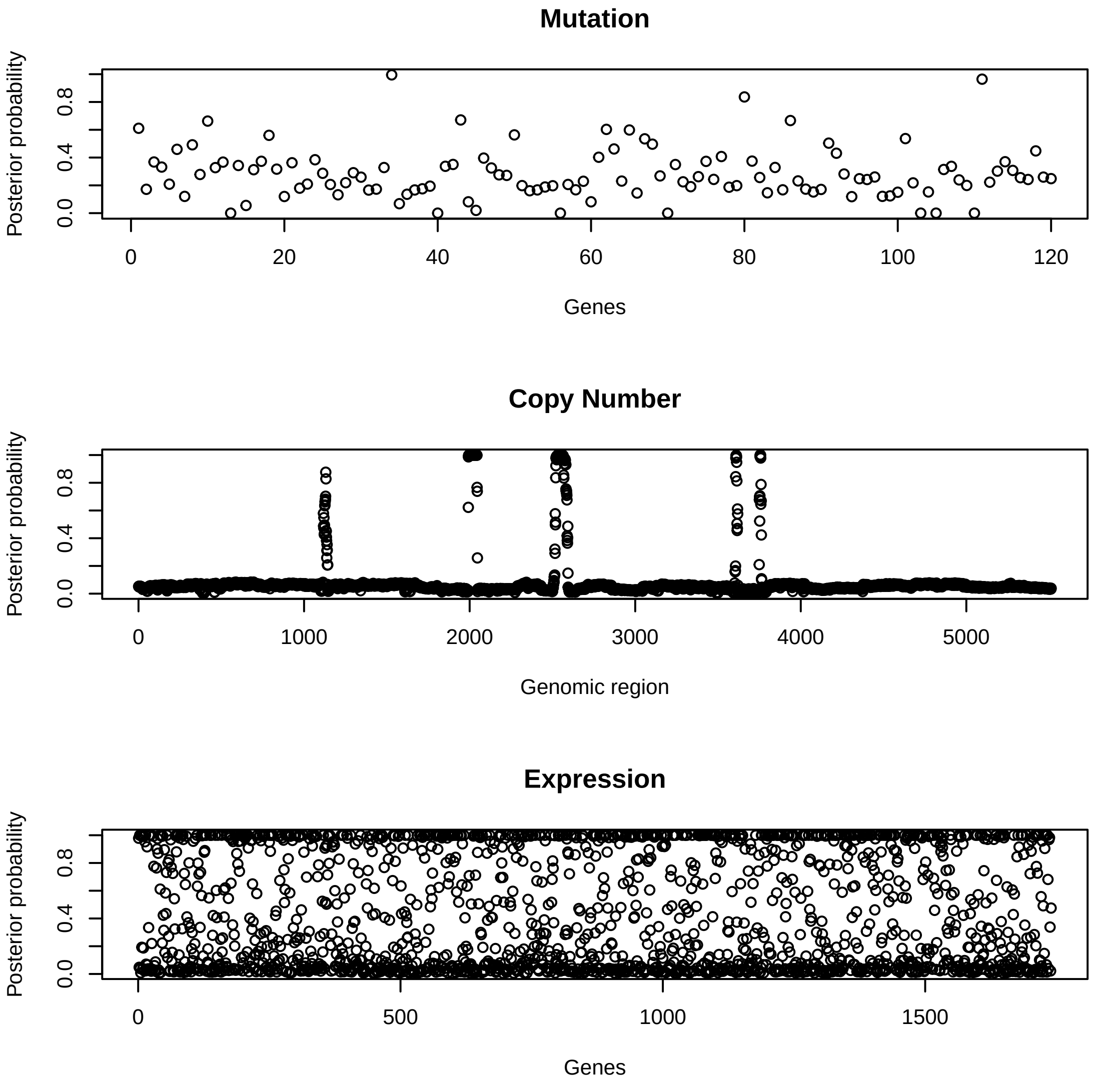

Figure 6: Posterior probabilities of genomic features

```
> plotHMBayes(fit=bayfit$fit[[3]],datasets=list(gbm.mut2,cn.image,exp.image),
+             type=c("binomial","gaussian","gaussian"), col.scheme = col.scheme,
+             threshold=c(0.5,0.5,0.5),row.order=c(F,F,T),chr=chr,plot.chr=c(F,T,F),
+             sparse=c(T,T,T),cap=c(F,T,F))

1   14
2   159
3   705
```

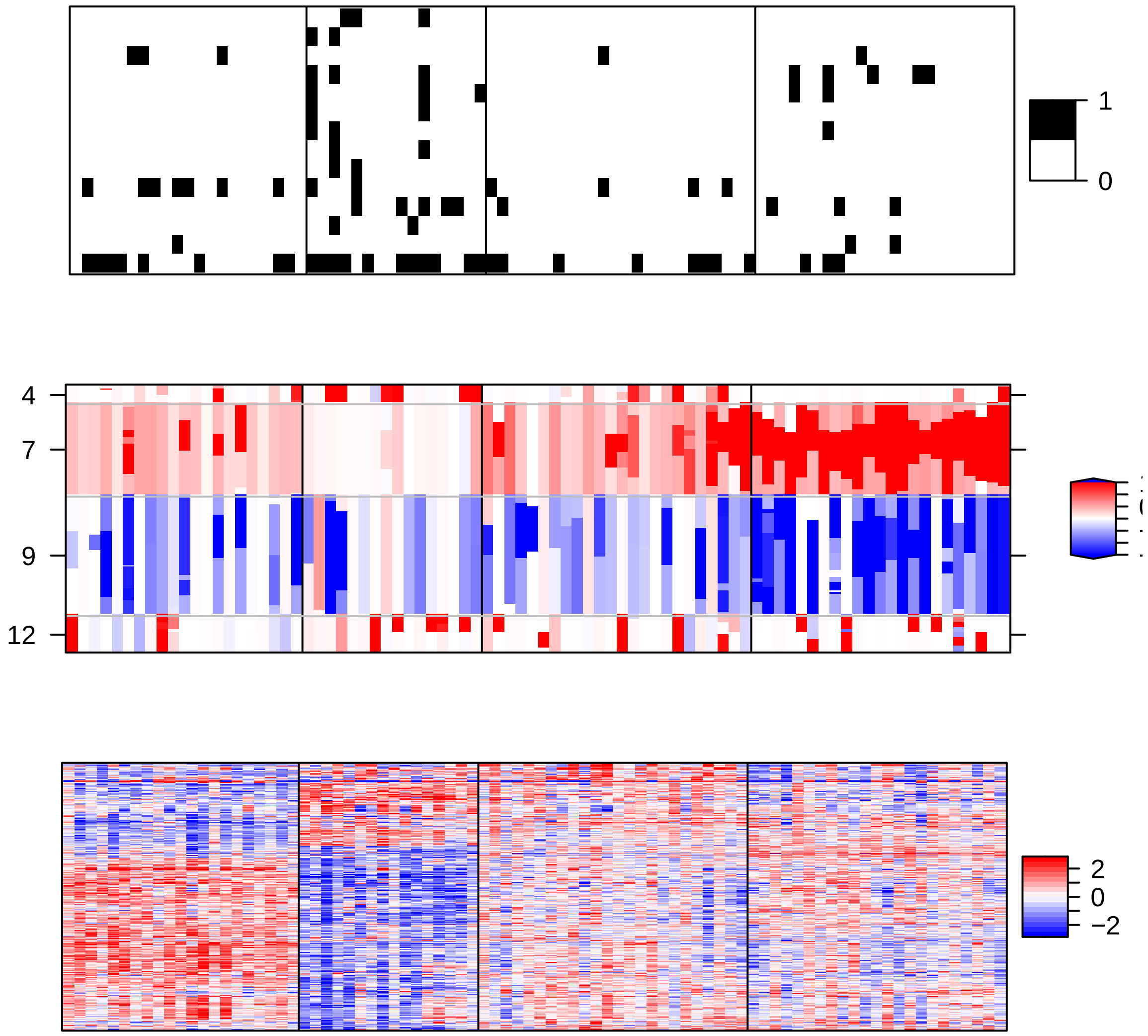

Figure 7: Heatmap of mutation (top panel), DNA copy number (middle panel), and mRNA expression (bottom panel) for the 4-cluster solution. Rows are genomic features and columns are samples.Genomic features with posterior probability > 0.5 are shown on the heatmap.

```
attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets 
[8] methods   base     

other attached packages:
[1] lattice_0.20-45      gplots_3.1.3         GenomicRanges_1.51.4
[4] GenomeInfoDb_1.35.15 IRanges_2.33.0       S4Vectors_0.37.3    
[7] BiocGenerics_0.45.0  iClusterPlus_1.35.1 

loaded via a namespace (and not attached):
 [1] zlibbioc_1.45.0        gtools_3.9.4           GenomeInfoDbData_1.2.9
 [4] KernSmooth_2.23-20     XVector_0.39.0         RCurl_1.98-1.10       
 [7] bitops_1.0-7           caTools_1.18.2         grid_4.3.0            
[10] compiler_4.3.0         tools_4.3.0           
```

## References

Qianxing Mo, Ronglai Shen, Cui Guo, Marina Vannucci, Keith S Chan, Susan G Hilsenbeck. (2018). A fully Bayesian latent variable model for integrative clustering analysis of multi-type omics data. Biostatistics 19(1):71-86.

Qianxing Mo, Sijian Wang, Venkatraman E. Seshan, Adam B. Olshen, Nikolaus Schultz, Chris Sander, R. Scott Powers, Marc Ladanyi, and Ronglai Shen. (2013). Pattern discovery and cancer gene identification in integrated cancer genomic data. Proc. Natl. Acad. Sci. USA 110(11):4245-50.

Ronglai Shen, Sijian Wang, Qianxing Mo. (2013). Sparse Integrative Clustering of Multiple Omics Data Sets. Annals of Applied Statistics 7(1) 269-294.

Ronglai Shen, Qianxing Mo, Nikolaus Schultz, Venkatraman E. Seshan, Adam B. Olshen, Jason Huse, Marc Ladanyi, Chris Sander (2012). Integrative Subtype Discovery in Glioblastoma Using iCluster. PLOS ONE 7(4):e35236.

Ronglai Shen, Adam B. Olshen, Marc Ladanyi (2009). Integrative clustering of multiple genomic data types using a joint latent variable model with application to breast and lung cancer subtype analysis. Bioinformatics 25(22):2906-12.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67, 301-320..

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 58, 267-288..

J. Friedman, T. Hastie, and R. Tibshirani, Regularized paths for generalized linear models via coordinate descent, Journal of statistical software 33(1).

TCGA Network, (2008). Comprehensive genomic characterization defines human glioblastoma genes and core pathways. Nature 455, 1061-1068..

K. Fang and Y. Wang, Number theoretic methods in statistics., Chapman abd Hall, London, UK, 1994.