moCluster: Integrative clustering using multiple omics data

Chen Meng

Modified: January 19, 2016. Compiled: January 22, 2016.

Contents

1 moCluster overview

2 Run moCluster

2.1 Quick start

Introduction to moCluter algorithm.

The moCluster method is described in [**?**]. In this vignette, we will define subtypes the NCI-60 cell lines using four transcriptomic data from different microarray platforms. First we load the package and data

```
# loading package and gene expression data
library(mogsa)
data(NCI60_4arrays)
```

`NCI60_4arrays` is a *list* of *data.frame*. The *list* consists of microarray data for NCI-60 cell lines from different platforms. In each of the *data.frame*, columns are the 60 cell lines and rows are genes. The data was downloaded from [**?**], but only a small subset of genes were selected. Therefore, the result in this vignette is not intended for biological interpretation. Check the dimension of each matrix:

```
sapply(NCI60_4arrays, dim) # check dimensions of expression data
```

```
##      agilent hgu133 hgu133p2 hgu95
## [1,]     300    298      268   288
## [2,]      60     60       60    60
```

Before performing the moCluster analysis, we define some auxiliary variable to indicate the tissue of origin of cell lines and the color for each:

```
tumorType <- sapply(strsplit(colnames(NCI60_4arrays$agilent), split="\\."), "[", 1)
colcode <- as.factor(tumorType)
levels(colcode) <- c("red", "green", "blue", "cyan", "orange",
                     "gray25", "brown", "gray75", "pink")
colcode <- as.character(colcode)
```

Then we can apply the moCluster algorithm to clustering the 60 cell lines. The moCluster employs consensus PCA (CPCA) approach to integrate multiple omics data. CPCA approach is particularly suitable for the integrative clustering algorithm. The First, we call `mbpca` to perform the consensus PCA:

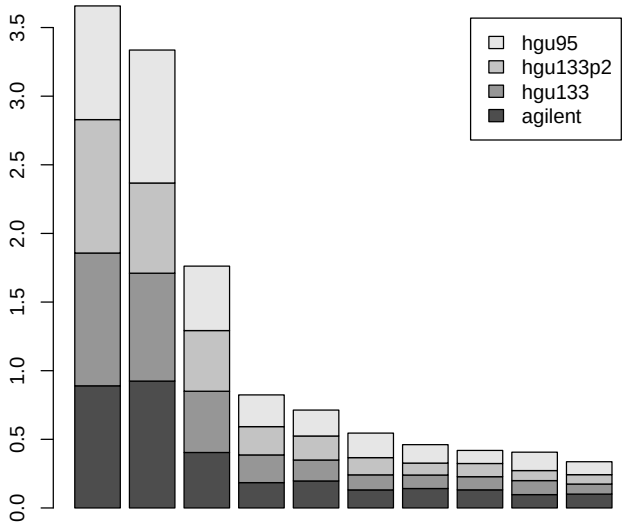


Figure 1: The variance associated with each latent variable. Colors distinguishes the contributions from different data sets.

```
moa <- mbpca(NCI60_4arrays, ncomp = 10, k = "all", method = "globalScore", option = "lambda1"
             center=TRUE, scale=FALSE, moa = TRUE, svd.solver = "fast", maxiter = 1000)
```

```
## calculating component 1 ...
## calculating component 2 ...
## calculating component 3 ...
## calculating component 4 ...
## calculating component 5 ...
## calculating component 6 ...
## calculating component 7 ...
## calculating component 8 ...
## calculating component 9 ...
## calculating component 10 ...
```

```
plot(moa, value="eig", type=2)
```

In the above commands, the argument `ncomp = 10` specifies that 10 latent variables should be calculated. The argument `k = "all"` suggest that no sparsity should be introduced to the loading factor. Sparsity in loading factor is a preferred property in multiple omics data analysis since only a subset of the variables would associated with the latent variables. Therefore, the interpret ability is increased. However, we do not introduce the sparsity for two reasons. First, we want to evaluate the relative contribution of variance as show in the figure. Second, we will use the permutation test to evaluate the latent variables that are representing the concordant structures in different data sets. This can be done with a permutation test, that is, permute the samples in each of the data and do the multi-block PCA:

```
r <- bootMbpca(moa, mc.cores = 1, B=20, replace = FALSE, resample = "sample")
```

```
## method is set to 'globalScore'.
```

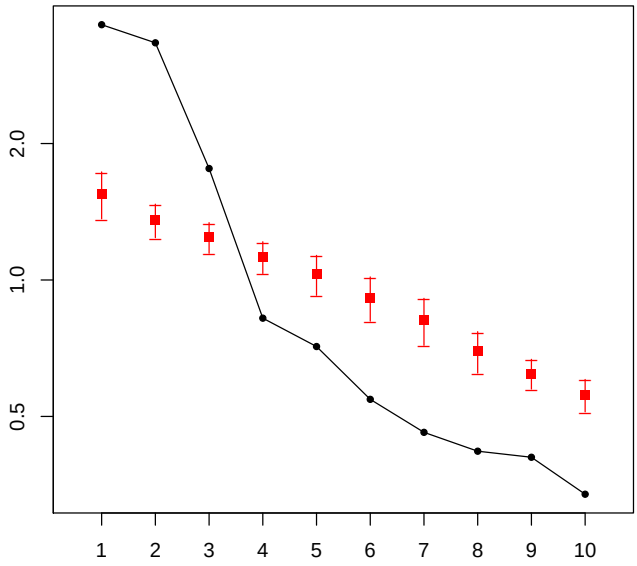


Figure 2: permutation test

The result suggested that the top 3 latent variables account for concordance structures across data data. The elbow test also shows that the top 3 variables are significant. Therefore, we will use 3 latent variable in the subsequent analysis. Next, we are ready to calculate the latent variables with the parse loadings:

```
moas <- mbpca(NCI60_4arrays, ncomp = 3, k = 0.1, method = "globalScore", option = "lambda1",
              center=TRUE, scale=FALSE, moa = TRUE, svd.solver = "fast", maxiter = 1000)
```

```
## calculating component 1 ...
## calculating component 2 ...
## calculating component 3 ...
```

`k = 0.1` indicates we only keep 10% variables with non-zero coefficients in the result. The cluster will be perform using the latent variable, which could be extract by `moaScore`. We first extract the latent variables for both the sparse and non-sparse loading results. Then compare their correlation.

```
scr <- moaScore(moa)
scrs <- moaScore(moas)
diag(cor(scr[, 1:3], scrs))
```

```
##       PC1       PC2       PC3
## 0.9741884 0.9889647 0.9546203
```

They have a very high correlation so we assume the should account for the same variance or biological effects. Visualize the plot in the first three dimensions.

```
layout(matrix(1:2, 1, 2))
plot(scrs[, 1:2], col=colcode, pch=20)
legend("topright", legend = unique(tumorType), col=unique(colcode), pch=20)
plot(scrs[, 2:3], col=colcode, pch=20)
```

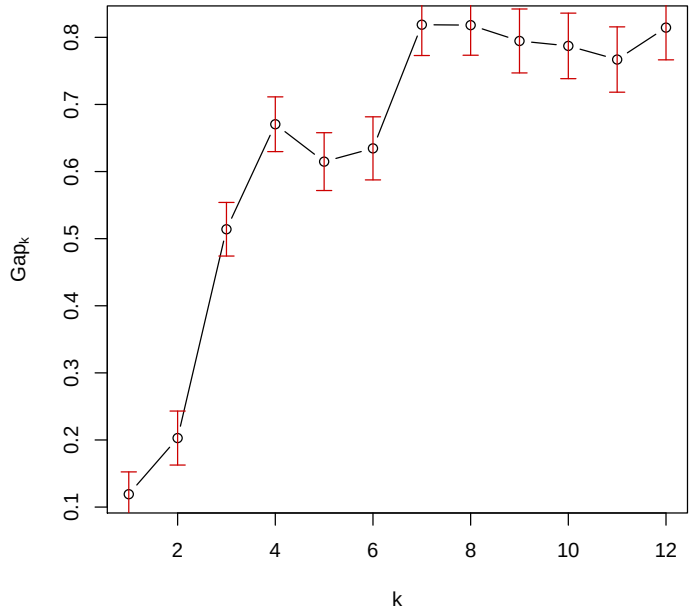
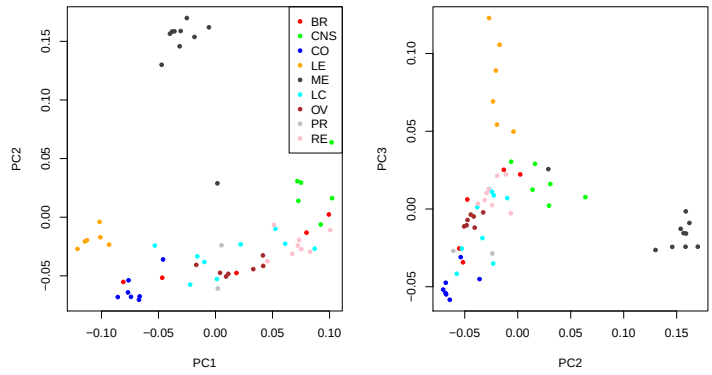


Figure 3: gap statistic plot



Using gap statistic to evaluate the optimal number of clusters

```
gap <- moGap(moas, K.max = 12, cluster = "hcl")
```

```
layout(matrix(1, 1, 1))
gap$nClust
```

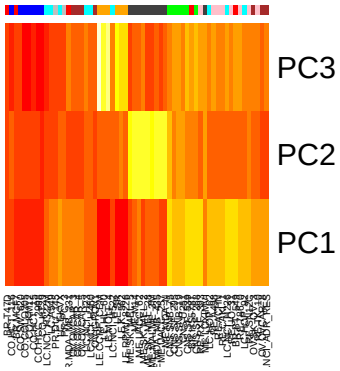
```
##    firstSEmax Tibs2001SEmax   globalSEmax      firstmax     globalmax
##             4              4             7             4             7
```

Using hierarchical cluster and plot

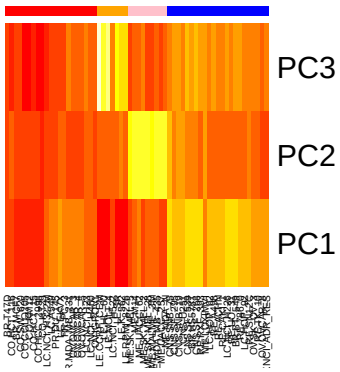
```
hcl <- hclust(dist(scrs))
cls <- cutree(hcl, k=4)
clsColor <- as.factor(cls)
```

```
levels(clsColor) <- c("red", "blue", "orange", "pink")
clsColor <- as.character((clsColor))

heatmap(t(scrs[hcl$order, ]), ColSideColors = colcode[hcl$order], Rowv = NA, Colv=NA)
```




```
heatmap(t(scrs[hcl$order, ]), ColSideColors = clsColor[hcl$order], Rowv = NA, Colv=NA)
```



In order to interpret the result, extract the variable with non-zero coefficients

```
genes <- moaCoef(moas)
genes$nonZeroCoef$agilent.V1.neg
```

```
##                              id         coef
## FGD3_agilent     FGD3_agilent -0.105242702
## TMC6_agilent     TMC6_agilent -0.045236957
## GMFG_agilent     GMFG_agilent -0.042502839
## IQGAP2_agilent IQGAP2_agilent -0.001185483
```

3 Session info

```
toLatex(sessionInfo())
```

- R version 3.2.3 (2015-12-10), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: knitr 1.12.3, mogsa 1.2.1
- Loaded via a namespace (and not attached): AnnotationDbi 1.32.3, Biobase 2.30.0, BiocGenerics 0.16.1, BiocStyle 1.8.0, DBI 0.3.1, GSEABase 1.32.0, IRanges 2.4.6, KernSmooth 2.23-15, RSQLite 1.0.0, S4Vectors 0.8.9, XML 3.98-1.3, annotate 1.48.0, bitops 1.0-6, caTools 1.17.1, cluster 2.0.3,

codetools 0.2-14, corpcor 1.6.8, digest 0.6.9, evaluate 0.8, formatR 1.2.1, gdata 2.17.0, genefilter 1.52.0, gplots 2.17.0, graph 1.48.0, graphite 1.16.0, gtools 3.5.0, highr 0.5.1, magrittr 1.5, parallel 3.2.3, rappdirs 0.3, splines 3.2.3, stats4 3.2.3, stringi 1.0-1, stringr 1.0.0, survival 2.38-3, svd 0.3.3-2, tools 3.2.3, xtable 1.8-0