iClusterPlus

Setup

This is a tutorial from the iClusterPlus package (see iClusterPlus::iManual() for details).

First we call the relevant libraries.

```r
# The packages available in CRAN
required_packages <- c(
  "BiocManager",
  "cluster",
  "gplots",
  "lattice",
  "dplyr"
)

# The packages available from Bioconductor
bioconductor_packages <- c(
  "iClusterPlus",
  "GenomicRanges",
  "DNAcopy"
)

# Check if the packages are avilable - if not install them
for (pkg in required_packages) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    install.packages(pkg)
  }
}

for (b_pkg in bioconductor_packages) {
  if (!requireNamespace(b_pkg, quietly = TRUE)) {
    BiocManager::install(b_pkg)
  }
}

# Load the required libraries
library(iClusterPlus)
library(GenomicRanges)
library(DNAcopy)
library(cluster) # used for finding medoids in the adaptive model fit
library(gplots)
library(lattice)
library(dplyr)
```

This tutorial uses the glioblastoma data from the Cancer Genome Atlas (TGCA) (Network and others 2008).

```r
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.

```
# 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.

```
dim(gbm.seg)
```

```
## [1] 16295     6
```

```
gbm.seg[1:3, ]
```

```
##         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 is the segmentation results produced by DNAcopy
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,
  nPobes = 5
)
```

```
## Removing CNV...
```

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 (the medoid is similar to the centroid of a dataset, but restricted to a point within the dataset). 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.

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

```
## Removing CNV...
```

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
```

Integrative clustering analysis

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 sparsity). Normally, we have to optimize k and lambda through model tuning which we discuss in the next section.

```
n_clusters <- 3
n_iter <- 10
```

```r
data_types <- c("binomial", "gaussian", "gaussian")
lambdas <- c(0.04, 0.61, 0.90)

fit.single <- iClusterPlus(
  dt1 = gbm.mut2,
  dt2 = gbm.cn,
  dt3 = gbm.exp,
  type = data_types,
  lambda = lambdas,
  K = n_clusters - 1,
  maxiter = n_iter
)
```

Model tuning

Using parallel computing, tune.iClusterPlus samples a series of lambda values from the parameter space based on the Uniform design (Fang 1994) 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.

We have a number of options for our choice of `n.lambda` depending on the number of data.types present.

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

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 the percentage of explained variation. For general practice, set `n.lambda=NULL` to use the default value.

```r
# This is slow. Setting to not evaluate in Markdown.

# Set seed for consistent results.
set.seed(1)

# iterate over k
K <- 5
for (k in 1:K) {
  cv.fit <- tune.iClusterPlus(
    cpus = 3,
    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 = ""))
}
```

Model selection

An illustration of model tuning. To select the vector λ of penalty parameters iClusterPlus iterates over the number of latent variables defined by k. Then the Bayesian information criterion (BIC) is used to select the optimal sparse model with regards to combinations of penalty parameters. To optimise k, the deviance ratio is the metric of choice. For a given model defined by parameters θ_{fit}, the deviance ratio is defined by the saturated model (the model for which there is a unique parameter for every observation) defined by θ_{sat}, and the null model (the intercept only model) defined by θ_{null}:

$$Dev(\theta_{fit}) = \frac{l(\theta_{fit}) - l(\theta_{null})}{l(\theta_{sat}) - l(\theta_{null})}.$$

This ratio can be interpreted as the percentage of explained variation. Using a plot of the deviance ratio a function of k, the value of k for which the deviance ratio levels off is selected.

```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)
```

From this one obtains the ID for the lambda vector at which the BIC is minimised. Then we obtain the corresponding deviance ratio.

```r
minBICid <- apply(BIC, 2, which.min)
devRatMinBIC <- rep(NA, nK)
for (i in 1:K) {
  devRatMinBIC[i] <- devR[minBICid[i], i]
}
```

We plot the deviance ratio as a function of the number of clusters $(k+1)$.

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

From this an optimal value of $k = 2$ (i.e. $n_{clusters} = 3$) is selected.

```r
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])]]
```

For noisy data a curve as shown above is unlikely - the choice of k will probably be a lot less clear. In this case it is recommended that one creates heat maps of the datasets and decide the optimal number of clusters based upon the features' pattern.

References

Fang, KT. 1994. "Uniform Design and Uniform Design Tables, Sci." *Press, Beijing, China.*

Network, Cancer Genome Atlas Research, and others. 2008. "Comprehensive Genomic Characterization Defines Human Glioblastoma Genes and Core Pathways." *Nature* 455 (7216). Nature Publishing Group: 1061.