Subclustering Strategy (2plus7)

Cody Ashby

8/25/2017

The purpose of this document is to show the subclustering strategy.

The premise is that the classical HRD (HRD) samples overshadow the other (nHRD) samples and cause us to miss interesting features hidden in the data.

#read in file with the copy number data for the 29 features  
all.dat = read.table("copy\_table\_29features.txt", header=T, sep="\t", stringsAsFactors = F, row.names=1)  
#count loss of heterozygosity as a deletion  
all.dat[all.dat == -2] = 1  
#spread values for more distinctness between del/gain and normal  
all.dat[all.dat == 1] = 0.5  
all.dat[all.dat == 3] = 3.5  
all.dat.unscaled = all.dat  
#scale and center data  
all.dat <- scale(all.dat,center = T,scale = T)  
#colors  
mycolors = rev(c("#7E0000","#d5322f", "#bdbdbd", "#377eb8", "#001852"))

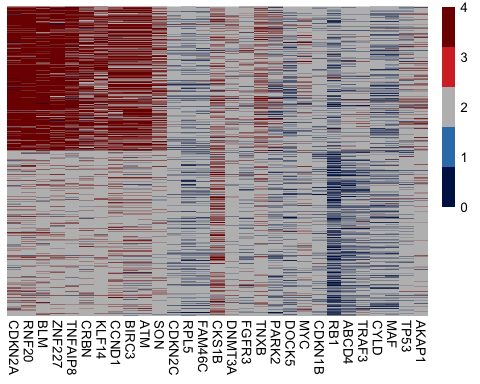
We're going to determine the optimal number of clusters in the data using sigclust. so I need to define a method to check all pairwise comparisons of clusters. I will also run the initial kmeans with k=2 to split the HRD and nHRD samples.

#writing this to run in parallel  
iterate\_sigclust\_sub = function(i, j, x, cluster, nsim=1000)  
{  
 if(i != j)  
 {  
 subcluster = x[rownames(x) %in% names(cluster$cluster[cluster$cluster %in% c(i,j)]),]  
 cluster\_labels = mapvalues(cluster$cluster[cluster$cluster %in% c(i,j)], from = c(i,j), to = c(1,2))  
 set.seed(1)  
 pvalue <- sigclust(subcluster, nsim=nsim, nrep = 1, labflag = 1, label = cluster\_labels, icovest = 2)  
 results = pvalue@pval  
 }  
 else  
 {  
 results = 1  
 }  
 results  
}  
  
iterate\_sigclust = function(x, cluster, nsim=1000)  
{  
 cluster\_num = max(cluster$cluster)  
 vec\_i = c()  
 vec\_j = c()  
 for(i in 1:cluster\_num)  
 {  
 for(j in 1:cluster\_num)  
 {  
 vec\_i = c(vec\_i,i)  
 vec\_j = c(vec\_j,j)  
 }  
 }  
 results = pbmcmapply(vec\_i, FUN=iterate\_sigclust\_sub, mc.cores = 8, MoreArgs = list(x=x, cluster=cluster, nsim=nsim), j=vec\_j)  
 matrix(results, nrow=cluster\_num, ncol=cluster\_num)  
}  
  
#run for k = 2  
set.seed(1)  
all.dat.cluster = kmeans(all.dat, 2, nstart = 30, iter.max = 1000)  
iterate\_sigclust(all.dat, all.dat.cluster)

## [,1] [,2]  
## [1,] 1 0  
## [2,] 0 1

So, they’re significantly different from each other. So we go ahead and split the samples.

set.seed(1)  
all.dat.cluster = kmeans(all.dat, 2, nstart = 30, iter.max = 2000)  
all.dat.cluster.order = names(all.dat.cluster$cluster[order(all.dat.cluster$cluster)])  
pheatmap(all.dat.unscaled[match(all.dat.cluster.order, rownames(all.dat)),], cluster\_rows = F, cluster\_cols = F, labels\_row = "", color=mycolors)



#in this case cluster 1 is the HRD and cluster 2 is the nHRD  
all.dat.hrd = all.dat[match(names(all.dat.cluster$cluster[all.dat.cluster$cluster == 1]), rownames(all.dat)),]  
all.dat.nhrd = all.dat[match(names(all.dat.cluster$cluster[all.dat.cluster$cluster == 2]), rownames(all.dat)),]

Next, we're going to determine the optimal number of clusters using sigclust. We begin at k=2 and increase k by one until the clusters are no longer significantly different. **Note: I've retrospectively updated this document for where the clusters fail: HRD(2-3), nHRD(2-8).** **The output from these loops will be 0 if it’s significant and >0 if it is not.** You can see that the HRD samples fail when k=3 and the nHRD samples fail when k=8. We’ll also check the combined cluster for significance as well.

#check for significance  
for(i in 2:3)  
{  
 set.seed(1)  
 all.dat.hrd.cluster = kmeans(all.dat.hrd, i, nstart = 30, iter.max = 2000)  
 r = iterate\_sigclust(all.dat.hrd, all.dat.hrd.cluster, 1000)  
 r = sum(r > 0.05) - i  
 print(paste("With k = ", i," : ",r, sep=""))  
}

## [1] "With k = 2 : 0"  
## [1] "With k = 3 : 2"

#check for significance  
for(i in 2:8)  
{  
 set.seed(1)  
 all.dat.nhrd.cluster = kmeans(all.dat.nhrd, i, nstart = 30, iter.max = 2000)  
 r = iterate\_sigclust(all.dat.nhrd, all.dat.nhrd.cluster, 1000)  
 r = sum(r > 0.05) - i  
 print(paste("With k = ", i," : ",r, sep=""))  
}

## [1] "With k = 2 : 0"  
## [1] "With k = 3 : 0"  
## [1] "With k = 4 : 0"  
## [1] "With k = 5 : 0"  
## [1] "With k = 6 : 0"  
## [1] "With k = 7 : 0"  
## [1] "With k = 8 : 2"

set.seed(1)  
all.dat.hrd.cluster = kmeans(all.dat.hrd, 2, nstart = 30, iter.max = 2000)  
set.seed(1)  
all.dat.nhrd.cluster = kmeans(all.dat.nhrd, 7, nstart = 30, iter.max = 2000)

all.dat.subcluster = all.dat.nhrd.cluster$cluster  
all.dat.subcluster = mapvalues(all.dat.subcluster, 1:7, 3:9)  
all.dat.subcluster = c(all.dat.subcluster, all.dat.hrd.cluster$cluster)  
all.dat.subcluster = as.data.frame(all.dat.subcluster)  
colnames(all.dat.subcluster) = "clustering"  
test = list(clustering = all.dat.subcluster$clustering)  
names(test$clustering) = rownames(all.dat.subcluster)  
all.dat.subcluster = test  
iterate\_sigclust(all.dat, all.dat.subcluster)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]  
## [1,] 1.000 0.027 1.000 1.000 1.000 1.000 1.000 1.000 1.000  
## [2,] 0.027 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000  
## [3,] 1.000 1.000 1.000 0.024 0.005 0.001 0.008 0.000 0.000  
## [4,] 1.000 1.000 0.024 1.000 0.001 0.001 0.003 0.000 0.003  
## [5,] 1.000 1.000 0.005 0.001 1.000 0.013 0.002 0.000 0.006  
## [6,] 1.000 1.000 0.001 0.001 0.013 1.000 0.001 0.000 0.002  
## [7,] 1.000 1.000 0.008 0.003 0.002 0.001 1.000 0.023 0.005  
## [8,] 1.000 1.000 0.000 0.000 0.000 0.000 0.023 1.000 0.002  
## [9,] 1.000 1.000 0.000 0.003 0.006 0.002 0.005 0.002 1.000

As you can see from the previous output, the subclusters from the HRD are **not** significantly different than the subclusters from the nHRD group. This is exactly the situation we were trying to avoid by clustering them seperately, so this makes sense. Finally, let's combine everything and output the final cluster groups.

z = all.dat.unscaled  
z[z==0.5] = 1  
z[z==3.5] = 3  
mygaps = cumsum(table(all.dat.subcluster$clustering))  
pheatmap(z[match(names(test$clustering[order(all.dat.subcluster$clustering)]), rownames(z)),], cluster\_rows = F, cluster\_cols = F, labels\_row = rep("",nrow(all.dat)), color=mycolors, gaps\_row = mygaps)

