stablize

October 23, 2017

In [1]:
```
library(Rcpp)
library(dplyr)
library(foreach)
library(doParallel)
library(parallel)
library(iterators)
library(microbenchmark)
```

```
Warning message:
package dplyr was built under R version 3.4.1
Attaching package: dplyr

The following objects are masked from package:stats:

    filter, lag

The following objects are masked from package:base:

    intersect, setdiff, setequal, union

Loading required package: iterators
Loading required package: parallel
```

In [102]:
```
#Set working directory
getwd()
load("data/lingBinary.RData")
```

'/accounts/grad/yizhou_zhao/Stat215'

In [103]:
```
#load cpp file
cpp_directory = ("/accounts/grad/yizhou_zhao/Stat215")
sourceCpp(file.path(cpp_directory, 'cor_sim.cpp'))
```

In [104]:
```
#Time comparison between R and Cpp

genMatrix <- function(A){
    #Function: From cluster to generate matrix
```

```
    #Input:
    #    A: a vector of indicators for cluster
    #Output:
    #    Cluster matrix
    l = length(A)
    M = matrix(0,nrow = l,ncol = l)
    for(i in 1:l){
       for(j in i:l){
           if (A[i] == A[j]){
               if (i != j){
                   M[i,j] = 1
                   M[j,i] = 1
               }
           }
       }
    }
    return(M)
}

dotProduct <- function(M1,M2){
    #Function: get the dot product of two cluster matrixs
    #Input:
    #   Two square matrixs of the same dimension
    #Output:
    #   dot product of the two matrixs
    if(length(M1) != length(M2)){stop("Error:wrong dimensions!")}
    sum = 0
    l = length(M1[1,])
    for(i in 1:l){
        for(j in 1:l){
            #cat(i,j)
            #cat(i,j,M1[i,j],M2[i,j],"\n")
            sum = sum + M1[i,j]*M2[i,j]
        }
    }
    return(sum)
}



matchCoeff_R <- function(M1,M2){
    # Function: get the matching coefficient of two cluster matrixs
    # Input:
    #     Two cluster matrixs
    # Ouput:
    #     matching coefficient \in [0,1]
     if(length(M1) != length(M2)){stop("Error:wrong dimensions!")}
    return(1-1/length(M1)*dotProduct(M1-M2,M1-M2))
```

```
}

jaccardCoeff_R <- function(M1,M2){
    if(length(M1) != length(M2)){stop("Error:wrong dimensions!")}
    return (dotProduct(M1,M2)/(dotProduct(M1,M1)+dotProduct(M2,M2)-dotProduct(M1,M2)))
}

corSim_R <- function(M1,M2){
    if(length(M1) != length(M2)){stop("Error:wrong dimensions!")}
    return (dotProduct(M1,M2)/sqrt(dotProduct(M1,M1)*dotProduct(M2,M2)))
}
```

In [105]:
```
#Random sample example
A = sample(1:5, 100, replace = TRUE)
B = sample(1:5, 100, replace = TRUE)
M1  = genMatrix(A)
M2 = genMatrix(B)

mb = microbenchmark(
    dotProduct(M1,M2),
    matchCoeff_R(M1,M2),
    corSim_R(M1,M2),
    jaccardCoeff_R(M1,M2),
    sumProduct(A,B),
    matchCoeff(A,B),
    corSim(A,B),
    jaccardCoeff(A,B)
)

#Comparison between cpp(improved version) and R for getting the product
mb %>%  group_by(expr) %>% summarise(mean_time = mean(time))
```

expr	mean_time
dotProduct(M1, M2)	2973985.6
matchCoeff_R(M1, M2)	3064470.0
corSim_R(M1, M2)	8400983.2
jaccardCoeff_R(M1, M2)	11466100.4
sumProduct(A, B)	64560.8
matchCoeff(A, B)	277244.8
corSim(A, B)	147763.2
jaccardCoeff(A, B)	187254.0

In [4]:
```
stabalize <- function(Df,k,N,ratio,method = "corSim"){
    #Funition to get the stablized K
    #Input:
    # Df: data; K: number of clusters; N: number of iterations;
    #           ratio: sampling ratio; method = "corSim","matchingCoeff" or "jaccardCoeff"
    similarityList = vector()
      for(i in 1:N){
```

```
            sample1 = sample_frac(Df,size = ratio, replace = F)
            sample2 = sample_frac(Df,size = ratio, replace = F)
            k.means1 = kmeans(sample1[7:ncol(Df)], centers = k, iter.max = 2000)$cluster
            k.means2 = kmeans(sample2[7:ncol(Df)], centers = k, iter.max = 2000)$cluster

            id <- as.vector(t(arrange(inner_join(sample1, sample2, by= 'ID') %>% select(ID))))
            id = as.factor(id)

            if(method == "corSim"){
              similarityList[i] = corSim(k.means1[id],k.means2[id])
            }
            else if(method == "matchCoeff"){
              similarityList[i] = matchCoeff(k.means1[id],k.means2[id])
            }
            else if(method == "jaccardCoeff"){
              similarityList[i] = jaccardCoeff(k.means1[id],k.means2[id])
            }
          }

        colName = paste0("cluster_",as.character(k))
        return(data.frame(colName = similarityList))
      }
```

In [5]:
```
nCores <- detectCores()
registerDoParallel(nCores)
```

In [11]:
```
k.max = 10
num.samples = 100
sample.ratio = 0.7
```

In [7]:
```
story = data.frame(rownum = 1:num.samples)
```

In [ ]:
```
parallel.results <- foreach(i = 2:k.max) %dopar% { #will specify k.max
  # Make sure that each process has its own random number stream.
  set.seed(i)
  column <- stabalize(lingBinary, i, num.samples, sample.ratio,method = "jaccardCoeff")
  filename <- paste0("cluster",i,".csv")
  write.csv(column, file=filename,
                quote=F, row.names=F)
  return(filename)
}
```

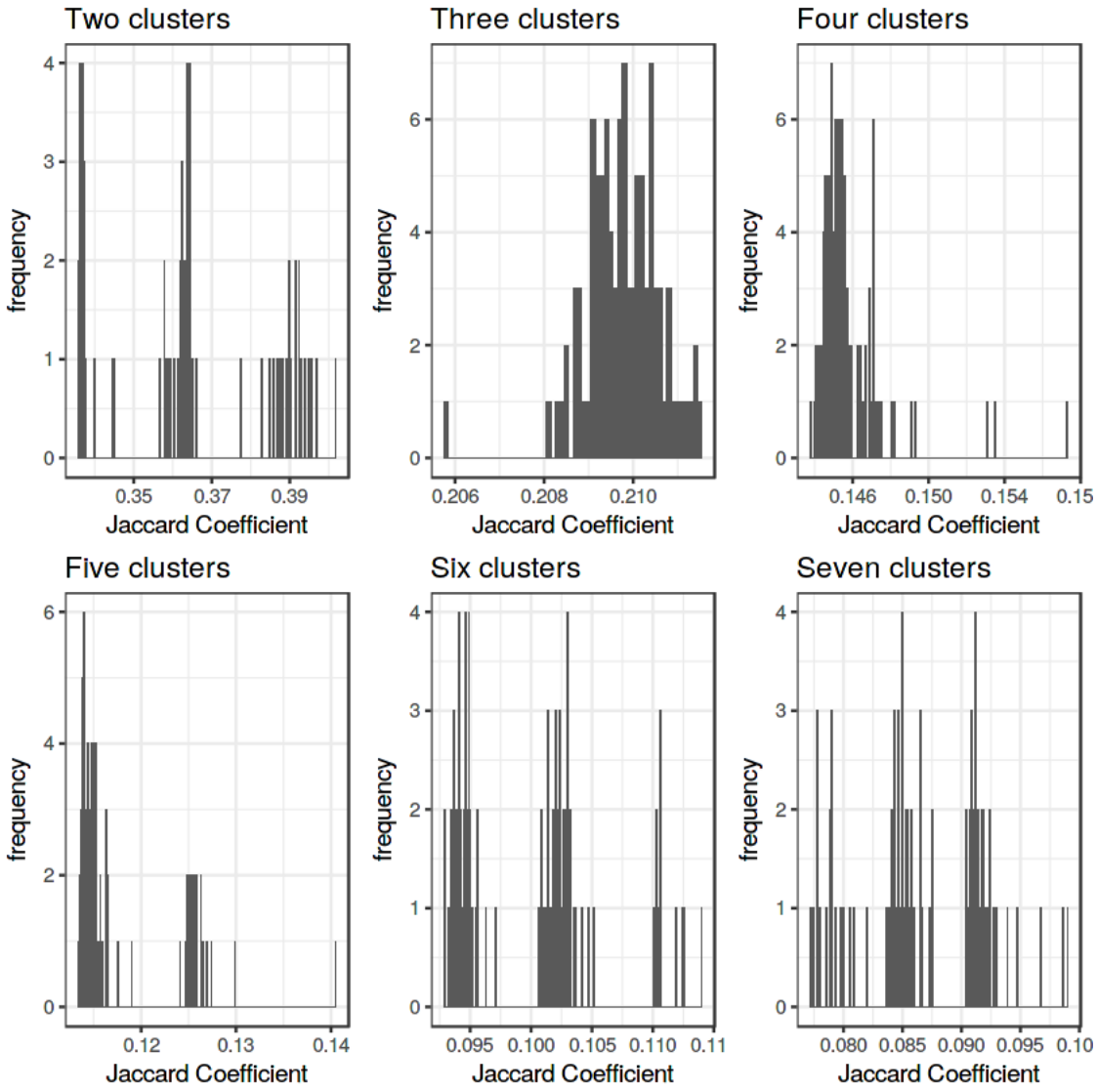
---

1 Plot

In [50]:
```
library(ggplot2)
library(gridExtra)
library(grid)
```

```
Attaching package: gridExtra

The following object is masked from package:dplyr:

    combine
```

```
In [52]: c2 = read.csv("cluster2.csv")
         p2 = ggplot(c2, aes(colName)) +
           geom_histogram(binwidth = 0.0002)+
           theme_bw() + ggtitle("Two clusters")+
           xlab("Jaccard Coefficient") + ylab("frequency")

In [53]: c3 = read.csv("cluster3.csv")
         p3 = ggplot(c3, aes(colName)) +
           geom_histogram(binwidth = 0.0001)+
           theme_bw() + ggtitle("Three clusters")+
           xlab("Jaccard Coefficient") + ylab("frequency")

In [54]: c4 = read.csv("cluster4.csv")
         p4 = ggplot(c4, aes(colName)) +
           geom_histogram(binwidth = 0.0001)+
           theme_bw() + ggtitle("Four clusters")+
           xlab("Jaccard Coefficient") + ylab("frequency")

In [55]: c5 = read.csv("cluster5.csv")
         p5 = ggplot(c5, aes(colName)) +
           geom_histogram(binwidth = 0.0001)+
           theme_bw() + ggtitle("Five clusters")+
           xlab("Jaccard Coefficient") + ylab("frequency")

In [56]: c6 = read.csv("cluster6.csv")
         p6 = ggplot(c6, aes(colName)) +
           geom_histogram(binwidth = 0.0001)+
           theme_bw() + ggtitle("Six clusters")+
           xlab("Jaccard Coefficient") + ylab("frequency")

In [57]: c7 = read.csv("cluster7.csv")
         p7 = ggplot(c7, aes(colName)) +
           geom_histogram(binwidth = 0.0001)+
           theme_bw() + ggtitle("Seven clusters")+
           xlab("Jaccard Coefficient") + ylab("frequency")

In [58]: grid.arrange(p2, p3, p4, p5, p6, p7, ncol=3)
```




```
In [64]: c8 = read.csv("cluster8.csv")
         c9 = read.csv("cluster9.csv")
         c10 = read.csv("cluster10.csv")
         cumudata = data.frame(c2,c3,c4,c5,c6,c7,c8,c9,c10)

In [76]: colnames(cumudata) = paste0(rep("cluster_",9), as.character(2:10))

In [78]: head(cumudata)
```

cluster_2	cluster_3	cluster_4	cluster_5	cluster_6	cluster_7	cluster_8	cluster_9	cluste
0.3627151	0.2108152	0.1456701	0.1158538	0.09455061	0.09131873	0.06809398	0.06737360	0.0600
0.3923200	0.2104748	0.1441307	0.1159966	0.09388517	0.09121403	0.07930914	0.06640034	0.0577
0.3641634	0.2107983	0.1450697	0.1134462	0.09479801	0.07860500	0.07377549	0.06320178	0.0573
0.3588959	0.2097471	0.1469255	0.1152332	0.10265389	0.07892100	0.06725492	0.06671670	0.0601
0.3652601	0.2089429	0.1471707	0.1139579	0.09353496	0.08525697	0.07452504	0.06249107	0.0579
0.3876530	0.2097501	0.1447580	0.1134527	0.11058405	0.09468781	0.07869207	0.06620155	0.0602

In [72]:

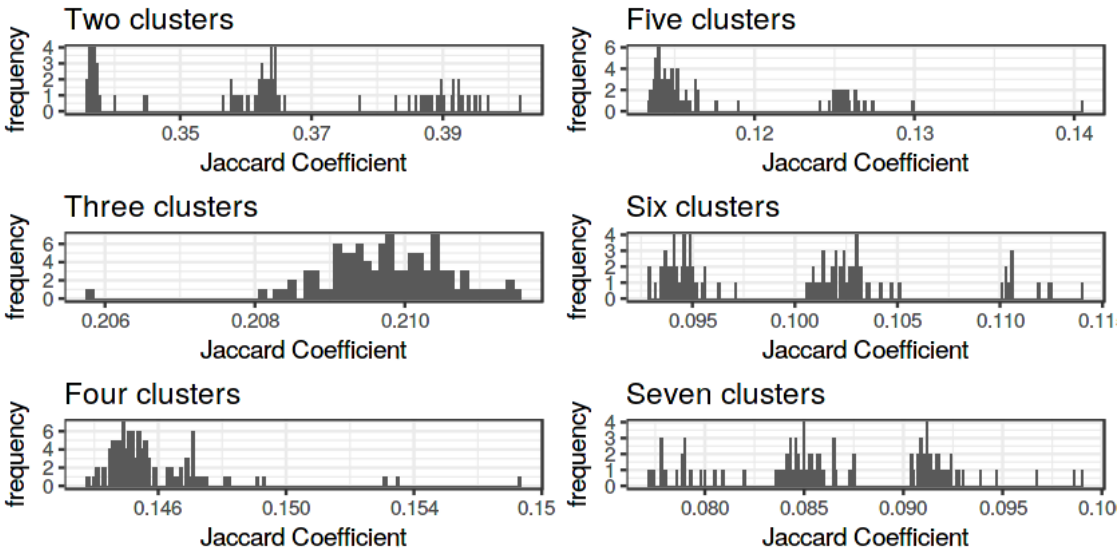
1. 'cluster_1' 2. 'cluster_2' 3. 'cluster_3' 4. 'cluster_4' 5. 'cluster_5' 6. 'cluster_6' 7. 'cluster_7' 8. 'cluster_8' 9. 'cluster_9' 10. 'cluster_10'

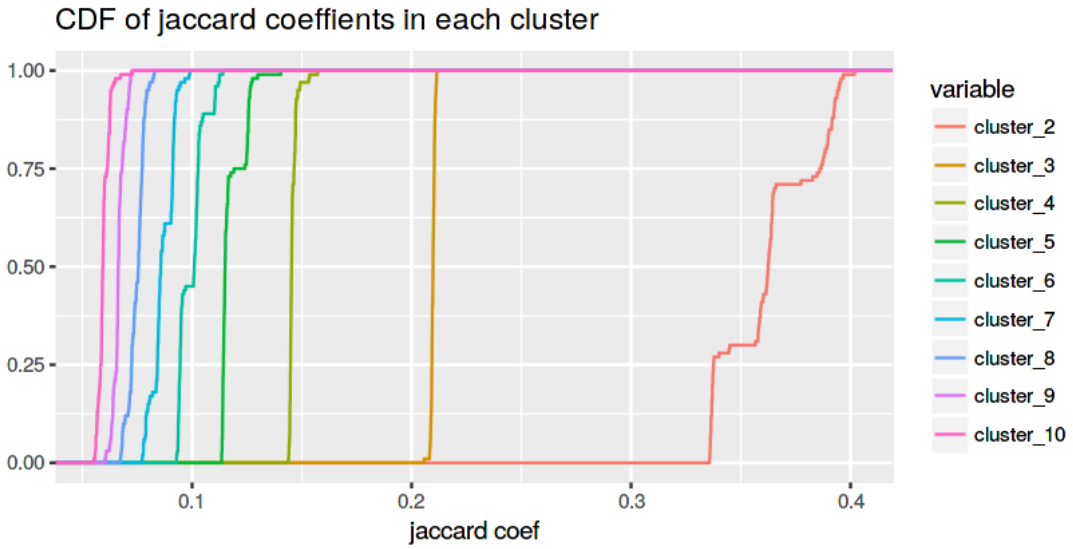
In [99]:
```
require(reshape2)
plotdata = melt(cumudata)

# Multiple ECDFs
pf = ggplot(plotdata, aes(value, colour = variable)) + stat_ecdf() +
     ggtitle("CDF of jaccard coeffients in each cluster") + xlab("jaccard coef") + ylab
```

```
No id variables; using all as measure variables
```

In [100]:
```
lay = cbind(c(1,2,3,7,7,7),c(4,5,6,7,7,7))
```

In [101]:
```
grid.arrange(p2, p3, p4, p5, p6, p7, pf, layout_matrix = lay)
```


Two clusters
frequency
Jaccard Coefficient
Five clusters
frequency
Jaccard Coefficient
Three clusters
frequency
Jaccard Coefficient
Six clusters
frequency
Jaccard Coefficient
Four clusters
frequency
Jaccard Coefficient
Seven clusters
frequency
Jaccard Coefficient


CDF of jaccard coeffients in each cluster
1.00
0.75
0.50
0.25
0.00
0.1
0.2
0.3
0.4
jaccard coef
variable
cluster_2
cluster_3
cluster_4
cluster_5
cluster_6
cluster_7
cluster_8
cluster_9
cluster_10