CBNA: A control theory based method for identifying coding and non-coding cancer drivers

Vu VH Pham[1], Lin Liu[1], Cameron P Bracken[2,3], Gregory J Goodall[2,3], Qi Long[4], Jiuyong Li[1] and Thuc D Le[1]

[1] School of Information Technology and Mathematical Sciences, University of South Australia, Mawson Lakes, 5095, Australia

[2] Centre for Cancer Biology, an alliance of SA Pathology and University of South Australia, Adelaide, SA 5000, Australia

[3] Department of Medicine, The University of Adelaide, Adelaide, SA 5005, Australia

[4] Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104, USA

This is the script to run methods other than the proposed method.

1. DawnRank

This is the script to run DawnRank.

To run the script, please prepare below input files and reset environment variables in the script.

The input files include the followings and are put in the folder "rootDir/Data" (rootDir is an environment variable):

- gadj.rda - Pathway data
- BRCA_matchedData_full.RData - Tumour expression data
- BRCA_matchedData_normal_samples_full.RData - Normal expression data
- mut.RData - Mutation data
- Census_allFri Sep 28 07_39_37 2018.tsv - Cancer Gene Census (CGC)

```
#=================================================================
#=================================================================
# DawnRank
#
# Reference:
# J. P. Hou and J. Ma. Dawnrank: discovering personalized driver genes in cancer.
# Genome Medicine, 6(7):56, 2014.
#
# Here is the link for DawnRank package:
# https://github.com/MartinFXP/DawnRank
#=================================================================
#=================================================================
# Clear the environment
rm(list = ls())

# Load libraries
# install.packages("devtools")
# library(devtools)
# install_github("MartinFXP/DawnRank")
```

```r
library(DawnRank)
library(CancerSubtypes)
library(clusterProfiler)
library(snowfall)

#----------------------------------------
# Set environment variables here if any
# Please remember to create necessary folders
rootDir <- "C:/Users/phavy022/MyDoc/05CancerDriver" # And put the input files in "rootDir/Data"
outDir <- "C:/Users/phavy022/MyDoc/05CancerDriver/DawnRank" # Output folder
topk_mR <- 6000 # Number of mRNAs selected for analysis
#----------------------------------------

# Functions if any
#========================================
#' This function allows you to select mRNAs
#' which have the most variant Median Absolute Deviation (MAD).
#' @param matchedData Expression data of mRNAs.
#' @param topk_mR Number of mRNAs to be cut off.
#' @return A list containing selected mRNAs. The list of elements includes:
#' \item{d} {The data with rows being samples and columns being mRNAs.}
#' \item{mRs} {The names of mRNAs.}
#========================================
getDatabyMAD <- function(matchedData, topk_mR) {
  # Identify significant mRNAs by using function FSbyMAD in CancerSubtypes package
  mRNAsData = FSbyMAD(t(matchedData$mRNAs), value=topk_mR)
  mRNAsData <- t(mRNAsData)

  # Remove duplicated data
  mRNAsData <- mRNAsData[,!duplicated(colnames(mRNAsData))]

  # Prepare the result
  mRs <- colnames(mRNAsData)
  l = list(d = mRNAsData, mRs = mRs)

  return(l)
}

#========================================
# Main script
#========================================
# Load the pathway data
load(paste(rootDir, "/Data/gadj.rda", sep = ""))

# Load the normal expression data
load(paste(rootDir, "/Data/BRCA_matchedData_normal_samples_full.RData", sep = ""))

# Load the tumor expression data
load(paste(rootDir, "/Data/BRCA_matchedData_full.RData", sep = ""))

# Combine data
combinedData <- BRCA_matchedData_normal_samples
combinedData$mRNAs <- rbind(combinedData$mRNAs, BRCA_matchedData$mRNAs)
```

```r
# Get significant mRNAs
l <- getDatabyMAD(combinedData, topk_mR)

# Get the normal data
mRNAsData_Normal <- t(BRCA_matchedData_normal_samples$mRNAs[,l$mRs])

# Get the cancer data
mRNAsData <-  t(BRCA_matchedData$mRNAs[,l$mRs])

# Load the mutation data, only get mutations which have effects on proteins
load(paste(rootDir, "/Data/mut.RData", sep = ""))
mutData <- mut[, c("gene_name_WU", "Tumor_Sample_Barcode", "trv_type_WU")]
colnames(mutData) <- c("symbol", "Sample", "ct")
proteinAffectingMut <- mutData[!(mutData$ct %in% c("silent", "rna")),]
proteinAffectingMut$Sample <- substr(proteinAffectingMut$Sample, 1, 12)

# Just keep the genes which are in the data of gene expression and pathway
# Get the gene intersection between gene expression and pathway
intersectionGenes <- intersect(row.names(gadj), row.names(mRNAsData))
# Update pathway
pathwayData <- gadj[intersectionGenes, intersectionGenes]
# Update the corresponding mRNA lists of normal and cancer samples
mRNAsData_Normal <- mRNAsData_Normal[intersectionGenes, ]
mRNAsData <- mRNAsData[intersectionGenes, ]
# Update mutation
proteinAffectingMut <- proteinAffectingMut[proteinAffectingMut$symbol %in% row.names(mRNAsData),]
proteinAffectingMut <- proteinAffectingMut[proteinAffectingMut$Sample %in% colnames(mRNAsData),]
nr <- nrow(proteinAffectingMut)
mutationData <- mRNAsData
mutationData[,] <- 0
for (i in 1:nr) {
  mutationData[proteinAffectingMut[i,1], proteinAffectingMut[i,2]] <- 1
}

# Load the gold standard (CGC)
gold_standard = read.table(
  file = paste(rootDir, "/Data/Census_allFri Sep 28 07_39_37 2018.tsv", sep = ""),
  sep = '\t', header = TRUE)
gold_standard <- gold_standard[,1]
gold_standard <- as.character(gold_standard)

# Normalize the tumor and normal data to get the differential expression
normalizedDawn <- DawnNormalize(tumorMat = mRNAsData, normalMat = mRNAsData_Normal)

# Get the DawnRank Score, this might take a while
dawnRankScore <- DawnRank(adjMatrix = pathwayData, mutationMatrix = mutationData,
                          expressionMatrix = normalizedDawn, mu = 3,
                          goldStandard = gold_standard, parallel = 2)

# Get the aggregate DawnRank scores, this might take a while
aggregateDawnRankScore <- condorcetRanking(scoreMatrix = dawnRankScore[[2]],
                                           mutationMatrix = mutationData, parallel = 2)
```

```r
# Write data
write.table(aggregateDawnRankScore[[2]], file = paste(outDir, "/DawnRank-drivers.txt", sep = ""),
            sep = "\t", quote = FALSE, row.names = TRUE, col.names = F)

# Validate with CGC
DawnRankDriver = read.table(
  file = paste(outDir, "/DawnRank-drivers.txt", sep = ""),
  sep = '\t', header = F)
DawnRankDriver_Sorted <- DawnRankDriver[order(DawnRankDriver$V2, decreasing = T),]
DawnRankDriver_Top50 <- DawnRankDriver_Sorted[1:50,]
DawnRankDriver_Top100 <- DawnRankDriver_Sorted[1:100,]
DawnRankDriver_Top150 <- DawnRankDriver_Sorted[1:150,]
DawnRankDriver_Top200 <- DawnRankDriver_Sorted[1:200,]
DawnRankDriver_Top50_Validated <- intersect(DawnRankDriver_Top50[, 1], gold_standard)
DawnRankDriver_Top100_Validated <- intersect(DawnRankDriver_Top100[, 1], gold_standard)
DawnRankDriver_Top150_Validated <- intersect(DawnRankDriver_Top150[, 1], gold_standard)
DawnRankDriver_Top200_Validated <- intersect(DawnRankDriver_Top200[, 1], gold_standard)
cat(paste("DawnRank:\n", sep = ""))
cat(paste("Number of cancer drivers in Top 50 validated: ",
          length(DawnRankDriver_Top50_Validated), "\n", sep = ""))
cat(paste("Number of cancer drivers in Top 100 validated: ",
          length(DawnRankDriver_Top100_Validated), "\n", sep = ""))
cat(paste("Number of cancer drivers in Top 150 validated: ",
          length(DawnRankDriver_Top150_Validated), "\n", sep = ""))
cat(paste("Number of cancer drivers in Top 200 validated: ",
          length(DawnRankDriver_Top200_Validated), "\n", sep = ""))
```

2. OncodriveCLUST and OncodriveFM

This is the script to run OncodriveCLUST and OncodriveFM.

To run the script, please prepare below input files and reset environment variables in the script.

The input files include the followings and are put in the folder "rootDir/Data" (rootDir is an environment variable):

- mut.RData - Mutation data
- Census_allFri Sep 28 07_39_37 2018.tsv - Cancer Gene Census (CGC)

Please note that OncodriveCLUST and OncodriveFM are run directly in the web interface of the IntOGen project, this script is only for preparing the input data and validating the result.

```r
#=================================================================
#=================================================================
# IntOGen, including OncodriveCLUST & OncodriveFM
#
# Reference:
# A. Gonzalez-Perez, C. Perez-Llamas, J. Deu-Pons, et al. Intogen-
# mutations identifies cancer drivers across tumor types. Nature
# Methods, 10:1081, 2013.
#
# Here is the link for IntOGen:
# https://www.intogen.org/analysis/home
#=================================================================
#==================================================================
# Clear the environment
rm(list = ls())

#-----------------------------------------
# Set environment variables here if any
# Please remember to create necessary folders
rootDir <- "C:/Users/phavy022/MyDoc/05CancerDriver" # And put the input files in "rootDir/Data"
outDir <- "C:/Users/phavy022/MyDoc/05CancerDriver/OncodriveCLUST_OncodriveFM" # Output folder
resultDir <- "2018-09-24_intogen" # Result folder in output folder
#-----------------------------------------

#=========================================
# Main script
#=========================================
# Read data
load(paste(rootDir, "/Data/mut.RData", sep = ""))

# Create input file
inputData <- mut[, c("chromosome_name_WU", "start_WU", "stop_WU", "strand_WU",
                     "reference_WU", "variant_WU", "Tumor_Sample_Barcode")]
inputData[,5] <- paste(inputData[,5], inputData[,6], sep = ">")
inputData <- inputData[,-6]
write.table(inputData, file = paste(outDir, "/inputData.txt", sep = ""),
            sep = "\t", quote = FALSE, row.names = FALSE, col.names = F)

#---------------------------------
# Run the method in https://www.intogen.org/analysis/home
# This is the information which has been used for the experiment:
#   The input file: inputData.txt
#   Analysis name: BRCA_20180810
#   Genome assembly: hg19 (GRCh37)
#   OncodriveFM genes threshold: 2
#   OncodriveCLUST genes threshold: 2
#---------------------------------

# Load the gold standard (CGC)
gold_standard = read.table(
  file = paste(rootDir, "/Data/Census_allFri Sep 28 07_39_37 2018.tsv", sep = ""),
  sep = '\t', header = TRUE)
gold_standard <- gold_standard[,1]
gold_standard <- as.character(gold_standard)

# Load result file
IntOGenResult = read.table(file = paste(outDir, "/", resultDir,"/gene.tsv", sep = ""),
  sep = '\t', header = TRUE)

# Validate with CGC
# OncodriveCLUST
OncodriveCLUST_Sorted <- IntOGenResult[order(IntOGenResult$QVALUE_ONCODRIVECLUST, decreasing = F),]
OncodriveCLUST_Top50 <- OncodriveCLUST_Sorted[1:50,]
OncodriveCLUST_Top100 <- OncodriveCLUST_Sorted[1:100,]
OncodriveCLUST_Top150 <- OncodriveCLUST_Sorted[1:150,]
```

```r
OncodriveCLUST_Top200 <- OncodriveCLUST_Sorted[1:200,]
OncodriveCLUST_Top50_Validated <- intersect(OncodriveCLUST_Top50[, "SYMBOL"], gold_standard)
OncodriveCLUST_Top100_Validated <- intersect(OncodriveCLUST_Top100[, "SYMBOL"], gold_standard)
OncodriveCLUST_Top150_Validated <- intersect(OncodriveCLUST_Top150[, "SYMBOL"], gold_standard)
OncodriveCLUST_Top200_Validated <- intersect(OncodriveCLUST_Top200[, "SYMBOL"], gold_standard)
cat(paste("OncodriveCLUST:\n", sep = ""))
cat(paste("Number of cancer drivers in Top 50 validated: ",
          length(OncodriveCLUST_Top50_Validated), "\n", sep = ""))
cat(paste("Number of cancer drivers in Top 100 validated: ",
          length(OncodriveCLUST_Top100_Validated), "\n", sep = ""))
cat(paste("Number of cancer drivers in Top 150 validated: ",
          length(OncodriveCLUST_Top150_Validated), "\n", sep = ""))
cat(paste("Number of cancer drivers in Top 200 validated: ",
          length(OncodriveCLUST_Top200_Validated), "\n", sep = ""))
# OncodriveFM
OncodriveFM_Sorted <- IntOGenResult[order(IntOGenResult$QVALUE_ONCODRIVEFM, decreasing = F),]
OncodriveFM_Top50 <- OncodriveFM_Sorted[1:50,]
OncodriveFM_Top100 <- OncodriveFM_Sorted[1:100,]
OncodriveFM_Top150 <- OncodriveFM_Sorted[1:150,]
OncodriveFM_Top200 <- OncodriveFM_Sorted[1:200,]
OncodriveFM_Top50_Validated <- intersect(OncodriveFM_Top50[, "SYMBOL"], gold_standard)
OncodriveFM_Top100_Validated <- intersect(OncodriveFM_Top100[, "SYMBOL"], gold_standard)
OncodriveFM_Top150_Validated <- intersect(OncodriveFM_Top150[, "SYMBOL"], gold_standard)
OncodriveFM_Top200_Validated <- intersect(OncodriveFM_Top200[, "SYMBOL"], gold_standard)
cat(paste("OncodriveFM:\n", sep = ""))
cat(paste("Number of cancer drivers in Top 50 validated: ",
          length(OncodriveFM_Top50_Validated), "\n", sep = ""))
cat(paste("Number of cancer drivers in Top 100 validated: ",
          length(OncodriveFM_Top100_Validated), "\n", sep = ""))
cat(paste("Number of cancer drivers in Top 150 validated: ",
          length(OncodriveFM_Top150_Validated), "\n", sep = ""))
cat(paste("Number of cancer drivers in Top 200 validated: ",
          length(OncodriveFM_Top200_Validated), "\n", sep = ""))

#========================================
# Script for driver gene prediction consistency
#========================================

# Functions if any
#========================================
#' This function allows you to split a dataset into 2 sub sets randomly
#' @param dat Input dataset.
#' @param s Seed.
#' @return A list containing 2 sub sets
#========================================
splitDataset <- function(dat, s) {
  set.seed(s)
  ss <- sample(1:2,size=nrow(dat),replace=TRUE,prob=c(0.5,0.5))
  dat1 <- dat[ss==1,]
  dat2 <- dat[ss==2,]
  r <- list(dat1 = dat1, dat2 = dat2)
  return(r)
}
```

```r
#--------------------------------

# Read data
load(paste(rootDir, "/Data/mut.RData", sep = ""))

# Create input file
inputData <- mut[, c("chromosome_name_WU", "start_WU", "stop_WU", "strand_WU",
                     "reference_WU", "variant_WU", "Tumor_Sample_Barcode")]
inputData[,5] <- paste(inputData[,5], inputData[,6], sep = ">")
inputData <- inputData[,-6]

# Create 10 repetitions of random two-way splits
for (i in 1:10) {
  res <- splitDataset(inputData, i)
  write.table(res$dat1, file = paste(outDir, "/inputData", i,"_1.txt", sep = ""),
            sep = "\t", quote = FALSE, row.names = FALSE, col.names = F)
  write.table(res$dat2, file = paste(outDir, "/inputData", i,"_2.txt", sep = ""),
            sep = "\t", quote = FALSE, row.names = FALSE, col.names = F)
}

#--------------------------------
# Run the method in https://www.intogen.org/analysis/home
# This is the information which has been used for the experiment:
#   The input file: inputDatai_j.txt (i: from 1 to 10, j from 1 to 2)
#   Analysis name: BRCA_20190313_i_j
#   Genome assembly: hg19 (GRCh37)
#   OncodriveFM genes threshold: 2
#   OncodriveCLUST genes threshold: 2
#--------------------------------
```

3. ActiveDriver

This is the script to run ActiveDriver.

To run the script, please prepare below input files and reset environment variables in the script.

The input files include the followings:

- all_mutations_for_TCGA_pancancer.tab
- all_phosphosites_for_TCGA_pancancer.tab
- sequences_for_TCGA_pancancer.fa
- sequence_disorder_for_TCGA_pancancer.fa

```r
#=================================================================
#=================================================================
# ActiveDriver
#
# Here is the link for ActiveDriver:
# http://reimandlab.org/software/activedriver/
#=================================================================
#=================================================================
# Clear the environment
rm(list = ls())
```

```r
# Load libraries
library(ActiveDriver)

#---------------------------------------
# Set environment variables here if any
# Please remember to create necessary folders
rootDir <- "C:/Users/phavy022/MyDoc/05CancerDriver"
outDir <- "C:/Users/phavy022/MyDoc/05CancerDriver/ActiveDriver" # Output folder
#---------------------------------------

#=======================================
# Main script
#=======================================

# # load required datasets
# muts = read.delim(paste(outDir, "/pancan12_example/mutations.txt", sep = ""))
# sites = read.delim(paste(outDir, "/pancan12_example/phosphosites.txt", sep = ""))
# seqs = read_fasta(paste(outDir, "/pancan12_example/sequences.fa", sep = ""))
# disorder = read_fasta(paste(outDir, "/pancan12_example/sequence_disorder.fa", sep = ""))
#
# # run ActiveDriver
# psnv_info = ActiveDriver(seqs, disorder, muts, sites)
#
# # save gene-based p-values and merged report as CSV files
# write.csv(psnv_info$all_gene_based_fdr, paste(outDir, "/pancan12_results_pvals.csv", sep = ""))
# write.csv(psnv_info$merged_report, paste(outDir, "/pancan12_results_merged.csv", sep = ""))
#
# # look at first few lines of every table in results
# lapply(psnv_info, head)

# load required datasets
muts = read.table(paste(outDir, "/pancancer/all_mutations_for_TCGA_pancancer.tab", sep = ""),
                  sep = "", header=TRUE, stringsAsFactors = FALSE)
sites = read.table(paste(outDir, "/pancancer/all_phosphosites_for_TCGA_pancancer.tab", sep = ""),
                   sep = "", header=TRUE, stringsAsFactors = FALSE)
seqs = read_fasta(paste(outDir, "/pancancer/sequences_for_TCGA_pancancer.fa", sep = ""))
disorder = read_fasta(paste(outDir, "/pancancer/sequence_disorder_for_TCGA_pancancer.fa", sep = ""))

# Just get BRCA mutations
muts_brca <- muts[which(grepl("brca", muts$sample_id)),]
n_row <- nrow(muts_brca)
for (i in 1:n_row) {
  muts_brca[i,3] <- strsplit(muts_brca[i,3], split=' ', fixed=TRUE)[[1]][4]
}

# run ActiveDriver
psnv_info = ActiveDriver(seqs, disorder, muts_brca, sites)

# save gene-based p-values and merged report as CSV files
write.csv(psnv_info$all_gene_based_fdr, paste(outDir, "/brca_results_pvals.csv", sep = ""))
write.csv(psnv_info$merged_report, paste(outDir, "/brca_results_merged.csv", sep = ""))

# look at first few lines of every table in results
```

```r
lapply(psnv_info, head)
```

4. DriverNet

This is the script to run DriverNet.

To run the script, please prepare below input files and reset environment variables in the script.

The input files include the followings:

- mut.RData - Mutation data
- FIsInGene_041709.txt - Graph
- BRCA_matchedData_full.RData - Tumour expression data

```r
#=================================================================
#=================================================================
# DriverNet
#
# Here is the link for DriverNet:
# https://www.bioconductor.org/packages/release/bioc/html/DriverNet.html
#=================================================================
#=================================================================
# Clear the environment
rm(list = ls())

# Load libraries
library(DriverNet)
library(igraph)
library(CancerSubtypes)
library(data.table)

#----------------------------------------
# Set environment variables here if any
# Please remember to create necessary folders
rootDir <- "C:/Users/phavy022/MyDoc/05CancerDriver"
outDir <- "C:/Users/phavy022/MyDoc/05CancerDriver/DriverNet" # Output folder
topk_mR <- 6000 # Number of mRNAs selected for analysis
#----------------------------------------

# Functions if any
#========================================
#' This function allows you to select mRNAs
#' which have the most variant Median Absolute Deviation (MAD).
#' @param matchedData Expression data of mRNAs.
#' @param topk_mR Number of mRNAs to be cut off.
#' @return A list containing selected mRNAs. The list of elements includes:
#' \item{d} {The data with rows being samples and columns being mRNAs.}
#' \item{mRs} {The names of mRNAs.}
#========================================
getDatabyMAD <- function(matchedData, topk_mR) {
  # Identify significant mRNAs by using function FSbyMAD in CancerSubtypes package
  mRNAsData = FSbyMAD(t(matchedData$mRNAs), value=topk_mR)
  mRNAsData <- t(mRNAsData)
  # Remove duplicated data
  mRNAsData <- mRNAsData[,!duplicated(colnames(mRNAsData))]

  # Prepare the result
  mRs <- colnames(mRNAsData)
  l = list(d = mRNAsData, mRs = mRs)

  return(l)
}

#========================================
# Main script
#========================================

# data(samplePatientMutationMatrix)
# data(samplePatientOutlierMatrix)
# data(sampleInfluenceGraph)
# data(sampleGeneNames)
# # The main function to compute drivers
# driversList = computeDrivers(samplePatientMutationMatrix,
#                              samplePatientOutlierMatrix,sampleInfluenceGraph,
#                              outputFolder=NULL, printToConsole=FALSE)
# drivers(driversList)[1:10]
# # random permute the gene labels to compute p-values
# randomDriversResult = computeRandomizedResult(
#   patMutMatrix=samplePatientMutationMatrix,
#   patOutMatrix=samplePatientOutlierMatrix,
#   influenceGraph=sampleInfluenceGraph,
#   geneNameList= sampleGeneNames, outputFolder=NULL,
#   printToConsole=FALSE,numberOfRandomTests=20, weight=FALSE,
#   purturbGraph=FALSE, purturbData=TRUE)
# # Summarize the results
# res = resultSummary(driversList, randomDriversResult,
#                     samplePatientMutationMatrix,sampleInfluenceGraph,
#                     outputFolder=NULL, printToConsole=FALSE)
# res[1:2,]
# write.csv(res, paste(outDir, "/sampleRes.csv", sep = ""))

# Load the tumor expression data
load(paste(rootDir, "/Data/BRCA_matchedData_full.RData", sep = ""))

# Get significant mRNAs
l <- getDatabyMAD(BRCA_matchedData, topk_mR)
mRNAs <- l$mRs
n_mRNAs <- length(mRNAs)

# Mutation
load(paste(rootDir, "/Data/mut.RData", sep = ""))
inputData <- mut[, c("Hugo_Symbol", "Tumor_Sample_Barcode")]
patientMutation <- inputData[which(inputData$Hugo_Symbol %in% mRNAs),]
mRNAs <- unique(patientMutation$Hugo_Symbol)
n_mRNAs <- length(mRNAs)
n_sample <- length(unique(patientMutation$Tumor_Sample_Barcode))
```

```
edge_list <- patientMutation
G <- graph.data.frame(edge_list, directed=FALSE)
A <- as_adjacency_matrix(G, type="both", names=TRUE, sparse=FALSE)
mutationMatrix <- A[(n_mRNAs+1):(n_mRNAs+n_sample), 1:n_mRNAs]

# Outlier
expressionMatrix <- l$d[, which(colnames(l$d) %in% mRNAs)]
outlierMatrix = getPatientOutlierMatrix(expressionMatrix)
mRNAs <- colnames(outlierMatrix)
n_mRNAs <- length(mRNAs)

# Update mutation
mutationMatrix <- mutationMatrix[, mRNAs]

# Graph
fi <- fread(paste(outDir, "/FIsInGene_041709.txt", sep = ""), select = c(1:2), header = FALSE)
fi <- fi[which(fi$V1 %in% mRNAs),]
fi <- fi[which(fi$V2 %in% mRNAs),]
edge_list <- fi
G <- graph.data.frame(edge_list, directed=FALSE)
A <- as_adjacency_matrix(G, type="both", names=TRUE, sparse=FALSE)
influenceGraph <- matrix(0, nrow =n_mRNAs, ncol = n_mRNAs)
colnames(influenceGraph) <- mRNAs
rownames(influenceGraph) <- mRNAs
influenceGraph <- merge(influenceGraph, A, by = "row.names", all.x = TRUE)
rownames(influenceGraph) <- influenceGraph[,1]
influenceGraph <- influenceGraph[,-1]
influenceGraph <- influenceGraph[, which(!grepl(".x", colnames(influenceGraph)))]
colnames(influenceGraph) <- gsub(".y", "", colnames(influenceGraph))
influenceGraph[is.na(influenceGraph)] <- 0
influenceGraph <- influenceGraph[mRNAs,]
influenceGraph <- influenceGraph[,mRNAs]
for (i in 1:n_mRNAs) {
  influenceGraph[i,i] <- 1
}
influenceGraph <- as.matrix(influenceGraph)

# Gene names
data(sampleGeneNames)

# Update mutation & outlier
rownames(mutationMatrix) <- substr(rownames(mutationMatrix), 1, 12)
com <- intersect(rownames(mutationMatrix), rownames(outlierMatrix))
mutationMatrix <- mutationMatrix[com,]
outlierMatrix <- rbind(outlierMatrix[com,],
                       outlierMatrix[which(!(rownames(outlierMatrix) %in% com)),])

# The main function to compute drivers
driversList = computeDrivers(mutationMatrix,
                             outlierMatrix, influenceGraph,
                             outputFolder=NULL, printToConsole=FALSE)
drivers(driversList)[1:10]
```

```r
# random permute the gene labels to compute p-values
randomDriversResult = computeRandomizedResult(
  patMutMatrix=mutationMatrix,
  patOutMatrix=outlierMatrix,
  influenceGraph=influenceGraph,
  geneNameList= sampleGeneNames, outputFolder=NULL,
  printToConsole=FALSE,numberOfRandomTests=20, weight=FALSE,
  purturbGraph=FALSE, purturbData=TRUE)

# Summarize the results
res = resultSummary(driversList, randomDriversResult,
                    mutationMatrix, influenceGraph,
                    outputFolder=NULL, printToConsole=FALSE)
res[1:2,]
write.csv(res, paste(outDir, "/res.csv", sep = ""))
```