**# Jaccard\_identification() 関数 仕様書**

# ＜機能概要＞

########################################################

# ＜RefCNとCNからJaccard係数行列をもとに近似的に同定したJ\_CNminを作成する＞

#

# Jaccard\_identification()関数、 入力：RefCN, CN、　出力：J\_CNmin

#

# 使い方： 参照用の分析元データがRefData、対象とする分析元データがDataのとき

# RefData.std <- scale(RefData)

# RefData.dist <- (1/2) \* (dist(RefData.std))^2

# RefData.hclust <- hclust(RefData.dist, method = ”ward.D”)

# RefCN <- cutree(RefData.hclust, k=12) # RefCN:参照用クラスターベクトル

#

# Data.std <- scale(RefData)

# Data.dist <- (1/2) \* (dist(RefData.std))^2

# Data.hclust <- hclust(Data.dist, method = ”ward.D”)

# CN <- cutree(Data.hclust, k=12) # CN:　対象クラスターベクトル

#

# J\_CNmin <- Jaccard\_identification(RefCN, CN) # CNからJ\_CNminを求める

#

###################################################

# ＜Jaccard\_identification()関数のプログラム構造概要＞

####################$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

# ＜RefCNとCNからJaccard係数行列をもとに最適化並び替えJ\_CNminを作成する＞ Jaccard\_identification()関数

#

# 入力：RefCN, CN

# 出力：J\_CNmin <- Jaccard\_identification() 最適化並び替えクラスターベクトル

#

#Jaccard\_identification <- function(RefCN, CN){

# #クラスターベクトルRefCNとCN を 展開行列J\_RefCMとJ\_CMに変換する

# J\_RefCM <- disassemble\_vec(RefCN)

# J\_CM <- disassemble\_vec(CN)

# #不一致行列を作成する

# J\_Dij <- diff\_MAT(J\_RefCM, J\_CM)

# #一致行列を作成する

# J\_Eij <- match\_MAT(J\_RefCM, J\_CM)

# #一致行列(J\_Eij) と 不一致行列(J\_Dij) からJaccard係数行列を作成する

# Jaccard\_Dij <- Jaccard\_MAT(J\_Eij, J\_Dij)

# # Jaccard係数行列を基に、J\_CMを並べ替えてJ\_CNminを作成する。For文で。

# # For文内部でwhich\_max\_for\_matrix()関数を用いている

# return(J\_CNmin)

#}

######################### 以下は、Jaccard\_identification()関数関連のRソースコード $$$$$$$$$$$######################

################$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

# ベクトルの行列展開EXPMを行う関数 MAT\_CN = disassemble\_vec(CN)

#

# ベクトル CN に対し、その展開行列 MAT\_CN を作成する

#

# 入力：CN クラスターベクトル

# 出力：MAT\_CN CNの展開行列

#

# ベクトル長： length(CN)

# ベクトル要素の最大値： max(CN)

#

# のとき

#

# 展開行列のサイズは、

# MAT\_CN <- matrix(data=rep(x=0, times=max(CN)\*length(CN)), nrow = max(CN), ncol = length(CN))

#

# for(i in 1:length(CN)){

# MAT\_CN[CN[i],i] <- as.numeric(1)

# }

#

disassemble\_vec <- function(CN){

MAT\_CN <- matrix(data=rep(x=0, times=max(CN)\*length(CN)), nrow = max(CN), ncol = length(CN))

for(i in 1:length(CN)){

MAT\_CN[CN[i],i] <- as.numeric(1)

}

return(MAT\_CN)

}

################$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

################$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

# 展開行列から展開元ベクトルを作成CNTする関数 ASEM\_CN = assemble\_vec(MAT\_CN)

#

# 展開行列 MAT\_CN から展開元ベクトル ASEM\_CN を作成する

#

# 入力：MAT\_CN 展開行列

# 出力：ASEM\_CN MAT\_CNから戻したクラスターベクトル

#

# ベクトル長： ncol(MAT\_CN) -> length(CN)

#

# のとき

#

# 展開元ベクトルのサイズは、

# ASEM\_CN <- rep(x = 0, times = ncol(MAT\_CN))

#

assemble\_vec <- function(MAT\_CN){

ASEM\_CN <- rep(x=0, times=ncol(MAT\_CN))

for(i in 1:ncol(MAT\_CN)){

count <- as.numeric(1)

while(MAT\_CN[count,i] == 0){

count <- count + 1

}

ASEM\_CN[i] <- count

}

return(ASEM\_CN)

}

################$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

####################$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

# ＜不一致行列の作成＞ diff\_MAT()関数

# 参照ベクトルの展開行列 MAT\_RefCN とMAT\_CN 間の不一致行列(Dij) を作成する

#

# 入力：MAT\_RefCN,MAT\_CN 参照クラスターベクトルの展開行列、クラスターベクトルの展開行列

# 出力：diff\_MAT 不一致行列D\_MAT\_RefCN\_CN(Dij)

#

diff\_MAT <- function(MAT\_RefCN, MAT\_CN){

#差異度行列D\_MAT\_RefCN\_CNの作成(=Dij)

D\_MAT\_RefCN\_CN <- matrix(0,nrow = nrow(MAT\_RefCN), ncol = nrow(MAT\_RefCN))

for(j in 1:nrow(MAT\_CN)){

for(i in 1:nrow(MAT\_RefCN)){

D\_MAT\_RefCN\_CN[i,j] <- sum((MAT\_RefCN[i,]-MAT\_CN[j,])^2)

}

}

return(D\_MAT\_RefCN\_CN)

}

####################$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

# ＜一致行列の作成＞ match\_MAT()関数

# 参照ベクトルの展開行列 MAT\_RefCN とMAT\_CN から一致行列E\_MAT\_RefCN\_CN(Eij)の作成

#

# 入力：MAT\_RefCN,MAT\_CN 参照クラスターベクトルの展開行列、クラスターベクトルの展開行列

# 出力：MATCH\_MAT 一致行列(Eij)

#

match\_MAT <- function(MAT\_RefCN, MAT\_CN){

#一致行列E\_MAT\_RefCN\_CNの作成(=Eij)

E\_MAT\_RefCN\_CN <- MAT\_RefCN %\*% t(MAT\_CN)

return(E\_MAT\_RefCN\_CN)

}

####################$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

# ＜Jaccard行列の作成＞ Jaccard\_MAT()関数

# 一致行列(Eij)と不一致行列(Dij)から Jaccard行列(Jaccard\_Dij)を作成する

#

# 入力：E\_MAT\_RefCN\_CN, D\_MAT\_RefCN\_CN 一致行列(Eij)と不一致行列(Dij)

# 出力：Jaccard\_MAT Jaccard行列(Jaccard\_Dij)

#

Jaccard\_MAT <- function(E\_MAT\_RefCN\_CN, D\_MAT\_RefCN\_CN){

#Jaccard行列Jaccard\_MATの作成(=Jaccard\_Dij)

Jaccard\_MAT <- E\_MAT\_RefCN\_CN / (E\_MAT\_RefCN\_CN + D\_MAT\_RefCN\_CN)

return(Jaccard\_MAT)

}

###############################################################

# ＜二次元配列の要素最大の行数と列数を返す関数＞ which\_max\_for\_matrix()

# 入力：MAT

# 出力：行と列番号（複数ある場合は、一番先に登場するものの行と列）

#

which\_max\_for\_matrix <- function(MAT){

location\_vec <- c(1,1)

nrow\_MAT <- nrow(MAT)

ncol\_MAT <- ncol(MAT)

location\_MAT <- which(MAT == max(MAT))[1]

j <- 1

while(location\_MAT - nrow\_MAT\*j > 0){

j<- j+1

}

i <- location\_MAT - (j-1)\*nrow\_MAT

location\_vec <- c(i,j)

return(location\_vec)

}

################################################### SUM\_DIFF\_VAL = 1/2 \* (RefCN - CN)^2

cluster\_min\_defferencial\_value <- function(RefCN, CN){

# RefCN と CNの「クラスター差異度(1/2倍したもの)」を返す関数

#

# ＜参照ベクトルの展開行列化　処理＞

# 参照ベクトルRefCNを行列展開する

# RefCN\_name <- names(RefCN) # 参照ベクトルのラベルを取得

MAT\_RefCN <- matrix(data=rep(x=0, times=max(RefCN)\*length(RefCN)), nrow = max(RefCN), ncol = length(RefCN))

MAT\_RefCN <- disassemble\_vec(RefCN)

###################################################

# ＜対象ベクトルの展開行列化　処理＞

# 対象ベクトルCNを行列展開する

MAT\_CN <- matrix(data=rep(x=0, times=max(RefCN)\*length(RefCN)), nrow = max(RefCN), ncol = length(RefCN))

MAT\_CN <- disassemble\_vec(CN)

######## クラスター差異度の計算

MAT\_SUM <- matrix(data=rep(x=0, times=max(RefCN)\*length(RefCN)), nrow = max(RefCN), ncol = length(RefCN))

MAT\_SUM <- (MAT\_RefCN - MAT\_CN)^2

SUM\_DIFF\_VAL <- 1/2 \* sum(MAT\_SUM)

return(SUM\_DIFF\_VAL)

}

######################### MAIN 関数 $$$$$$$$$$$$$$$$$$$$##############################

####################$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

# ＜RefCNとCNからJaccard係数行列をもとに最適化並び替えJ\_CNminを作成する＞ Jaccard\_identification()関数

#

# 入力：RefCN, CN

# 出力：J\_CNmin <- Jaccard\_identification() 最適化並び替えクラスターベクトル

#

Jaccard\_identification <- function(RefCN, CN){

#クラスターベクトルRefCNとCN を 展開行列J\_RefCMとJ\_CMに変換する

J\_RefCM <- disassemble\_vec(RefCN)

J\_CM <- disassemble\_vec(CN)

#不一致行列を作成する

J\_Dij <- diff\_MAT(J\_RefCM, J\_CM)

#一致行列を作成する

J\_Eij <- match\_MAT(J\_RefCM, J\_CM)

#一致行列(J\_Eij) と 不一致行列(J\_Dij) からJaccard係数行列を作成する

Jaccard\_Dij <- Jaccard\_MAT(J\_Eij, J\_Dij)

#列交換用作業行列J\_MAT に Jaccard\_Dijをコピー

J\_MAT <- Jaccard\_Dij

#行交換用作業行列CMm に J\_CMをコピー

CMm <- J\_CM

# 列入れ替えカウンター作成:k

k <- ncol(Jaccard\_Dij) -1

for(i in 1:k){

#J\_MATの最大値(一番目のもの)の行sと列tを求める

max\_Jaccard\_s <- which\_max\_for\_matrix(J\_MAT)[1]

max\_Jaccard\_t <- which\_max\_for\_matrix(J\_MAT)[2]

#CMmのs行とt行を交換する

CMm\_temp\_s <- CMm[max\_Jaccard\_s,]

CMm\_temp\_t <- CMm[max\_Jaccard\_t,]

CMm[max\_Jaccard\_s,] <- CMm\_temp\_t

CMm[max\_Jaccard\_t,] <- CMm\_temp\_s

#J\_MATのs列とt列を交換する

J\_temp\_s <- J\_MAT[,max\_Jaccard\_s]

J\_temp\_t <- J\_MAT[,max\_Jaccard\_t]

J\_MAT[,max\_Jaccard\_s] <- J\_temp\_t

J\_MAT[,max\_Jaccard\_t] <- J\_temp\_s

#J\_MATのs行s列すべてに－１を入れる。s行を確定させるため

J\_MAT[max\_Jaccard\_s,] <- -1

J\_MAT[,max\_Jaccard\_s] <- -1

}

J\_CNmin <- assemble\_vec(CMm)

names(J\_CNmin) <- names(RefCN)

return(J\_CNmin)

}

######################### Jaccard\_identification()関数関連の定義終わり $$$$$$$$$$$$$$$$$$$$######################

#######################$$$$$$$$$$$$ 関数の終わり $$$$$$$$$$$$$$$$$$$#############