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

# ＜機能概要＞

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

# 参照クラスターベクトル RefCN と 対象クラスターベクトル CN から

# クラスター差異度最小化ベクトル CNmin を作成する

#

# cluster\_identification()関数、　　　入力：RefCN, CN、　出力：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

#

# CNmin <- cluster\_identification(RefCN, CN) # CNからCNminを求める

#

#

# ただし、k-u-v >= 11 の場合は、permutation()関数が正常に実施できずクラスター同定

# ができない。この場合は、ERR\_CN を返り値(CNmin)として返す。

# ERR\_CN = (A1, A2, A3, ・・・, AN)、ただしA1 = 0, A2 = k-u-v, A3～AN = 0

#

# 返り値 CNmin の判定：

# if(CNmin[1] == 0){

# # クラスター同定できなかった！ERR\_CN　が戻った。

# # CNmin[2] には k-u-v が入っている。

# }else{

# # クラスター同定は正常にできた。

# }

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

# ＜cluster\_identification()関数プログラムの構造概要＞

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

## RefCNから展開行列MAT\_RefCN を作成する。disassemble\_vec()関数

# MAT\_RefCN <- disassemble\_vec(RefCN)

# CNから展開行列MAT\_CN を作成する。disassemble\_vec()関数

# MAT\_CN <- disassemble\_vec(CN)

## 展開行列間の同等行の検出 match\_line\_RefCN()関数

## MAT\_RefCNの行（クラスター）と一致もしくは最も類似するMAT\_CNの

##行を対応づけるベクトル(MATCH\_LINE)を返す。一致もしくは類似が未定な行

##に対応するベクトル要素には”０”が入る。k-u-v の処理である。

# MATCH\_LINE <- match\_line\_RefCN(MAT\_RefCN, MAT\_CN)

## match\_line\_RefCN()関数の中では、

##一致行列E\_MAT\_RefCN\_CNの作成(=Eij)

##差異度行列D\_MAT\_RefCN\_CNの作成(=Dij)

## 非一致参照行列作成

# UMATCH\_MAT\_RefCN <- unmatch\_MAT\_RefCN(MAT\_RefCN, MATCH\_LINE)

## 非一致対象行列作成

# UMATCH\_MAT\_CN <- unmatch\_MAT\_CN(MAT\_CN, MATCH\_LINE)

#＃ 非一致行列のすべての組み合わせで行入れ替えを行い、差異度最小を求める。

# if(nrow(UMATCH\_MAT\_CN) != 0 && nrow(UMATCH\_MAT\_CN) < 11){

# # 非一致行列の行数の階乗を計算し、順列組合せ行列を作成する

# kaijyou <- gamma(nrow(UMATCH\_MAT\_CN)+1)

# # kaijyouが大きすぎると行列が確保できなくなるので注意

# PER\_MAT <- matrix(rep(x=0, times=nrow(UMATCH\_MAT\_CN)\*kaijyou), nrow=nrow(UMATCH\_MAT\_CN), ncol=kaijyou)

# PER\_MAT <- t(permutations(nrow(UMATCH\_MAT\_CN),nrow(UMATCH\_MAT\_CN)))

# # クラスター差異度ベクトル SUM\_VAL の作成

# SUM\_VAL <- diff\_cluster(UMATCH\_MAT\_RefCN, UMATCH\_MAT\_CN, PER\_MAT, kaijyou)

# min\_diff\_cluster\_PTN <- which(SUM\_VAL == min(SUM\_VAL))[1] # クラスター差異度最小の最初のパターン番号(PRE\_MAT)

# # クラスター差異度最小化行列 min\_diffcluster\_MAT\_CN の生成

# min\_diffcluster\_MAT\_CN<-min\_diffcluster\_mat(MATCH\_LINE,MAT\_RefCN,MAT\_CN,UMATCH\_MAT\_CN,

# min\_diff\_cluster\_PTN, PER\_MAT)

# # クラスター差異度最小化行列 min\_diffcluster\_MAT\_CN から ベクトル CNmin への畳み込み（逆展開）

# CNmin <- assemble\_vec(min\_diffcluster\_MAT\_CN)

# names(CNmin) <- RefCN\_name # 名前付きベクトルに戻るためにRefCNのラベルを貼る

# #browser()

# return(CNmin)

# }else{

# # k-u-v が11以上の場合は、エラーベクトルERR\_CNを返す

# # ERR\_CN = (A1, A2, ・・・,AN) であるが、

# # A1 = 0, A2 = k-u-v, A3～AN = 0 という特殊なベクトルである。

# #

# # ERR\_CN を初期化"all 0"にする。

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

# # A2 に k-u-v を入れる

# ERR\_CN[2] <- k-u-v

# names(ERR\_CN) <- RefCN\_name # 関数の返り値のスタイルを統一するために名前付きベクトルに戻す

# return(ERR\_CN)

# }

#}

#########$$$$$$$$$$ 以下は、cluster\_identification()関数のRソースコード$$$$$$$$$$$$$$$$######################

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

# 複数の階層型クラスター分析結果に関するクラスター同定手法「クラスター差異度最小化法」

# cluster\_identification()関数に関する一連の関数定義

#

library(gtools) # premutations()関数を使用するためにパッケージ読み込み

#########$$$$$$$$$$ ここから cluster\_identification()関数用の一連の内部関数定義 $$$$$$$$$$$$$$$$######################

# 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)

}

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

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

# ＜同等行の検出＞ match\_line\_RefCN()関数

# 参照ベクトルの展開行列 MAT\_RefCN とMAT\_CN 間の同等行（クラスター）の判定

# による同等ベクトル MATCH\_LINE の作成

# MATCH\_LINE[クラスター数(nrow(MAT\_RefCN))]

#

# MATCH\_LINE <- match\_Line\_RefCN(MAT\_RefCN, MAT\_CN)

#

# 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\_LINE 同等行ベクトル

#

# MATCH\_LINEベクトルにはMAT\_RefCNの各行が一致もしくは差異度1のとき

# もしくは、差異度2以上で、かつ、差異度行列-一致行列が正となる組合せにおいて

# MAT\_CNの行番号が入る。その他の場合には0が入る。

#

# 差異度行列Dij = 0 (u) or 1 (v) は一つに定まるのでここで計算回数削減する

# (k－u - v)! になる。

#

match\_line\_RefCN <- 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)

#差異度行列D\_MAT\_RefCN\_CNの作成(=Dij)

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

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)

}

}

# 差異度行列からの差異度唯一最小列ベクトルD\_min\_rowの作成

D\_min\_row <- rep(x=0, times=nrow(MAT\_RefCN))

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

if(length(which(D\_MAT\_RefCN\_CN[i,] == min(D\_MAT\_RefCN\_CN[i,]))) == 1){

D\_min\_row[i] <- which(D\_MAT\_RefCN\_CN[i,] == min(D\_MAT\_RefCN\_CN[i,]))[1]

}

}

S\_MAT\_RefCN\_CN <- E\_MAT\_RefCN\_CN - D\_MAT\_RefCN\_CN

# S行列からの唯一最大列ベクトルS\_min\_rowの作成

S\_max\_row <- rep(x=0, times=nrow(MAT\_RefCN))

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

if(length(which(S\_MAT\_RefCN\_CN[i,] == max(S\_MAT\_RefCN\_CN[i,]))) == 1){

S\_max\_row[i] <- which(S\_MAT\_RefCN\_CN[i,] == max(S\_MAT\_RefCN\_CN[i,]))[1]

}

}

# 一致行列Eからの唯一最大列ベクトルE\_min\_rowの作成

E\_max\_row <- rep(x=0, times=nrow(MAT\_RefCN))

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

if(length(which(E\_MAT\_RefCN\_CN[i,] == max(E\_MAT\_RefCN\_CN[i,]))) == 1){

E\_max\_row[i] <- which(E\_MAT\_RefCN\_CN[i,] == max(E\_MAT\_RefCN\_CN[i,]))[1]

}

}

match\_counter <- 0

MATCH\_LINE <- rep(x = 0, times= nrow(MAT\_RefCN)) #クラスター数分0を埋める

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

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

if (max((MAT\_RefCN[i,]-MAT\_CN[j,])^2) == 0){ #　一致行の場合 (k-u)

MATCH\_LINE[i] <- j

match\_counter <- match\_counter + 1

}else{

if(sum((MAT\_RefCN[i,]-MAT\_CN[j,])^2) == 1){　# 差異度1の同等行の場合

MATCH\_LINE[i] <- j

match\_counter <- match\_counter + 1

}else{

if( E\_MAT\_RefCN\_CN[i,j] - D\_MAT\_RefCN\_CN[i,j] > 0 ){ # 一致行列Eij -差異度行列Dij > 0の場合 (k-u-v)

MATCH\_LINE[i] <- j

match\_counter <- match\_counter + 1

}

}

}

}

}

return(MATCH\_LINE)

}

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

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

#　非一致参照行列の作成　　unmatch\_MAT\_RefCN() 関数

# match\_line\_RefCN()関数の出力であるMATCH\_LINEベクトルとMAT\_RefCN

#　　から、非一致参照行列 UMATCH\_MAT\_RfCN を作成する

#

# 入力：MAT\_RefCN, MATCH\_LINE

#

# UMATCH\_MAT\_RefCN <- unmatch\_MAT\_RefCN(MAT\_RefCN, MATCH\_LINE)

#

unmatch\_MAT\_RefCN <- function(MAT\_RefCN, MATCH\_LINE){

k <- 0　# 非一致参照行列の行数を確定するために k を計算する

for(i in 1:length(MATCH\_LINE)){

if (MATCH\_LINE[i] == 0){

k <- k + 1

}

}

UMATCH\_MAT\_RefCN <- matrix(data=rep(x=0, times=k\*ncol(MAT\_RefCN)), nrow = k, ncol = ncol(MAT\_RefCN))

k <- 1

for(i in 1:length(MATCH\_LINE)){

if (MATCH\_LINE[i] == 0){

UMATCH\_MAT\_RefCN[k,] <- MAT\_RefCN[i,]

k <- k + 1

}

}

return(UMATCH\_MAT\_RefCN)

}

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

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

# 非一致行列の作成 unmatch\_MAT\_CN() 関数

# match\_line\_RefCN()関数の出力であるMATCH\_LINEベクトルとMAT\_CN

# から、非一致行列 UMATCH\_MAT\_CN を作成する

#

# 入力：MAT\_CN, MATCH\_LINE

#

# UMATCH\_MAT\_CN <- unmatch\_MAT\_CN(MAT\_CN, MATCH\_LINE)

unmatch\_MAT\_CN <- function(MAT\_CN, MATCH\_LINE){

sort\_MATCH\_LINE <- rep(x=0, times=nrow(MAT\_CN)) # UMATCH\_MAT\_CN作成のためのMATCH\_LINE相当ベクトル定義

k <- 0 # 非一致行列の行数を確定するために k を計算する

for(i in 1:length(MATCH\_LINE)){

if (MATCH\_LINE[i] == 0){

# sort\_MATCH\_LINE[MATCH\_LINE[i]] <- MATCH\_LINE[i]

k <- k + 1

}

}

UMATCH\_MAT\_CN <- matrix(data=rep(x=0, times=k\*length(RefCN)), nrow = k, ncol = length(RefCN))

# UMATCH\_MAT\_CN 作成のためのsort\_MATCH\_LINEベクトルを設定

for(i in 1:length(MATCH\_LINE)){

if (MATCH\_LINE[i] != 0){

sort\_MATCH\_LINE[MATCH\_LINE[i]] <- MATCH\_LINE[i]

}

}

k <- 1

for(i in 1:length(MATCH\_LINE)){

if (sort\_MATCH\_LINE[i] == 0){

UMATCH\_MAT\_CN[k,] <- MAT\_CN[i,]

k <- k + 1

}

}

return(UMATCH\_MAT\_CN)

}

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

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

# クラスター差異度の計算 diff\_cluster()関数

# 非一致参照行列 UMATCH\_MAT\_RefCN と 非一致行列　UMATCH\_MAT\_CN

# から、クラスター差異度ベクトル SUM\_VAL を計算する

# 入力：UMATCH\_MAT\_RefCN, UMATCH\_MAT\_CN, PER\_MAT, kaijyou

# 出力：SUM\_VAL <- diff\_cluster(UMATCH\_MAT\_RefCN, UMATCH\_MAT\_CN, PER\_MAT, kaijyou)

diff\_cluster <- function(UMATCH\_MAT\_RefCN, UMATCH\_MAT\_CN, PER\_MAT, kaijyou){

SUM\_VAL <- rep(x=0, times=kaijyou)

for(j in 1:kaijyou){

for(i in 1:nrow(UMATCH\_MAT\_RefCN)){

SUM\_VAL[j] <- sum((UMATCH\_MAT\_RefCN[i,]-UMATCH\_MAT\_CN[PER\_MAT[i,j],])^2) + SUM\_VAL[j]

}

}

return(SUM\_VAL)

}

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

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

# min\_diffcluster\_MAT\_CN の作成 min\_diffcluster\_mat() 関数

#

# 入力：MATCH\_LINE, MAT\_RefCN, MAT\_CN, UMATCH\_MAT\_CN, min\_diff\_cluster\_PTN, PER\_MAT

# 出力：min\_diffcluster\_MAT\_CN <- min\_diffcluster\_MAT()

min\_diffcluster\_mat <- function(MATCH\_LINE, MAT\_RefCN, MAT\_CN, UMATCH\_MAT\_CN, min\_diff\_cluster\_PTN, PER\_MAT){

min\_diffcluster\_MAT\_CN <- MAT\_RefCN # MAT\_RefCNを丸ごとコピー ＜バグ！MAT\_RefCNが生き残ってしまう！！＞

k <- 1 # 非一致行列 UMATCH\_MAT\_CNの行から設定するためのカウンター初期化

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

if(MATCH\_LINE[i] == 0){

# 不一致行は、差異度最小PTNのPER\_MAT[k,min\_diff\_cluster\_PTN]のなかの行番号のUMATCH\_MAT\_CNの行ベクトル

min\_diffcluster\_MAT\_CN[i,] <- UMATCH\_MAT\_CN[PER\_MAT[k,min\_diff\_cluster\_PTN],]

k <- k + 1 # MATCH\_LINE[i] が 0 の時だけk をインクリメントする。

}else{

min\_diffcluster\_MAT\_CN[i,] <- MAT\_CN[MATCH\_LINE[i],] # MAT\_CN の一致行で上書き

}

}

return(min\_diffcluster\_MAT\_CN)

}

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

######################### MAIN 関数(cluster\_identification)$$$$$$$$$$$$$$$$$$$$##############################

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

# 参照ベクトル RefCN と 対象ベクトル CN から クラスター差異度最小化ベクトル ASEM\_CN を作成する

# cluster\_identification()関数

#

# 入力：RefCN, CN

# 出力：ASEM\_CN <- cluster\_identification()

#

# 使い方： 参照ベクトル RefCN 2017-2019の ssdse.cnum

# 対象ベクトル CN <- ssdse.cnum

#

# ssdse.cnum <- cluster\_identification(RefCN, CN)

#

# 仕様の変更：正常終了しなかった場合

# k-u-v >= 11 でpermutation()関数が使えなかった場合の本関数の返り値を ERR\_CN とし、

#

# ERR\_CN = (A1, A2, A3, ・・・,AN)

# A1 = 0, A2 = k-u-v, A3～AN = 0

# とする。

#

# 返り値の正否判定：

# if(cluster\_identification(RefCN, CN)[1] == 0){

# # クラスター同定できなかった！

# # cluster\_identification(RefCN, CN)[2] には k-u-v が入っている。

# }else{

# # クラスター同定は正常にできた。

# }

#

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

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

# ＜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)

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

#

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

#ASEM\_CN <- assemble\_vec(MAT\_RefCN)

#names(ASEM\_CN) <- RefCN\_name # 名前付きベクトルに戻るためにRefCNのラベルを貼る

#CN <- ssdse.cnum

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

# ＜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)

MATCH\_LINE <- rep(x = 0, times= max(RefCN))

MATCH\_LINE <- match\_line\_RefCN(MAT\_RefCN, MAT\_CN)

# 全ての行で類似行があった場合

if(min(MATCH\_LINE) > 0){

# MATCH\_LINE から「クラスター差異度最小化ベクトル」の展開行列を作成して返す

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

for(i in 1:length(MATCH\_LINE)){

MAT\_CNm[i,] <- MAT\_CN[MATCH\_LINE[i],]

}

ASEM\_CN <- assemble\_vec(MAT\_CNm)

names(ASEM\_CN) <- RefCN\_name # 名前付きベクトルに戻るためにRefCNのラベルを貼る

return(ASEM\_CN)

}

UMATCH\_MAT\_RefCN <- unmatch\_MAT\_RefCN(MAT\_RefCN, MATCH\_LINE) # 非一致参照行列作成

UMATCH\_MAT\_CN <- unmatch\_MAT\_CN(MAT\_CN, MATCH\_LINE) # 非一致行列作成

if(nrow(UMATCH\_MAT\_CN) != 0 && nrow(UMATCH\_MAT\_CN) < 11){

# 非一致行列の行数の階乗を計算し、順列組合せ行列を作成する

kaijyou <- gamma(nrow(UMATCH\_MAT\_CN)+1)

# kaijyouが大きすぎると行列が確保できなくなるので注意

PER\_MAT <- matrix(rep(x=0, times=nrow(UMATCH\_MAT\_CN)\*kaijyou), nrow=nrow(UMATCH\_MAT\_CN), ncol=kaijyou)

PER\_MAT <- t(permutations(nrow(UMATCH\_MAT\_CN),nrow(UMATCH\_MAT\_CN)))

# クラスター差異度ベクトル SUM\_VAL の作成

SUM\_VAL <- diff\_cluster(UMATCH\_MAT\_RefCN, UMATCH\_MAT\_CN, PER\_MAT, kaijyou)

min\_diff\_cluster\_PTN <- which(SUM\_VAL == min(SUM\_VAL))[1] # クラスター差異度最小の最初のパターン番号(PRE\_MAT)

# 差異度最小となるすべての順列組み合わせパターンを求める

min\_diff\_cluster\_PTN\_all <- which(SUM\_VAL == min(SUM\_VAL))

write.csv(min\_diff\_cluster\_PTN\_all, file="min\_diff\_cluster\_PTN\_all.csv")

# クラスター差異度最小化行列 min\_diffcluster\_MAT\_CN の生成

min\_diffcluster\_MAT\_CN <- min\_diffcluster\_mat(MATCH\_LINE, MAT\_RefCN, MAT\_CN, UMATCH\_MAT\_CN, min\_diff\_cluster\_PTN, PER\_MAT)

# クラスター差異度最小化行列 min\_diffcluster\_MAT\_CN から ベクトル ASEM\_CN への畳み込み（逆展開）

ASEM\_CN <- assemble\_vec(min\_diffcluster\_MAT\_CN)

names(ASEM\_CN) <- RefCN\_name # 名前付きベクトルに戻るためにRefCNのラベルを貼る

#browser()

return(ASEM\_CN)

}else{

#return(CN) # k-u-v が11以上の場合は、そのままCNを返す

# という初期のエラー仕様ではエラーなのか正常解なのかが判別し難いので仕様変更する

# k-u-v が11以上の場合は、エラーベクトルERR\_CNを返す

# ERR\_CN = (A1, A2, ・・・,AN) であるが、 A1 = 0, A2 = k-u-v, A3～AN = 0 という特殊なベクトルである。

#

# ERR\_CN を初期化"all 0"にする。

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

# A2 に k-u-v を入れる

ERR\_CN[2] <- k-u-v

names(ERR\_CN) <- RefCN\_name # 関数の返り値のスタイルを統一するために名前付きベクトルに戻す

return(ERR\_CN)

}

}

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

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

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

# RefCN と CNの「展開行列の非一致行数」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)

MATCH\_LINE <- rep(x = 0, times= max(RefCN))

MATCH\_LINE <- match\_line\_RefCN(MAT\_RefCN, MAT\_CN)

UMATCH\_MAT\_RefCN <- unmatch\_MAT\_RefCN(MAT\_RefCN, MATCH\_LINE) # 非一致参照行列作成

UMATCH\_MAT\_CN <- unmatch\_MAT\_CN(MAT\_CN, MATCH\_LINE) # 非一致行列作成

UMATCH\_NROW <- nrow(UMATCH\_MAT\_CN)

return(UMATCH\_NROW)

}

#########$$$$$$$$$$ ここまでが cluster\_identification()関数用の一連の関数定義 $$$$$$$$$$$$$$$$######################