**This file contains my comments/explanations on the script here (**[**https://github.com/heejungshim/multiscale\_analysis/blob/master/src/R/prepare.DNase.funcs.R**](https://github.com/heejungshim/multiscale_analysis/blob/master/src/R/prepare.DNase.funcs.R)**). My comments start with [HJ] and are in blue.**

## `prepare.DNase.funcs.R' contains R functions for 1) reading DNase-seq data from files in hdf5, 2) reading mappability information from file in hdf5, 3) masking 5bp surrounding any SNP (i.e., the SNP position and 2bp on either side) to eliminate biases stemming from DNase I sequence preference, and 4) combining DNase-seq data from two strands while taking mappability into account as Shim and Stephens (2014) did.

## The funcrion get.counts.h5 in utils.R is used.

## see prepare.DNase.R for usage.

##

##

## Copyright (C) 2014 Heejung Shim

##

##

## This program is free software: you can redistribute it and/or modify

## it under the terms of the GNU General Public License as published by

## the Free Software Foundation, either version 3 of the License, or

## (at your option) any later version.

##

## This program is distributed in the hope that it will be useful,

## but WITHOUT ANY WARRANTY; without even the implied warranty of

## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

## GNU General Public License for more details.

##

## You should have received a copy of the GNU General Public License

## along with this program. If not, see <http://www.gnu.org/licenses/>.

##' Prepare DNase data from raw data.

##'

##'

##' This function performs 1) reading DNase-seq data from files in hdf5, 2) reading mappability information from file in hdf5, 3) masking 5bp surrounding any SNP (i.e., the SNP position and 2bp on either side) to eliminate biases stemming from DNase I sequence preference, and 4) combining DNase-seq data from two strands while taking mappability into account as in Shim and Stephens (2014) did.

##'

##'

##' @param hdf5.data.path string; path to a directory which contains DNase data as hdf5 format.

##' @param hdf5.mapp.path string; path to file of mappability as hdf5 format

##' @param geno.info.path string; default = NULL; path to file which contains SNP positions (6th column) located in a region of interest.

##' @param inds.IDs a vector of individual IDs

##' @param chrIX string; chromosome name

##' @param locus.start

##' @param locus.end

##' @return DNase.dat a matrix of numIND by size of region;

##' @return mappability a vector of numBPs (size of region); either 0 or 1; 1 indicates mappable position in either strand.

read.DNase.data <- function(hdf5.data.path, hdf5.mapp.path, geno.info.path = NULL, inds.IDs, chrIX, locus.start, locus.end){

numINDs = length(inds.IDs)

numBPs = locus.end - locus.start + 1

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

### 1. Read phenotype data

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

DNase.hdf5 = matrix(data=NA, nr = numINDs, nc = numBPs\*2)

for(i in 1:numINDs){

path.fwd = paste0(hdf5.data.path, "dnase\_", inds.IDs[i], "\_fwd.h5")

DNase.hdf5[i, 1:numBPs] = as.matrix(get.counts.h5(path.fwd, chrIX, locus.start+1, locus.end+1))

path.rev = paste0(hdf5.data.path, "dnase\_", inds.IDs[i], "\_rev.h5")

DNase.hdf5[i, ((1:numBPs)+numBPs)] = as.matrix(get.counts.h5(path.rev, chrIX, locus.start+1, locus.end+1))

}

**[HJ] the function “get.counts.h5” is implemented here (**[**https://github.com/heejungshim/multiscale\_analysis/blob/master/src/R/utils.R**](https://github.com/heejungshim/multiscale_analysis/blob/master/src/R/utils.R)**).**

#dim(DNase.hdf5)

# 70 by 2048; each row corresponds to each individual; the first (second) 1024 columns contain DNass-seq read count from +(-) strand in each positions;

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

# 2. Read mappability information

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

map.hdf5 = matrix(data=NA, nr = 1, nc = numBPs\*2)

map.hdf5[1, 1:numBPs] = as.matrix(get.counts.h5(hdf5.mapp.path, chrIX, locus.start+1, locus.end+1))

map.hdf5[1, ((1:numBPs)+numBPs)] = as.matrix(get.counts.h5(hdf5.mapp.path, chrIX, locus.start-20+2, locus.end-20+2))

#dim(map.hdf5) # 1 by 2048; the first (second) 1024 rows indicates mappability from +(-) strand in each positions; `1' indicates uniquely mappable base.

map.dat = map.hdf5

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

## 3. Mask 5bp surrounding any SNP (i.e., the SNP position and 2bp on either side)

## to eliminate biases stemming from DNase I sequence preference

## (see the supplementary material of Degner et al 2012 for details).

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

loc\_info = rep(NA, numBPs\*2)

loc\_info[1:numBPs] = locus.start:locus.end

loc\_info[(1:numBPs)+numBPs] = locus.start:locus.end

DNase.in = DNase.hdf5

## read all SNP information at the site

if(is.null(geno.info.path)){

DNase.out = DNase.in

}else{

if(file.info(geno.info.path)$size == 0){

DNase.out = DNase.in

}else{

geno = read.table(geno.info.path, as.is = TRUE)

SNP\_posi = as.numeric(geno[,6])

del\_posi = sort(unique(union(union(union(union(SNP\_posi-2, SNP\_posi -1), SNP\_posi), SNP\_posi+1), SNP\_posi+2)))

wh\_del = which((loc\_info %in% del\_posi)==TRUE)

DNase.out = DNase.in

if(length(wh\_del) > 0){

DNase.out[,wh\_del] = matrix(data=0, nr = numINDs, nc = length(wh\_del))

}

}

}

DNase.dat = DNase.out

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

## 4. Combine two strands while taking mappability into account

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

# take mappability into account

map = rep(0, numBPs\*2)

wh = (map.dat[1,] == 1)

map[wh] = 1

dat = matrix(data = 0, nr = numINDs, nc = numBPs\*2)

dat[,wh] = as.matrix(DNase.dat[,wh])

# prepare mappability as an output

map.out = rep(0, numBPs)

wh = which((map[1:numBPs] == 1) | (map[(numBPs+1):(numBPs+numBPs)] == 1))

map.out[wh] = rep(1, length(wh))

# combine two strands

all.dat = dat[,1:numBPs] + dat[,(numBPs+1):(numBPs+numBPs)]

all.map = map[1:numBPs] + map[(numBPs+1):(numBPs+numBPs)]

pheno.dat = matrix(data = 0, nr = numINDs, nc = numBPs)

wh2 = which(all.map > 0)

pheno.dat[,wh2] = t(t(all.dat[,wh2])/all.map[wh2])

return(list(DNase.dat = pheno.dat, mappability = map.out))

}

**[HJ] outputs:**

**##' @return DNase.dat a matrix of numIND by size of region;**

**##' @return mappability a vector of numBPs (size of region); either 0 or 1; 1 indicates mappable position in either strand.**