Package 'HBP'

July 7, 2017

Type Package

Title Hi-C BED File Analysis Pipeline

Version 0.1.2

Date 2017-07-05

Depends R (>= 3.0.0)

Imports
grid,lattice,BiocInstaller,rtracklayer,OmicCircos,ggplot2,igraph,reshape2,pgirmess,coin,multcomp,flexclust,HiTC,gplots

Suggests BSgenome.Hsapiens.UCSC.hg19

Author Chao He

Maintainer Chao He <hechao2010@tsinghua.org.cn>

Description
Hi-C dataset processing algorithm introduced by Chao He (2016) <doi.org/10.1101/083576>.
The optimized and flexible pipeline for analyzing the
folding of whole chromosome and interactions between some
specific sites from the Hi-C raw sequencing reads to the
partially processed datasets. The other complex genetic and
epigenetic datasets from public sources such as GWAS, ENCODE
consortiums etc. will also easily be integrated into HBP,
hence the final output results of HBP could provide a
comprehensive in-depth understanding for the specific
chromatin interactions, potential molecular mechanisms and
biological significance. We believe that HBP is a reliable
tool for the rapidly analysis of Hi-C data and will be very
useful for a wide range of researchers, particularly those
who lack of background in computational biology.

LazyData TRUE

LazyLoad yes

Repository CRAN

NeedsCompilation no

Date/Publication 2017-07-07 05:01:44 UTC

R topics documented:

`calculate_omiccircos_data` *output the interaction that our bed interact with ourbed.*

Description

output the interaction that our bed interact with ourbed.

Arguments

`cmap`

`bed_matrix`

`bed_bin`

`bedfile`

`chr`

`st_cmap`

`m_threshold`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (cmap, bed_matrix, bed_bin, bedfile, chr = "chr4", st_cmap,
    m_threshold = 0)
{
    n = dim(cmap)[1]
    bed_count = 0
    nm_count = 0
    all_count = 0
    cccc = 0
    ooo <- data.frame(chr_nm1 = character(0), n_start1 = numeric(0),
        bed_seq1 = character(0), chr_nm2 = character(0), n_start2 = numeric(0),
        bed_seq2 = character(0), read_count = numeric(0), stringsAsFactors = FALSE)
    ooo = lapply(rbind(1:(dim(bed_matrix)[1] - 1)), calculate_omiccircos_data_solo,
        cmap = cmap, bed_matrix = bed_matrix, bed_bin = bed_bin,
        bedfile = bedfile, chr = chr, st_cmap = st_cmap, m_threshold = m_threshold)
    final_ooo <- data.frame(chr_nm1 = character(0), n_start1 = numeric(0),
        bed_seq1 = character(0), chr_nm2 = character(0), n_start2 = numeric(0),
        bed_seq2 = character(0), read_count = numeric(0), stringsAsFactors = FALSE)
    for (tt in 1:(dim(bed_matrix)[1] - 1)) {
        if (!is.null(ooo[tt][[1]])) {
            final_ooo = rbind(final_ooo, as.data.frame(ooo[tt][[1]]))
        }
    }
    colnames(final_ooo) = c("chr_nm1", "n_start1", "bed_seq1",
        "chr_nm2", "n_start2", "bed_seq2", "read_count")
    return(final_ooo)
  }
```

---

calculate_omiccircos_data_solo

help calculate_omiccircos_data to finish its work

Description

help calculate_omiccircos_data to finish its work

Arguments

- i
- cmap
- bed_matrix
- bed_bin
- bedfile

chr

st_cmap

m_threshold

Author(s)

Chao He

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (i, cmap, bed_matrix, bed_bin, bedfile, chr = "chr4",
    st_cmap, m_threshold = 0)
{
    jj = which(cmap[i, (i + 1):(dim(bed_matrix)[1])] > m_threshold)
    jjj = jj + i
    ooo = NULL
    if (bed_matrix[i] != 0) {
        jjjj = which(bed_matrix[jjj] != 0)
        jjjjj = jjj[jjjj]
        j_length = length(jjjjj)
        if (j_length >= 1) {
            for (k in 1:4) {
                m_all = which(bed_bin[, k] == i)
                m_all_length = length(m_all)
                if (m_all_length > 0) {
                  for (mn in 1:m_all_length) {
                    m = m_all[mn]
                    if (bedfile[m, 1] == chr) {
                      for (hh in 1:j_length) {
                        mj = jjjjj[hh]
                        for (kk in 1:4) {
                          mm_all = which(bed_bin[, kk] == mj)
                          mm_all_length = length(mm_all)
                          if (mm_all_length > 0) {
                            for (mmn in 1:mm_all_length) {
                              mm = mm_all[mmn]
                              if (bedfile[mm, 1] == chr) {
                                ooo = rbind(ooo, cbind(bedfile[m,
                                  1], bedfile[m, 2], bedfile[m,
                                  4], bedfile[mm, 1], bedfile[mm,
                                  2], bedfile[mm, 4], st_cmap[i,
                                  mj]))
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
            }
```

```
                    }
                  }
                }
            }
        }
    }
    return(ooo)
}
```

---

check_bed_bin *process the bed file data to bin*

Description

process the bed file data to bin

Usage

```
check_bed_bin(m_bed, bin = 2000)
```

Arguments

m_bed

bin

Author(s)

Chao He

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (m_bed, bin = 2000)
{
    ooo = matrix(data = 0, nrow = dim(m_bed)[1], ncol = 4)
    for (i in 1:dim(m_bed)[1]) {
        st = (ceiling(m_bed[i, 2]/bin))
        ed = (ceiling(m_bed[i, 3]/bin))
        st1 = (ceiling((m_bed[i, 2] - 500)/bin))
        ed1 = (ceiling((m_bed[i, 3] + 500)/bin))
        if (st != ed) {
            ooo[i, 2] = st
            ooo[i, 3] = ed
            if (st1 != st) {
                ooo[i, 1] = st1
```

```
            }
            if (ed1 != ed) {
                ooo[i, 4] = ed1
            }
        }
        else {
            ooo[i, 2] = st
            if (st1 != st) {
                ooo[i, 1] = st1
            }
            if (ed1 != ed) {
                ooo[i, 4] = ed1
            }
        }
    }
    return(ooo)
}
```

---

choose_chr_bed *choose which chrom from the BED file*

Description

choose which chrom from the BED file

Usage

```
choose_chr_bed(m_bed, chrom_list = "chr2L")
```

Arguments

`m_bed`

`chrom_list`

Author(s)

Chao He

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (m_bed, chrom_list = "chr2L")
{
    sec = NULL
    for (j in 1:length(chrom_list)) {
```

```
        sec = rbind(sec, m_bed[m_bed$chr == chrom_list[j], ])
    }
    return(sec)
  }
```

`circos_plot` *Visualization of interactions and tracks*

Description

help user plot the circos picture with tracks

Arguments

`bedFile`	the path of specific sites, should use BED format
`wig_dir`	the path of tracks file, which should be WIG or BEDGRAPH format
`matrix_dir`	the path of matrix, which is output by function generate_matirx()
`bedWindow`	the window of sites, Default is 0
`outputpdf`	the picture format of result, if set FALSE, result will be plotted at JPEG format, if TRUE, result will be ploted at PDF format
`chrom`	which chromatin will be analysis, Default is all
`chrstart`	where will start the analysis in the chromatin, default = 0
`chrend`	where will end the analysis in the chromatin, when set 0, the whole chromatin will be analysed. default = 0
`resolution`	the resolution of Hi-C dataset.
`circosLineWidth`	circos line width. default = 0.01
`circosLinecolor`	circos line color, if choose rainbow will use random color. default = rainbow
`if_threshold`	the threshold of interacion readcounts, default = 0
`circosTrackWidth`	circos Track width. default = 40

Details

The package can visualize the interaction of interested sequences. It is compatible with the adding of some tracks such as histone modifications or transcription factors binding sites enrichment. From the graph, users can easily find out the association rules of the chromatin interactions and other genetic and epigenetic features of the interested sequences, put forward hypotheses and further design related experiments to verify.

can be used as following command: circos_plot(bedFile,wig_dir="wig",matrix_dir="CTCF_dm3",chrom="chr2L",chrstart=1
circos_plot(bedFile,wig_dir="wig",matrix_dir="CTCF_dm3",bedWindow=0,outputpdf="TRUE",chrom="chr2L",chrstart=1

Author(s)

Chao He

convert_bed_to_matrix *convert the bed file to matrix file for the next analysis*

Description

convert the bed file to matrix file for the next analysis

Arguments

bedfile

bin

chrom_list

tot_size

bed_window

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (bedfile, bin = 2000, chrom_list = "chr2L", tot_size = 0,
    bed_window = 2000)
{
    if (tot_size == 0) {
        print("please input tot_size")
    }
    else {
        cmap = matrix(data = 0, nrow = tot_size, ncol = 1)
        sec = NULL
        for (j in 1:length(chrom_list)) {
            sec = rbind(sec, bedfile[bedfile$chr == chrom_list[j],
                ])
        }
        ordereduniques = sec
        count = 0
        for (i in 1:dim(ordereduniques)[1]) {
            st = (ceiling(ordereduniques[i, 2]/bin))
            ed = (ceiling(ordereduniques[i, 3]/bin))
            st1 = (ceiling((ordereduniques[i, 2] - bed_window)/bin))
            ed1 = (ceiling((ordereduniques[i, 3] + bed_window)/bin))
            if (st < 1) {
                st = 1
            }
            if (st1 < 1) {
                st1 = 1
            }
```

```
            if (ed < 1) {
                ed = 1
            }
            if (ed1 < 1) {
                ed1 = 1
            }
            if (st > tot_size) {
                st = tot_size
            }
            if (st1 > tot_size) {
                st1 = tot_size
            }
            if (ed > tot_size) {
                ed = tot_size
            }
            if (ed1 > tot_size) {
                ed1 = tot_size
            }
            if (st != ed) {
                cmap[st, 1] = cmap[st, 1] + 1
                count = count + 1
                cmap[ed, 1] = cmap[ed, 1] + 1
                count = count + 1
                if (st1 != st) {
                  cmap[st1, 1] = cmap[st1, 1] + 1
                  count = count + 1
                }
                if (ed1 != ed) {
                  cmap[ed1, 1] = cmap[ed1, 1] + 1
                  count = count + 1
                }
            }
            else {
                cmap[ed, 1] = cmap[ed, 1] + 1
                count = count + 1
                if (st1 != st) {
                  cmap[st1, 1] = cmap[st1, 1] + 1
                  count = count + 1
                }
                if (ed1 != ed) {
                  cmap[ed1, 1] = cmap[ed1, 1] + 1
                  count = count + 1
                }
            }
        }
    }
    cat(sprintf("Mapped Fragments: %s\n", count))
    return(cmap)
}
```

`find_bed_to_bed_interaction` *find the interaction between specific sites*

Description

find the interaction between specific sites.

Arguments

`cmap`

`bed_matrix`

`bed_bin`

`bedfile`

`chr`

`st_cmap`

`m_threshold`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (cmap, bed_matrix, bed_bin, bedfile, chr = "chr4", st_cmap,
    m_threshold = 0)
{
    n = dim(cmap)[1]
    bed_count = 0
    nm_count = 0
    all_count = 0
    cccc = 0
    ooo <- data.frame(bed_id1 = numeric(0), chr_nm1 = character(0),
        n_start1 = numeric(0), n_end1 = numeric(0), n_location1 = numeric(0),
        bed_seq1 = character(0), bed_id2 = numeric(0), chr_nm2 = character(0),
        n_start2 = numeric(0), n_end2 = numeric(0), n_location2 = numeric(0),
        bed_seq2 = character(0), read_count = numeric(0), bed_bin1 = numeric(0),
        bed_bin2 = numeric(0), stringsAsFactors = FALSE)
    ooo = lapply(rbind(1:(dim(bed_matrix)[1] - 1)), find_bed_to_bed_interaction_solo,
        cmap = cmap, bed_matrix = bed_matrix, bed_bin = bed_bin,
        bedfile = bedfile, chr = chr, st_cmap = st_cmap, m_threshold = m_threshold)
    final_ooo <- data.frame(bed_id1 = numeric(0), chr_nm1 = character(0),
        n_start1 = numeric(0), n_end1 = numeric(0), n_location1 = numeric(0),
        bed_seq1 = character(0), bed_id2 = numeric(0), chr_nm2 = character(0),
        n_start2 = numeric(0), n_end2 = numeric(0), n_location2 = numeric(0),
        bed_seq2 = character(0), read_count = numeric(0), bed_bin1 = numeric(0),
        bed_bin2 = numeric(0), stringsAsFactors = FALSE)
    for (tt in 1:(dim(bed_matrix)[1] - 1)) {
```

```
        if (!is.null(ooo[tt][[1]])) {
            final_ooo = rbind(final_ooo, as.data.frame(ooo[tt][[1]]))
        }
    }
    colnames(final_ooo) = c("bed_id1", "chr_nm1", "n_start1",
        "n_end1", "n_location1", "bed_seq1", "bed_id2", "chr_nm2",
        "n_start2", "n_end2", "n_location2", "bed_seq2", "read_count",
        "bed_bin1", "bed_bin2")
    final_ooo[, 3] = as.numeric(final_ooo[, 3])
    final_ooo[, 4] = as.numeric(final_ooo[, 4])
    final_ooo[, 10] = as.numeric(final_ooo[, 10])
    final_ooo[, 9] = as.numeric(final_ooo[, 9])
    final_ooo[, 13] = as.numeric(final_ooo[, 13])
    return(final_ooo)
  }
```

`find_bed_to_bed_interaction_solo` *help find_bed_to_bed_interaction finish its work*

Description

help find_bed_to_bed_interaction finish its work

Arguments

`i`

`cmap`

`bed_matrix`

`bed_bin`

`bedfile`

`chr`

`st_cmap`

`m_threshold`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (i, cmap, bed_matrix, bed_bin, bedfile, chr = "chr4",
    st_cmap, m_threshold = 0)
{
    jj = which(cmap[i, (i + 1):(dim(bed_matrix)[1])] > m_threshold)
    jjj = jj + i
```

```
    ooo = NULL
    if (bed_matrix[i] != 0) {
        jjjj = which(bed_matrix[jjj] != 0)
        jjjjj = jjj[jjjj]
        j_length = length(jjjjj)
        if (j_length >= 1) {
            for (k in 1:4) {
                m_all = which(bed_bin[, k] == i)
                m_all_length = length(m_all)
                if (m_all_length > 0) {
                  for (mn in 1:m_all_length) {
                    m = m_all[mn]
                    if (bedfile[m, 1] == chr) {
                      for (hh in 1:j_length) {
                        mj = jjjjj[hh]
                        for (kk in 1:4) {
                          mm_all = which(bed_bin[, kk] == mj)
                          mm_all_length = length(mm_all)
                          if (mm_all_length > 0) {
                            for (mmn in 1:mm_all_length) {
                              mm = mm_all[mmn]
                              if (bedfile[mm, 1] == chr) {
                                ooo = rbind(ooo, cbind(m, bedfile[m,
                                  1], bedfile[m, 2], bedfile[m,
                                  3], k, bedfile[m, 4], mm, bedfile[mm,
                                  1], bedfile[mm, 2], bedfile[mm,
                                  3], kk, bedfile[mm, 4], st_cmap[i,
                                  mj], i, mj))
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
            }
        }
    }
    return(ooo)
}
```

---

generate_enzyme_file *generate enzyme file*

Description

help user generate enzyme file

Arguments

enzyme	the restriction enzyme name.
enzymesite	the restriction enzyme cutting sites
chrom_file	the chromatin information file
enzymedir	the output dir of this function
enzymeoverhangs5	5' overhangs on the DNA resulted from the cutting
genomeName	the name of genome

Details

this function is used to generate restriction fragment annotation files. and some annotation files were generated in https://github.com/hechao0407/HBP/tree/master/annotation

can be used as following command: generate_enzyme_file(enzyme="DpnII",enzymesite="GATC",chrom_file="chrom_dm3.

---

generate_matrix *generate matrix files*

Description

generate matrix files from HiC-Pro result can be used as following command: generate_matrix(all_hic_file="demo.matrix",all

Arguments

all_hic_file	the path of hic matrix file generated by HiC-Pro, can be found in the ../hic_result/matrix/myhic/iced/resolu
all_bed_file	the path of hic index file generated by HiC-Pro, can be found in the ../hic_result/matrix/myhic/raw/resoluti
outputpdf	the picture format of result, if set FALSE, result will be plotted at JPEG format, if TRUE, result will be ploted at PDF format
matrix_dir	the output path of this function.
chrom_file	the chromatin information file

`if_distribution_analysis` *analyse the interaction frequency distribution*

Description

analyse the interaction frequency distribution

Arguments

`all_hic_file`	the path of hic matrix file generated by HiC-Pro, can be found in the ../hic_result/matrix/myhic/iced/resolu
`all_bed_file`	the path of hic index file generated by HiC-Pro, can be found in the ../hic_result/matrix/myhic/raw/resoluti
`bedFile`	the path of specific sites, should use BED format
`matrix_dir`	the path of matrix, which is output by function generate_matirx()
`inter_chromfile`	the inter-chromatin interaction file generated by generate_matrix(), when set NULL, it will be set as matrix_dir/inter_chrom.iam.
`groupNum`	the random group number. Default is 50
`random_analysis`	Whether run random analysis, Default is TRUE
`threshold_percent`	the up-threshold of Hi-C interactions, part of the largest interaction number will be deleted. Default is 0.005
`if_bin_number`	interacion frequency bin number. all interacions will be sort to these bins according to their interacion number. Default is 20
`outputpdf`	the picture format of result, if set FALSE, result will be plotted at JPEG format, if TRUE, result will be ploted at PDF format
`resolution`	the resolution of Hi-C dataset.
`chrom_file`	the chromatin information file
`slide_window`	if set TRUE, there will use a slide window to make picture smooth

Details

HBP sets up and down thresholds, and discards the interaction-pairs whose frequencies are lower than the down threshold or higher than the up threshold. Then HBP separates chromatin interaction frequencies into many bins (referred to as frequency bin), and computes density for specific sites in each frequency bins. Meanwhile, HBP will compare the results with that of random control groups which were generated according to the distribution of these imported specific sites. Based on these imported specific region, every member of random control groups was calculated and made in the similar region. This comparison results make it easier for users to ascertain whether the observed interactions were biologically significant and whether these specific sites have effect on the conformation of chromatin or not.

can be used as following command: if_distribution_analysis(all_hic_file="demo.matrix",all_bed_file="demo.bed",bedFile="

if_distribution_analysis(all_hic_file="demo.matrix",all_bed_file="demo.bed",bedFile="dm3_mars.bed",inter_chromfile=NU

load_all_wig *the function to load wig file*

Description

the function to load wig file

Usage

```
load_all_wig(wigfile)
```

Arguments

```
wigfile
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (wigfile)
{
    uniques <- read.table(file = wigfile, fill = TRUE, stringsAsFactors = FALSE,
        skip = 1)
    uniques <- uniques[, c(1, 2, 3, 4)]
    colnames(uniques) <- c("chr", "start", "end", "value")
    if (dim(uniques)[1] == 0) {
        stop("No data loaded! \n\n")
    }
    else {
        return(uniques)
    }
  }
```

load_bed *the function to load bed file*

Description

the function to load bed file

Usage

```
load_bed(bedfile)
```

Arguments

bedfile

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (bedfile)
{
    uniques <- read.table(file = bedfile, fill = TRUE, stringsAsFactors = FALSE,
        skip = 1)
    uniques <- uniques[, c(1, 2, 3, 4)]
    colnames(uniques) <- c("chr", "start", "end", "name")
    uniques <- na.omit(uniques)
    ordereduniques <- uniques[with(uniques, order(chr, as.numeric(start))),
        ]
    if (dim(ordereduniques)[1] == 0) {
        stop("No data loaded! \n\n")
    }
    else {
        return(ordereduniques)
    }
  }
```

---

load_bed_wig *the function to load wig file in the specific sites*

Description

the function to load wig file in the specific sites

Usage

```
load_bed_wig(wigfile, chrBed, chrom, chrstart, chrend, m_win)
```

Arguments

wigfile

chrBed

chrom

chrstart

chrend

m_win

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (wigfile, chrBed, chrom, chrstart, chrend, m_win)
{
    all_wig = load_all_wig(wigfile)
    all_wig = all_wig[all_wig$chr == chrom, ]
    all_wig[, 2] = as.numeric(all_wig[, 2])
    all_wig[, 3] = as.numeric(all_wig[, 3])
    m_result = data.frame(chrom = character(0), start = numeric(0),
        end = numeric(0), wig_value = numeric(0), stringsAsFactors = FALSE)
    dim_bed = dim(chrBed)[1]
    m_result[1:dim_bed, 1:3] = chrBed[, 1:3]
    if (chrend > 0) {
        all_wig = all_wig[which(all_wig[, 3] < chrend), ]
    }
    if (chrstart > 0) {
        all_wig[, 2] = all_wig[, 2] - chrstart
        all_wig[, 3] = all_wig[, 3] - chrstart
        all_wig = all_wig[which(all_wig[, 2] >= 0), ]
    }
    for (i in 1:dim_bed) {
        tmp_bed_wig = which(all_wig[, 2] >= chrBed[i, 2] - m_win)
        tmp_bed_wig = tmp_bed_wig[which(all_wig[tmp_bed_wig,
            3] <= chrBed[i, 3] + m_win)]
        m_result[i, 4] = mean(as.numeric(all_wig[tmp_bed_wig,
            4]))
    }
    return(m_result)
  }
```

---

load_wig *the function to load tracks*

Description

the function to load tracks

Usage

```
load_wig(wigfile, resolution = 2000, chrom = "chr2L", chrTotSize, chrstart, chrend)
```

Arguments

wigfile

resolution

chrom

chrTotSize

chrstart

chrend

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (wigfile, resolution = 2000, chrom = "chr2L", chrTotSize,
    chrstart, chrend)
{
    tmp_wig <- read.table(file = wigfile, fill = TRUE, stringsAsFactors = FALSE,
        skip = 1)
    ooo <- data.frame(seg.name = character(chrTotSize), seg.po = numeric(chrTotSize),
        value = numeric(chrTotSize), stringsAsFactors = FALSE)
    ooo_num <- data.frame(wig_num = numeric(chrTotSize))
    sec = NULL
    sec = rbind(sec, tmp_wig[tmp_wig[, 1] == chrom, ])
    sec[, 2] = as.numeric(sec[, 2])
    sec[, 3] = as.numeric(sec[, 3])
    sec[, 4] = as.numeric(sec[, 4])
    if (chrend > 0) {
        sec = sec[which(sec[, 3] < chrend), ]
        sec[, 2] = sec[, 2] - chrstart
        sec[, 3] = sec[, 3] - chrstart
        sec = sec[which(sec[, 2] > 0), ]
    }
    secdim = dim(sec)[1]
    if (is.null(secdim)) {
        return(NULL)
    }
    else if (secdim < 2) {
        return(NULL)
    }
    else {
        ooo[, 1] = chrom
        ooo[, 2:3] = 0
        ooo_num[, 1] = 0
        wig_bin = 1
        sec_num = dim(sec)[1]
        for (i in 1:sec_num) {
            wig_bin = ceiling((sec[i, 2] + sec[i, 3])/(2 * resolution))
            if (wig_bin > chrTotSize) {
                wig_bin = chrTotSize
            }
            ooo[wig_bin, 3] = ooo[wig_bin, 3] + sec[i, 4]
            ooo_num[wig_bin, 1] = ooo_num[wig_bin, 1] + 1
            ooo[wig_bin, 2] = wig_bin
```

```
        }
        for (i in 1:chrTotSize) {
            if (ooo[i, 2] == 0) {
                ooo[i, 2] = i
            }
            if (ooo_num[i, 1] > 0) {
                ooo[i, 3] = (ooo[i, 3])/(ooo_num[i, 1])
            }
            else {
                ooo[i, 3] = 0
            }
        }
        return(ooo)
    }
}
```

`network_analysis` *Interaction network topological analysis*

Description

help user plot interaction network picture and make clusters

Arguments

`bedFile`	the path of specific sites, should use BED format
`matrix_dir`	the path of matrix, which is output by function generate_matirx()
`outputpdf`	the picture format of result, if set FALSE, result will be plotted at JPEG format, if TRUE, result will be ploted at PDF format
`chrom`	which chromatin will be analysis, Default is all
`chrstart`	where will start the analysis in the chromatin, default = 0
`chrend`	where will end the analysis in the chromatin, when set 0, the whole chromatin will be analysed. default = 0
`resolution`	the resolution of Hi-C dataset.
`bedWindow`	the window of sites, Default is 0
`netplot`	draw the network plot,when node number is too much,it may be not working. default = TRUE
`net_layout`	the layout of net plot, can be choose from layout.auto,layout.circle and layout.fruchterman.reingold. default = layout.fruchterman.reingold
`NetClusterType`	the method of topological clustering, can be choose from NULL,multileve,edgeBetweenness,walktrap,lab default = multileve
`NetVertexSize`	network vertex size. default = 2
`NetVertexChangeSize`	the parameter to change the node vertex, can be choose from NULL,degree,closeness,betweenness,Local_ default = degree

`NetVertexLableDist`
: distance between label and vertex in the network. default = 0.1

`NetVertexColor`
: the color of vertex. default = #7fbc41

`NetVertexLabelCex`
: the size of network vertex label. default = 0.3

`if_threshold`
: the threshold of interacion readcounts, default = 0

Details

HBP maps sequence sites of interest to the probability matrix, and calculate the interaction frequency and pinpoint the interaction networks mediated by these sites. The igraph package was used to plot an interaction network graph and make topological cluster analysis. According to the clustering results, users can classify all sites of interest into different types or clusters, and examine their potential differential properties respectively.

can be used as following command: network_analysis(bedFile="CTCF_hg19_encodeCluster_GM12878.bed",matrix_dir="C

network_analysis(bedFile="CTCF_hg19_encodeCluster_GM12878.bed",matrix_dir="CTCF_GM12878",outputpdf=TRUE

---

run_hicpro *run the hic-pro*

Description

help users to map and normalize Hi-C raw dataset by HiC-Pro

Arguments

`hicpro_path`
: if the path of HiC-Pro is already added to environment PATH, you can use hicpro_path = "HiC-Pro"

`inputfile`
: the raw fastq data file folder. for example, my Hi-C data sequencing file is myhic_R1.fastq and myhic_R2,fastq, they should be placed in ../rawdata/myhic/myhic_R1.fastq and ../rawdata/myhic/myhic_R2.fastq.

`configfile`
: the HiC-Pro config file, there is a example file in https://github.com/hechao0407/HBP/blob/master/config-hicpro.txt.

`outdir`
: the output folder of this function

Details

this function is used to help users to map and normalize Hi-C raw dataset by HiC-Pro. HiC-Pro can be download at http://github.com/nservant/HiC-Pro

statistical_analysis *Statistical significance tests*

Description

help user make statistical analysis and make clusters

Arguments

bedFile	the path of specific sites, should use BED format
wig_dir	the path of tracks file, which should be WIG or BEDGRAPH format
bedWindow	the window of sites, Default is 0
matrix_dir	the path of matrix, which is output by function generate_matirx()
outputpdf	the picture format of result, if set FALSE, result will be plotted at JPEG format, if TRUE, result will be ploted at PDF format
chrom	which chromatin will be analysis, Default is all
chrstart	where will start the analysis in the chromatin, default = 0
chrend	where will end the analysis in the chromatin, when set 0, the whole chromatin will be analysed. default = 0
resolution	the resolution of Hi-C dataset.
groupNum	the random group number. Default is 100
dist_method	the method to calculater the statistical distance, can be chosen from manhattan,euclidean,minkowski,chebyshev,mahalanobis,canberra. default = euclidean
clust_method	the method to make tracks enrichment clusters , can be chosen from average,centroid,median,complete,sin default = complete
clust_label	add labels at the cluster tree picture or not. default = FALSE
clust_k	the cluster number to make. default = 4
threshold	analysis threshold. default = 0
hm_trace	whether plot the trace in the heatmap, default = TRUE

Details

HBP can perform cluster analysis using histone modifications or transcription factors binding tracks, and make Kruskal-Wallis rank sum or T test to evaluate the statistical difference of interaction frequency, or the interaction strength of the specific sites respectively. These tests will tell us whether there exist significant differences, suggesting different properties of these sites and the interactions.

can be used as following command: statistical_analysis(bedFile="dm3_mars.bed",wig_dir="wig",matrix_dir="CTCF_dm3",

statistical_analysis(bedFile="dm3_mars.bed",wig_dir="wig",bedWindow=0,matrix_dir="CTCF_dm3",outputpdf=TRUE,ch

whitered *generate the color of heatmap*

Description

generate the color of heatmap

Arguments

n

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (n)
{
    colorpanel(n, "white", "red")
  }
```

Index