Package ‘geneHapR’

April 9, 2023

Type Package

Title Gene Haplotype Statistics, Phenotype Association and Visualization

Description Import genome variants data and perform gene haplotype Statistics, visualization and phenotype association with 'R'.

biocViews NucleosomePositioning, DataImport

Encoding UTF-8

Maintainer Zhang Renliang <zhang_renliang@163.com>

Version 1.1.9

RoxygenNote 7.2.3

LazyData True

Imports ape, Biostrings, ggpubr, genetics, GenomicRanges, lolliplot, maps, methods, IRanges, pegas, reshape2, rlang, rtracklayer, shiny, shinyjs, stats, stringdist, stringr, tibble, tidyr, utils, vcfR

Depends ggplot2, R (>= 4.0.0)

Suggests mapdata, maptools, muscle, knitr, rmarkdown, testthat (>= 3.0.0)

License GPL-3

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Zhang Renliang [aut, cre], Jia Guanqing [aut]

Repository CRAN

Date/Publication 2023-04-09 18:00:02 UTC
## R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addINFO</td>
<td>3</td>
</tr>
<tr>
<td>addPromoter</td>
<td>4</td>
</tr>
<tr>
<td>ashaplotype</td>
<td>4</td>
</tr>
<tr>
<td>DataSet</td>
<td>5</td>
</tr>
<tr>
<td>displayVarOnGeneModel</td>
<td>5</td>
</tr>
<tr>
<td>filterLargeep.link</td>
<td>6</td>
</tr>
<tr>
<td>filterLargeVCF</td>
<td>8</td>
</tr>
<tr>
<td>filter_hap</td>
<td>9</td>
</tr>
<tr>
<td>filter_hmp</td>
<td>10</td>
</tr>
<tr>
<td>filter_plink.pedmap</td>
<td>11</td>
</tr>
<tr>
<td>filter_table</td>
<td>12</td>
</tr>
<tr>
<td>filter_vcf</td>
<td>13</td>
</tr>
<tr>
<td>getGenePOS</td>
<td>14</td>
</tr>
<tr>
<td>getGeneRanges</td>
<td>15</td>
</tr>
<tr>
<td>hap2hmp</td>
<td>16</td>
</tr>
<tr>
<td>hapDistribution</td>
<td>17</td>
</tr>
<tr>
<td>hapVsPheno</td>
<td>19</td>
</tr>
<tr>
<td>hapVsPhenoPerSite</td>
<td>23</td>
</tr>
<tr>
<td>hapVsPhenos</td>
<td>24</td>
</tr>
<tr>
<td>hap_summary</td>
<td>26</td>
</tr>
<tr>
<td>import_AccINFO</td>
<td>27</td>
</tr>
<tr>
<td>import_bed</td>
<td>28</td>
</tr>
<tr>
<td>import_gff</td>
<td>29</td>
</tr>
<tr>
<td>import_hap</td>
<td>30</td>
</tr>
<tr>
<td>import_MultipleAlignment</td>
<td>31</td>
</tr>
<tr>
<td>import_plink.pedmap</td>
<td>31</td>
</tr>
<tr>
<td>import_seqseqs</td>
<td>32</td>
</tr>
<tr>
<td>import_vcf</td>
<td>33</td>
</tr>
<tr>
<td>LDheatmap</td>
<td>34</td>
</tr>
<tr>
<td>network</td>
<td>37</td>
</tr>
<tr>
<td>plink_pedmap2hap</td>
<td>38</td>
</tr>
<tr>
<td>plotEFF</td>
<td>39</td>
</tr>
<tr>
<td>plotHapNet</td>
<td>41</td>
</tr>
<tr>
<td>plotHapTable</td>
<td>43</td>
</tr>
<tr>
<td>seqs2hap</td>
<td>46</td>
</tr>
<tr>
<td>SetATGAs0</td>
<td>47</td>
</tr>
<tr>
<td>siteEFF</td>
<td>48</td>
</tr>
<tr>
<td>table2hap</td>
<td>49</td>
</tr>
<tr>
<td>vcf2hap</td>
<td>50</td>
</tr>
<tr>
<td>write.hap</td>
<td>51</td>
</tr>
</tbody>
</table>

## Index

<table>
<thead>
<tr>
<th>Page</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td></td>
</tr>
</tbody>
</table>
addINFO

Add Information to Haplotype Results

Description
add annotations to INFO fields used for plotHapTable()

Usage
addINFO(hap,
tag = "", values = values,
replace = FALSE, sep = ";")
sites(hap)

Arguments
 hap object of hapResult or hapSummary class
tag tag names, usually is a single word used before "=
values annotation for each site. Length of values must be equal with sites in hapResult
replace whether replace origin INFOs in hapResult or not. Default as FALSE
sep a character string to separate the terms. Not NA_character_

Value
object of hapSummary or hapResult class with added/replaced INFOs

See Also
plotHapTable()
plotHapTable()

Examples
data("geneHapR_test")

# length of values must be equal with number of sites in hap result
values <- paste0("newInfo",c(1:9))
 hapResult <- addINFO(hapResult, tag = "new", values = values, replace = TRUE)
data("geneHapR_test")

# check how many sites were concluded in hapResult/hapSummary
 sites(hapResult)
addPromoter  
add promoter to annotation

Description
add promoter to annotation

Usage
addPromoter(anno, PromoterLength = 1500, bedFile = NULL)

Arguments
anno annotation, imported gff/bed
PromoterLength the length of promoter region, default as 1500
bedFile the output bed file name

Examples
data("geneHapR_test")
bed <- addPromoter(gff)

ashaplotype  as.haplotype

Description
convert hapSummary or hapResult class into haplotype class (pegas)

Usage
as.haplotype(hap)

Arguments
hap object of hapSummary or hapResult class

Value
haplotype class

Note
It's not advised for hapSummary or hapResult with indels, due to indels will convert to SNPs with equal length of each indel.
Examples

data("geneHapR_test")
hap <- as.haplotype(hapResult)
hapSummary <- hap_summary(hapResult)
hap <- as.haplotype(hapSummary)

DataSet

Datasets gff contains a example of gff file used for test of visualization mutations on gene model.

Description

pheno contains a simulated test pheno data used for test of comparison between different haps
vcf, a vcfR object provide a data set for test of seq2hap(). vcf contains indels, snps, biallelic sites and multiallelic sites.
AccINFO a data.frame provide additional information of accessions, including accession type, source and location.

displayVarOnGeneModel Display Variants on Gene Model

Description

show variants on gene model using hapSummary and gene annotations

Usage

displayVarOnGeneModel(
  hapSummary, gff, Chr, startPOS, endPOS, type = "pin", cex = 0.7, CDS_h = 0.05, fiveUTR_h = 0.02, threeUTR_h = 0.01, geneElement = geneElement, hap
)
filterLargep.link

Pre-process of Large VCF File(s)

Description

Filter/extract one or multiple gene(s)/range(s) from a large p.link file.

Usage

```r
filterLargeP.link(
  root,
  rootOut = rootOut,
  Chr = Chr,
  POS = NULL,
  start = start,
  end = end,
  override = TRUE,
  sep = "\t"
)
```
filterLargep.link

Arguments

- **root**: The file name without suffix. This function only support p.link file format stored in "map" and "ped" format, the file names after removed suffix should be same with each other.
- **rootOut**: Path(s) of output p.link file stored in "ped&map" format.
- **Chr**: a single CHROM name or CHROM names vector.
- **POS, start, end**: provide the chromosome name should be extract from original p.link dataset. POS: a vector consist with start and end position, eg.: c(1,200) indicates 3 ranges (1~200, 300~500 and 300~400). if POS is NULL, start and end are needed.
- **override**: whether override existed file or not, default as TRUE.
- **sep**: a character indicate the separation of map and ped file, default is \t.

Details

This package import P.link files. However, import a large P.link file is time and memory consuming. It’s suggested that extract variants in target range with filterLargeP.link() before identification of haplotype.

When filter/extract multi genes/ranges, the parameter of Chr and POS must have equal length. Results will save to a single file if the user provide a single file path or save to multiple P.link file(s) when a equal length vector consist with file paths is provided.

Value

No return value

Examples

```r
# The filteration of P.link of regular size should be done with `filter_plink.pedmap()`.
# however, here, we use a mini vcf instead just for example and test

pedfile <- system.file("extdata", "snp3kvars-CHR8-25947258-25951166-plink.ped", package = "geneHapR")
mapfile <- system.file("extdata", "snp3kvars-CHR8-25947258-25951166-plink.map", package = "geneHapR")

oldDir <- getwd()
temp_dir <- tempdir()
if(! dir.exists(temp_dir))
  dir.create(temp_dir)
setwd(temp_dir)
file.copy(pedfile, "test.ped")
file.copy(mapfile, "test.map")

# extract a single gene/range from large vcf
filterLargeP.link(root = "test",)
```
rootOut = "filtered_test",
Chr = "scaffold_1", POS = c(4300, 5000), override = TRUE)

setwd(oldDir)

# delete temp_dir
unlink(temp_dir, recursive = TRUE)

c

filterLargeVCF

Pre-process of Large VCF File(s)

Description
Filter/extract one or multiple gene(s)/range(s) from a large *.vcf/*.vcf.gz file.

Usage
filterLargeVCF(VCFin = VCFin, VCFout = VCFout,
   Chr = Chr,
   POS = NULL,
   start = start,
   end = end,
   override = TRUE)

Arguments
VCFin Path of input *.vcf/*.vcf.gz file.
VCFout Path(s) of output *.vcf/*.vcf.gz file.
Chr a single CHROM name or CHROM names vector.
POS, start, end provide the range should be extract from original vcf. POS: a vector consist with start and end position or a list with length equal to Chr. eg.: list(c(1, 200), c(300, 500), c(300, 400)) indicates 3 ranges (1~200, 300~500 and 300~400). if POS is NULL, start and end are needed, eg.: start = c(1, 30) and end = c(200, 150) indicates 2 ranges (1~200 and 30~150)
override whether override existed file or not, default as TRUE.

Details
This package import VCF files with 'vcfR' which is more efficient to import/manipulate VCF files in 'R'. However, import a large VCF file is time and memory consuming. It's suggested that filter/extract variants in target range with filterLargeVCF().

When filter/extract multi genes/ranges, the parameter of Chr and POS must have equal length. Results will save to a single file if the user provide a single file path or save to multiple VCF file(s) when a equal length vector consist with file paths is provided.

However, if you have hundreds gene/ranges need to extract from very large VCF file(s), it's prefer to process with other linux tools in a script on server, such as: 'vcftools' and 'bcftools'.
filter_hap

Value

No return value

Examples

# The filtration of small vcf should be done with 'filter_vcf()'.
# however, here, we use a mini vcf instead just for example and test.

vcfPath <- system.file("extdata", "var.vcf.gz", package = "geneHapR")

oldDir <- getwd()
temp_dir <- tempdir()
if(! dir.exists(temp_dir))
dir.create(temp_dir)
setwd(temp_dir)

# extract a single gene/range from large vcf
filterLargeVCF(VCFin = vcfPath, VCFout = "filtered.vcf.gz",
Chr = "scaffold_1", POS = c(4300,5000), override = TRUE)

# extract multi genes/ranges from large vcf
filterLargeVCF(VCFin = vcfPath,
VCFout = c("filtered1.vcf.gz",
"filtered2.vcf.gz",
"filtered3.vcf.gz"),
Chr = rep("scaffold_1", 3),
POS = list(c(4300, 5000),
c(5000, 6000),
c(5000, 7000)),
override = TRUE)

setwd(oldDir)

filter_hap

Filter hap

Description

filter hapResult or hapSummary by remove positions or accessions or haplotypes

Usage

filter_hap(hap,
rm.mode = c("position", "accession", "haplotype", "freq"),
position.rm = position.rm,
accession.rm = accession.rm,
haplotype.rm = haplotype.rm,
freq.min = 5)
filter_hmp

Arguments

- **hap**: object of hapSummary or hapResult class
- **rm.mode**: filter mode, one of "position", "accession", "haplotype"
- **position.rm**: numeric vector contains positions need to be removed
- **accession.rm**: character vector contains accessions need to be removed, only hapResult can be filtered by accessions
- **haplotype.rm**: character vector contains haplotypes need to be removed
- **freq.min**: numeric, haplotypes with accessions number less than freq.min will be removed

Value

hapSummary or hapResult depend input

Examples

```r
data("geneHapR_test")
hap <- filter_hap(hapResult,
  rm.mode = c("position", "accession", "haplotype", "freq"),
  position.rm = c(4879, 4950),
  accession.rm = c("C1", "C9"),
  haplotype.rm = c("H009", "H008"),
  freq.min = 5)
```

filter_hmp

filter variants in hapmap format

Description

filter variants in hapmap format

Usage

```r
filter_hmp(
  x,
  mode = c("POS", "type", "both"),
  Chr = Chr,
  start = start,
  end = end,
  gff = gff,
  type = type,
  cusTyp = cusTyp,
  geneID = geneID
)
```
**filter_plink.pedmap**

**Arguments**

- **x**: genotype dataset in hapmap format, object of data.frame class
- **mode**: filter mode, one of "POS", "type", "both"
- **Chr**: chromosome name, needed if mode set to "POS" or "both"
- **start**: start position, needed if mode set to "POS" or "both"
- **end**: end position, needed if mode set to "POS" or "both"
- **gff**: object of GRanges class, genome annotations imported by `import_gff()`
- **type**: filter type, needed if mode set to "type" or "both", one of "CDS", "exon", "gene", "genome", "custom", if type was set to "custom", then custom_type is needed.
- **cusTyp**: character vector, custom filter type, needed if type set to "custom"
- **geneID**: gene ID

**Examples**

```r
# create a dataset of genotype in hapmap format
hmp <- hap2hmp(hapResult);

# example
hmp <- filter_hmp(hmp, mode = "POS",
    Chr = "scaffold_1", start = 4100, end = 5000)
```

**Description**

used for filtration of p.link

**Usage**

`filter_plink.pedmap(x,`

```
mode = c("POS", "type", "both"),
Chr =Chr, start = start, end = end,
gff = gff, type = type, cusTyp = cusTyp,
geneID = geneID)
```

**Arguments**

- **x**: a list stored the p.link information
- **mode**: filtration mode, one of c("POS", "type", "both")
- **Chr**: the chromosome name, need if mode set as POS or both
- **start, end**: numeric, the range of filtration, and the start should smaller than end
- **gff**: the imported gff object
type should be in unique(gff$type), usually as "CDS", "genome".
cusTyp if type set as custom, then cusTyp is needed
geneID

Value
list, similar with x, but filtered

Examples

```r
pedfile <- system.file("extdata", "snp3kvars-CHR8-25947258-25951166-plink.ped", package = "geneHapR")
mapfile <- system.file("extdata", "snp3kvars-CHR8-25947258-25951166-plink.map", package = "geneHapR")
p.link <- import_plink.pedmap(pedfile = pedfile, mapfile = mapfile, sep_map = "\t", sep_ped = "\t")
p.link <- filter_plink.pedmap(p.link, mode = "POS", Chr = "chr08", start = 25948004, end = 25949944)
hapResult <- plink.pedmap2hap(p.link, hapPrefix = "H", hetero_remove = TRUE, na_drop = TRUE)
```

---

**Description**
filter variants stored in table

**Usage**

```r
filter_table(
  x,
  mode = c("POS", "type", "both"),
  Chr = Chr,
  start = start,
  end = end,
  gff = gff,
  type = type,
  cusTyp = cusTyp,
  geneID = geneID
)
```
### filter_vcf

**Filter VCF**

**Description**

filter VCF by GFF annotation or by position or both

**Usage**

```r
filter_vcf(vcf, gff = gff,
    mode = c("POS", "type", "both"),
    Chr = Chr, start = start, end = end,
    type = c("CDS", "exon", "gene", "genome", "custom"),
    cusTyp = c("CDS", "five_prime_UTR", "three_prime_UTR"),
    geneID = geneID)
```

**Arguments**

- `vcf`: object of vcfR class, VCF file imported by `import_vcf()`
- `gff`: object of GRanges class, genome annotations imported by `import_gff()`
- `mode`: filter mode, one of "POS", "type", "both"
- `Chr`: chromosome name, needed if mode set to "POS" or "both"
- `start`: start position, needed if mode set to "POS" or "both"
- `end`: end position, needed if mode set to "POS" or "both"
- `type`: filter type, needed if mode set to "type" or "both", one of "CDS", "exon", "gene", "genome", "custom", if type was set to "custom", then `cusTyp` is needed.
- `cusTyp`: character vector, custom filter type, needed if type set to "custom"
- `geneID`: character vector, gene ID
getGenePOS

**Description**

Get Gene Position

**Usage**

```r
getGenePOS(gff = gff,
            geneID = geneID,
            type = type,
            gffTermContaingeneID = "Parent")
```

**Arguments**

- **gff**: imported gff
- **geneID**: target geneID
- **type**: vector consist with one or more types in gff
- **gffTermContaingeneID**: which term contains the geneID in your gff, default is Parent
getGeneRanges

Description

Get Gene Ranges

Usage

getGeneRanges(gff, geneID, type, gffTermContainsGeneID)

data("geneHapR_test")
genePOS <- getGenePOS(gff = gff, geneID = "test1G0387", type = "CDS", gffTermContainsGeneID = "Parent")

geneRanges <- getGeneRanges(gff = gff, geneID = "test1G0387", type = "CDS", gffTermContainsGeneID = "Parent")

Arguments

- **gff**: imported gff
- **geneID**: target geneID
- **type**: vector consist with one or more types in gff
- **gffTermContainsGeneID**: which term contains the geneID in your gff, default is Parent

Value

GRanges

Examples

data("geneHapR_test")
geneRanges <- getGeneRanges(gff = gff, geneID = "test1G0387", type = "CDS", gffTermContainsGeneID = "Parent")


named vectors contains start, end and strand

Examples

data("geneHapR_test")
genePOS <- getGenePOS(gff = gff, geneID = "test1G0387", type = "CDS", gffTermContainsGeneID = "Parent")
**Description**

Convert hapResult object to hapmap (hmp) format, for interact with other packages

**Usage**

```r
hap2hmp(hap)

hmp2hap(hmp, hapPrefix = "H", hetero_remove = TRUE, na_drop = TRUE, ...)
```

**Arguments**

- `hap`: object of "hapResult" class
- `hmp`: object of "data.frame" class in hapmap format
- `hapPrefix`: prefix of haplotype names
- `hetero_remove`: whether remove accessions contains hyb-sites, Character not A T C G
- `na_drop`: whether drop accessions contains missing data ("N", "NA", ".")
- `...`: Arguments passed on to `table2hap`

x a data.frame contains variants information. The first file column are fix as Chrome name, position, reference nuclieotide, alter nuclieotide and INFO. Accession genotype should be in followed columns. "." will be treated as Indel. "." and "N" will be treated as missing data. Heterozygotes should be "A/T", "AAA/A"

**Pad** The number length in haplotype names should be extend to.

**Value**

a data.frame in hapmap format.

**Examples**

```r
data("geneHapR_test")
hmp <- hap2hmp(hapResult)
hap <- hmp2hap(hmp)
```
hapDistribution

Display of Geography Distribution

Description

show distribution of interest haplotypes on maps

Usage

hapDistribution(
  hap, 
  AccINFO, 
  LON.col, LAT.col, 
  hapNames, 
  database = "world", 
  regions = ".", 
  hap.color = hap.color, 
  zColours = zColours, 
  legend = TRUE, 
  symbolSize = 1, 
  symbol.lim = c(1, 10), 
  ratio = 1, 
  cex.legend = 0.8, 
  lwd.pie = 1, 
  borderCol.pie = NA, 
  lty.pie = 1, 
  showlabel = TRUE, 
  label.col = "black", 
  label.cex = 0.8, 
  label.font = 1, 
  label.adj = c(0.5, 0.5), 
  map.fill.color = 1, 
  ... 
)

Arguments

hap an object of hapResult class
AccINFO a data.frame contains accession information
LON.col, LAT.col column names of longitude(LON.col) and latitude(LAT.col)
 hapNames haplotype names used for display
database character string naming a geographical database, a list of x, y, and names obtained from a previous call to map or a spatial object of class SpatialPolygons or SpatialLines. The string choices include a world map, three USA databases
hapDistribution

(usa, state, county), and more (type help(package='maps') to see the package index). If the required database is in a different package that has not been attached, the string may be started with 'packagename::'. The location of the map databases may be overridden by setting the R_MAP_DATA_DIR environment variable.

regions  character vector that names the polygons to draw. Each database is composed of a collection of polygons, and each polygon has a unique name. When a region is composed of more than one polygon, the individual polygons have the name of the region, followed by a colon and a qualifier, as in michigan:north and michigan:south. Each element of regions is matched against the polygon names in the database and, according to exact, a subset is selected for drawing. The regions may also be defined using (perl) regular expressions. This makes it possible to use 'negative' expressions like "Norway(?!:Svalbard)", which means Norway and all islands except Svalbard. All entries are case insensitive. The default selects all polygons in the database.

hap.color, zColours  colors to apply to the pie section for each attribute column, "zColours" will be detached in future.

legend  a keyword specified the position of legend, one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center"; or a numeric vector of length two contains x,y coordinate of the legend

symbolSize  a numeric specified the symbol size. It will be detached in future. Please use "symbol.size" instead.

symbol.size  a numeric vector give the maximum and minimum size of each symbol

cex.legend  character expansion factor for legend relative to current par("cex")

lwd.pie  line width of the pies

borderCol.pie  The color of pie's border, default is NA, which means no border will be plotted

lty.pie  the line type of pie border

showlabel  a bool vector indicates whether show the labels which represents number of individuals. Default as TRUE.

label.col  color of the labels, default as "black"

label.cex  a number indicates the text size in label, default as 0.8

label.font  Font of label, 1 for normal, 2 for bold, 3 for italic, 4 for bold-italic

label.adj  the position of label, default as c(0.5, 0.5)

map.fill.color  vector of colors. If fill is FALSE, the first color is used for plotting all lines, and any other colors are ignored. Otherwise, the colors are matched one-one with the polygons that get selected by the region argument (and are reused cyclically, if necessary). If fill = TRUE, the default boundary line colour is given by par("fg"). To change this, you can use the border argument (see '...'). A color of NA causes the corresponding region to be deleted from the list of polygons to be drawn. Polygon colors are assigned after polygons are deleted due to values of the xlim and ylim arguments

Extra arguments passed to polygon or lines. Of particular interest may be the options border and lty that control the color and line type of the polygon borders when fill = TRUE.
hapVsPheno

Value

No return value

Examples

```r
data("geneHapR_test")
hapDistribution(hapResult, 
  AccINFO = AccINFO,  
  LON.col = "longitude",  
  LAT.col = "latitude",  
  hapNames = c("H001", "H002", "H003")
)```

Description

hapVsPheno

Usage

```r
hapVsPheno(
  hap,  
  pheno,  
  phenoName,  
  hapPrefix = "H",  
  title = "",  
  comparisons = comparisons,  
  method = "t.test",  
  method.args = list(),  
  symnum.args = list(),  
  mergeFigs = FALSE,  
  angle = angle,  
  hjust = hjust,  
  vjust = vjust,  
  minAcc = minAcc,  
  freq.min = freq.min,  
  outlier.rm = TRUE,  
  ...  
)
```

Arguments

- **hap**: object of hapResult class, generate with vcf2hap() or seqs2hap()
- **pheno**: object of data.frame class, imported by import_pheno()
hapVsPheno

phenoName

pheno name for plot, should be one column name of pheno

hapPrefix

prefix of haplotypes, default as "H"

title

a character which will used for figure title

comparisons

a list contains comparison pairs eg. list(c("H001", "H002"), c("H001", "H004")), or a character vector contains haplotype names for comparison, or "none" indicates do not add comparisons.

method

a character string indicating which method to be used for comparing means.

method.args

a list of additional arguments used for the test method. For example one might use method.args = list(alternative = "greater") for wilcoxon test.

symnum.args

a list of arguments to pass to the function symnum for symbolic number coding of p-values. For example, symnum.args <- list(cutpoints = c(0, 0.0001, 0.001, 0.01, 0.05, Inf), symbols = c("****", "***", "**", "*", "ns")).

In other words, we use the following convention for symbols indicating statistical significance:

• ns: p > 0.05
• *: p <= 0.05
• **: p <= 0.01
• ***: p <= 0.001
• ****: p <= 0.0001

mergeFigs

bool type, indicate whether merge the heat map and box plot or not. Default as FALSE

angle

the angle of x labels

hjust, vjust

hjust and vjust of x labels

minAcc, freq.min

If observations number of a Hap less than this number will not be compared with others or be plotted. Should not less than 3 due to the t-test will meaninglessly. Default as 5

outlier.rm

whether remove outliers, default as TRUE

Arguments passed on to ggpubr::ggviolin
data

a data frame

x

character string containing the name of x variable.

y

character vector containing one or more variables to plot

combine

logical value. Default is FALSE. Used only when y is a vector containing multiple variables to plot. If TRUE, create a multi-panel plot by combining the plot of y variables.

merge

logical or character value. Default is FALSE. Used only when y is a vector containing multiple variables to plot. If TRUE, merge multiple y variables in the same plotting area. Allowed values include also "asis" (TRUE) and "flip". If merge = "flip", then y variables are used as x tick labels and the x variable is used as grouping variable.

color

outline color.

fill

fill color.
palette  the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".

alpha  color transparency. Values should be between 0 and 1.

xlab  character vector specifying x axis labels. Use xlab = FALSE to hide xlab.

ylab  character vector specifying y axis labels. Use ylab = FALSE to hide ylab.

facet.by  character vector, of length 1 or 2, specifying grouping variables for faceting the plot into multiple panels. Should be in the data.

panel.labs  a list of one or two character vectors to modify facet panel labels. For example, panel.labs = list(sex = c("Male", "Female")) specifies the labels for the "sex" variable. For two grouping variables, you can use for example panel.labs = list(sex = c("Male", "Female"), rx = c("Obs", "Lev", "Lev2"))

short.panel.labs  logical value. Default is TRUE. If TRUE, create short labels for panels by omitting variable names; in other words panels will be labelled only by variable grouping levels.

linetype  line types.

trim  If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don’t trim the tails.

size  Numeric value (e.g.: size = 1). change the size of points and outlines.

width  violin width.

draw_quantiles  If not(NULL) (default), draw horizontal lines at the given quantiles of the density estimate.

select  character vector specifying which items to display.

remove  character vector specifying which items to remove from the plot.

order  character vector specifying the order of items.

add  character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_hilow", "median_q1q3", "median_mad", "median_range"; see ?desc_statby for more details.

add.params  parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").

error.plot  plot type used to visualize error. Allowed values are one of c("pointrange", "linerange", "crossbar", "errorbar", "upper_errorbar", "lower_errorbar", "upper_pointrange", "lower_pointrange", "upper_linerange", "lower_linerange"). Default value is "pointrange" or "errorbar". Used only when add != "none" and add contains one "mean_*" or "med_*" where "*" = sd, se, ...

label  the name of the column containing point labels. Can be also a character vector with length = nrow(data).

font.label  a list which can contain the combination of the following elements: the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of labels. For example font.label = list(size = 14, face = "bold", color = "red"). To specify only the size and the style, use font.label = list(size = 14, face = "plain").
label.select can be of two formats:

- a character vector specifying some labels to show.
- a list containing one or the combination of the following components:
  - top.up and top.down: to display the labels of the top up/down points. For example, `label.select = list(top.up = 10, top.down = 4)`.
  - criteria: to filter, for example, by x and y variables values, use this: `label.select = list(criteria = "y > 2 & y < 5 & x %in% c('A', 'B')")`.

repel a logical value, whether to use ggrepel to avoid overplotting text labels or not.

label.rectangle logical value. If TRUE, add rectangle underneath the text, making it easier to read.

position Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

ggtheme function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ....

Value

list. A list contains a character vector with Haps were applied student test, a matrix contains p-value of each compare of Haps and a ggplot2 object named as figs if mergeFigs set as TRUE, or two ggplot2 objects names as fig_pvalue and fig_Violin

Examples

data("geneHapR_test")
# plot the figs directly
hapVsPheno(hap = hapResult,
    pheno = pheno,
    phenoName = "GrainWeight.2021",
    minAcc = 3)

# do not merge the files
results <- hapVsPheno(hap = hapResult,
    pheno = pheno,
    phenoName = "GrainWeight.2021",
    minAcc = 3,
    mergeFigs = FALSE)

plot(results$fig_pvalue)
plot(results$fig_Violin)
Description

Comparie phenotype site by site.

Usage

hapVsPhenoPerSite(
  hap,
  pheno,
  phenoName,
  sitePOS,
  fileName,
  fileType = NULL,
  freq.min = 5,
  ...
)

Arguments

hap an R object of hapresult class
pheno, phenoName pheno, a data.frame contains the phenotypes; Only one phenotype name is required.
sitePOS the coordinate of site
fileName, fileType file name and file type will be needed for saving result, file type could be one of "png, tiff, jpg"
freq.min miner allies frequency less than freq.min will not be skipped
... additional params will be passed to plot saving function like tiff(), png(), pdf()

Examples

data("geneHapR_test")
hapVsPhenoPerSite(hapResult, pheno, sitePOS = "4300")
hapVsPhenos

Description
hapVsPhenos

Usage
hapVsPhenos(
  hap,
  pheno,
  outPutSingleFile = TRUE,
  hapPrefix = "H",
  title = "Seita.0G000000",
  width = 12,
  height = 8,
  res = 300,
  compression = "lzw",
  filename.prefix = filename.prefix,
  filename.suffix = "pdf",
  filename.sep = "_",
  outlier.rm = TRUE,
  mergeFigs = TRUE,
  ...
)

Arguments
hap object of hapResult class, generate with vcf2hap() or seqs2hap() 
pheno object of data.frame class, imported by importPheno() 
outPutSingleFile TRUE or FALSE indicate whether put all figs into to each pages of single file or generate multi-files. Only worked while file type is pdf 
hapPrefix prefix of hapotypes, default as "H" 
title a character which will used for figure title 
width manual option for determining the output file width in inches. (default: 12) 
height manual option for determining the output file height in inches. (default: 8) 
res The nominal resolution in ppi which will be recorded in the bitmap file, if a positive integer. Also used for units other than the default, and to convert points to pixels 
compression the type of compression to be used.
filename.prefix, filename.surfix, filename.sep
if multi files generated, file names will be formed by prefix filename .prefix, a separate character filename . sep, pheno name, a dot and suffix filename.surfix, and file type was decide by filename.surfix; if single file was generated, file name will be formed by prefix filename. prefix, a dot and suffix filename.surfix
outlier.rm whether remove outliers, default as TRUE
mergeFigs bool type, indicate whether merge the heat map and box plot or not. Default as FALSE
... Arguments passed on to hapVsPheno
phenoName pheno name for plot, should be one column name of pheno
minAcc,freq.min If observations number of a Hap less than this number will not be compared with others or be plotted. Should not less than 3 due to the t-test will meaninglessly. Default as 5
angle the angle of x labels
hjust,vjust hjust and vjust of x labels
comparisons a list contains comparison pairs eg. list(c("H001", "H002"), c("H001", "H004")), or a character vector contains haplotype names for comparison, or "none" indicates do not add comparisons.
method a character string indicating which method to be used for comparing means.
method.args a list of additional arguments used for the test method. For example one might use method.args = list(alternative = "greater") for wilcoxon test.
symnum.args a list of arguments to pass to the function symnum for symbolic number coding of p-values. For example, symnum.args <- list(cutpoints = c(0, 0.0001, 0.001, 0.01, 0.05, Inf), symbols = c("****", "***", "**", ", *, "ns").
In other words, we use the following convention for symbols indicating statistical significance:

• ns: p > 0.05
• *: p <= 0.05
• **: p <= 0.01
• ***: p <= 0.001
• ****: p <= 0.0001

Value
No return value

Examples

data("geneHapR_test")
oriDir <- getwd()
temp_dir <- tempdir()
if(! dir.exists(temp_dir))
dir.create(temp_dir)
setwd(temp_dir)
# analysis all pheno in the data.frame of pheno
hapVsPhenos(hapResult, pheno, outputSingleFile = TRUE, hapPrefix = "H", title = "Seita.0G000000", filename.prefix = "test", width = 12, height = 8, res = 300)
setwd(oriDir)

---

**hap_summary**

### Summary Hap Results

**Description**

A function used for summarize hapResult to visualization and calculation.

**Usage**

```r
hap_summary(hap, hapPrefix = "H", file = file)
```

**Arguments**

- **hap**: object of hapResult class, generated by `vcf2hap()` or `seqs2hap` or `import_hap()`
- **hapPrefix**: prefix of hap names, default as "H"
- **file**: file path where to save the hap summary result. If missing, nothing will be saved to disk.

**Details**

It is suggested to use the result of `vcf2hap()` or `seqs2hap()` as input directly. However the user can import previously hap result from local file with `import_hap()`

**Value**

hapSummary, first four rows are fixed to meta information: CHR, POS, INFO, ALLELE Hap names were placed in first column, Accessions and freqs were placed at the last two columns.

**Note**

If the user have changed the default hapPrefix in `vcf2hap()` or `seqs2hap()`, then the parameter `hapPrefix` is needed. Furthermore, a multi-letter prefix of hap names is possible.
import_AccINFO

Examples

data(“geneHapR_test”)  
  hapSummary <- hap_summary(hapResult, hapPrefix = "H")

import_AccINFO  

Import Accession Information from File

Description

import accession information including phenotype data, accession group, location from a tab delimited table file

Usage

import_AccINFO(file, comment.char = “#”,  
  check.names = FALSE, row.names = 1, …)

Arguments

file  
  file path, this file should be a tab delimited table

comment.char  
  character: a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether.

check.names  
  logical. If TRUE then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names. If necessary they are adjusted (by make.names) so that they are, and also to ensure that there are no duplicates.

row.names  
  a vector of row names. This can be a vector giving the actual row names, or a single number giving the column of the table which contains the row names, or character string giving the name of the table column containing the row names. If there is a header and the first row contains one fewer field than the number of columns, the first column in the input is used for the row names. Otherwise if row.names is missing, the rows are numbered. Using row.names = NULL forces row numbering. Missing or NULL row.names generate row names that are considered to be ‘automatic’ (and not preserved by as.matrix).

...  
  Further arguments to be passed to read.table.

Details

First column should be Accessions; phenos/accession information should begin from second column, phenoName/group/locations should located at the first row. If a dot ‘.’ is located in pheno name, then the part before the dot will be set as y axis name and the latter will be set as foot when plot figures.
Value

data.frame. Accession names were set as rownames and columns were named by pheno/info names.

Examples

```r
oldDir <- getwd()
temp_dir <- tempdir()
if(! dir.exists(temp_dir))
dir.create(temp_dir)
setwd(temp_dir)
data("geneHapR_test")
write.table(pheno, file = "test.pheno.txt", sep = "\t")
pheno <- import_AccINFO("test.pheno.txt")
pheno
setwd(oldDir)
```

---

import_bed

**import annotation files in BED format**

---

Description

import bed files contains annotations into R as GRanges object.

Usage

```r
import_bed(con, quite = FALSE)
```

Arguments

- `con`: A path, URL, connection or BEDFile object. For the functions ending in `.bed`, `.bedGraph` and `.bed15`, the file format is indicated by the function name. For the base export and import functions, the format must be indicated another way. If `con` is a path, URL or connection, either the file extension or the format argument needs to be one of “bed”, “bed15”, “bedGraph”, “bedpe”, “narrow-Peak”, or “broadPeak”. Compressed files (“gz”, “bz2” and “xz”) are handled transparently.

- `quite`: whether show message

Details

If there is no genome annotation file in GFF format for your interest species, a BED file is convenient to custom a simple annotation file for a gene. Here we suggest two type of BED format: BED6 and BED4.

As the definition of UCSC. The BED6 contains 6 columns, which are 1) chrom, 2) chromStart, 3) chromEnd, 4) name, 5) score and 6) strand. The BED4 format contains the first 4 column of BED6 format.
However, in gene haplotype statistics, we only care about the type of each site. Thus we use the fourth column to define the transcripts name and "CDS" or "UTRs", separated by a space, e.g.:

Chr8 678 890 HD1.1 CDS . -
Chr8 891 989 HD1.1 five_prime_UTR . -
Chr8 668 759 HD1.2 CDS . -
Chr8 908 989 HD1.2 CDS . -

This example indicates a small gene named as HD1 have two transcripts, named as HD1.1 and HD1.2, separately. HD1 has a CDS and a UTR region; while HD1.2 has two CDS regions.

Value

GRange object

Examples

```r
bed.Path <- system.file("extdata", "annotation.bed6", package = "geneHapR")
bed <- import_bed(bed.Path)
bed
```

---

**import_gff**  
**Import Annotations from GFF Format File**

Description

import genome annotations in GFF/GFF3 format

Usage

```r
import_gff(gffFile, format = "gff")
```

Arguments

- `gffFile` - the gff file path
- `format` - should be one of "gff", "gff1", "gff2", "gff3", "gvf", or "gtf". Default as gff

Value

GRange object

Examples

```r
gff.Path <- system.file("extdata", "annotation.gff", package = "geneHapR")
gff <- import_gff(gff.Path, format = "gff")
gff
```
import_hap  

Import hapResult/hapSummary

Description

This function could be used for import hap result or hap summary result. The type of returned object is decided by input file, see details.

Usage

import_hap(file, ...)

Arguments

- file: hapSummary or hapResult file path
- ...: extras will pass to read.delim()

Details

The hap result and hap summary result have common features. The common features of these two types are: First four rows contains extra information: CHR, POS, INFO and ALLELE Hap names were in the first column. The differences are: Hap summary result have a freq column while hap result not. Rows represent haplotypes in hap summary result, while rows represent accessions in hap result. In addition, the accessions of each haplotype in hap summary result were separated by ";".

Value

hapSummary or hapResult

Examples

```r
oldDir <- getwd()
temp_dir <- tempdir()
if(! dir.exists(temp_dir))
  dir.create(temp_dir)
setwd(temp_dir)
data("geneHapR_test")
write.hap(hapResult, file = "test.pheno.txt", sep = "\t")
hap <- import_hap("test.pheno.txt")
setwd(oldDir)
```
import_MultipleAlignment

Import MultipleAlignment Result

Description

import sequences aligned results

Usage

import_MultipleAlignment(filepath, format = "fasta", type = "DNA")

Arguments

filepath
A character vector (of arbitrary length when reading, of length 1 when writing) containing the paths to the files to read or write. Note that special values like "" or "\cmd" (typically supported by other I/O functions in R) are not supported here. Also filepath cannot be a connection.

format
Either "fasta" (the default), stockholm, or "clustal".

type
one of "DNA" and "Protein"

Value

object of DNAMultipleAlignment

Examples

aliSeqPath <- system.file("extdata", "seqs.fa", package = "geneHapR")

geneSeqs <- import_MultipleAlignment(filepath = aliSeqPath,
  format = "fasta",
  type = "DNA")

geneSeqs <- import_MultipleAlignment(filepath = aliSeqPath,
  format = "fasta",
  type = "Protein")

import_plink.pedmap

Description

used for import regular p.link file stored in map and ped format
import_seqs

Usage

import_plink.pedmap(root = root, 
    sep_ped = \"\t\", sep_map = \"\t\", 
    pedfile = pedfile, mapfile = mapfile)

Arguments

root The file name without suffix. This function only support plink file format stored in \"map\" and \"ped\" format, the file names after removed suffix should be same with each other.

sep_ped a character indicate the separation of ped file

sep_map a character indicate the separation of map file

pedfile, mapfile if root is missing then pedfile and mapfile are needed

Value

list, contains map information stored in data.frame and ped information stored in data.frame

Examples

pedfile <- system.file("extdata", 
    "snp3kvars-CHR8-25947258-25951166-plink.ped", 
    package = "geneHapR")
mapfile <- system.file("extdata", 
    "snp3kvars-CHR8-25947258-25951166-plink.map", 
    package = "geneHapR")
p.link <- import_plink.pedmap(pedfile = pedfile, mapfile = mapfile, 
    sep_map = \"\t\", sep_ped = \"\t\")
p.link <- filter_plink.pedmap(p.link, mode = "POS", 
    Chr = \"chr08\", start = 25948004, 
    end = 25949944)
hapResult <- plink.pedmap2hap(p.link, hapPrefix = \"H\", 
    hetero_remove = TRUE, 
    na_drop = TRUE)

import_seqs

Import Sequences

Description

import DNA sequences in FASTA format

Usage

import_segs(filepath, format = \"fasta\")
import_vcf

Arguments

filepath A character vector containing the path to the DNA sequences file. Reading files in gzip format (which usually have the '.gz' extension) is supported. Note that only DNA supported here.

format Either "fasta" (the default) or "fastq"

Value

object of DNAStringSet class

Examples

seqPath <- system.file("extdata", "seqs.fa", package = "geneHapR")
geneSeqs <- import_seqs(filepath = seqPath, format = "fasta")

import_vcf

Import VCF from File

Description

import *.vcf structured text format, as well as the compressed *.vcf.gz format.

Usage

import_vcf(file = file, ...)

import_vcf(file = file, ...)

Arguments

file file path of VCF file

... pass to vcfR::read.vcfR()

Value

vcfR object

Author(s)

Zhangrenl

See Also

vcfR::read.vcfR()
LDheatmap

Examples

vcfPath <- system.file("extdata", "var.vcf.gz", package = "geneHapR")
vcf <- import_vcf(file = vcfPath)
vcf

LDheatmap

This function produces a pairwise LD plot.

Description

LDheatmap() is used to produce a graphical display, as a heat map, of pairwise linkage disequilibrium (LD) measurements for SNPs. The heat map is a false color image in the upper-left diagonal of a square plot. Optionally, a line parallel to the diagonal of the image indicating the physical or genetic map positions of the SNPs may be added, along with text reporting the total length of the genomic region considered.

Usage

plot_LDheatmap(
  hap,
  gff,
  Chr,
  start,
  end,
  geneID = NULL,
  distances = "physical",
  LDmeasure = "r",
  title = "Pairwise LD",
  add.map = TRUE,
  map.height = 1,
  colorLegend = TRUE,
  geneMapLocation = 0.15,
  geneMapLabelX = NULL,
  geneMapLabelY = NULL,
  SNP.name = TRUE,
  color = NULL,
  color_gmodel = "grey",
  color_snp = "grey",
  color_snpname = "grey40",
  cex_snpname = 0.8,
  snpmarks_height = NULL,
  newpage = TRUE,
  name = "ldheatmap",
  vp.name = NULL,
  pop = FALSE,


LDheatmap

```r
text = FALSE
```

Arguments

- **hap**: R object of hapSummary or hapResult class.
- **gff**: annotations
  - Chr, start, end, geneID chromosome, start and end position, gene ID for extract annotation in target range.
- **distances**: A character string to specify whether the provided map locations are in physical or genetic distances. If `distances = "physical"` (default), the text describing the total length of the region will be “Physical Length:XXkb” where XX is the length of the region in kilobases. If `distances = "genetic"`, the text will be “Genetic Map Length:YYcM” where YY is the length of the region in centi-Morgans. If `gdat` is an object of class LDheatmap, distances is taken from gdat.
- **LDmeasure**: A character string specifying the measure of LD
  - either allelic correlation $r^2$ or Lewontin’s $|D'|$; default = “r” for $r^2$; type “$D/'$ for $|D'|$. This argument is ignored when the user has already supplied calculated LD measurements through gdat (i.e., when gdat is a matrix of pairwise LD measurements or an object of class "LDheatmap").
- **title**: A character string for the main title of the plot. Default is “Pairwise LD”.
- **add.map**: If TRUE (default) a diagonal line indicating the physical or genetic map positions of the SNPs will be added to the plot, along with text indicating the total length of the genetic region.
- **map.height**: the height of gene map, default is 0.02
- **colorLegend**: If TRUE (default) the color legend is drawn.
- **geneMapLabelX**: A numeric value specifying the position of the line parallel to the diagonal of the matrix; the larger the value, the farther it lies from the matrix diagonal. Ignored when add.map = FALSE.
- **geneMapLabelY**: A numeric value specifying the y-coordinate of the text indicating the total length of the genomic region being considered. Ignored when add.map = FALSE.
- **SNP.name**: a logical vector indicated whether display SNP names using positions.
- **color**: A range of colors to be used for drawing the heat map. Default is `grDevices::colorRampPalette(c("red", "grey"))(30)`.
- **color_gmodel, color_snp, color_snpname**: the color of gene model and snp and snp names respectively, default as grey80.
- **cex_snpname**: the size of snp names/labels
- **snpmarks_height**: the height of snp marks, if set as NULL, nothing will display on gene model where the heat map is going to be drawn.
newpage If TRUE (default), the heat map will be drawn on a new page.

name A character string specifying the name of the LDheatmap graphical object (grob) to be produced.

vp.name A character string specifying the name of the viewport

pop If TRUE, the viewport where the heat map is drawn is popped (i.e. removed) from the viewport tree after drawing. Default = FALSE.

text If TRUE, the LD measurements are printed on each cell.

Details

The input object gdat can be a data frame of genotype objects (a data structure from the genetics package), a SnpMatrix object (a data structure from the snpStats package), or any square matrix with values between 0 and 1 inclusive. LD computation is much faster for SnpMatrix objects than for genotype objects. In the case of a matrix of LD values between 0 and 1, the values above the diagonal will be plotted. In the display of LD, SNPs appear in the order supplied by the user as the horizontal and vertical coordinates are increased and one moves along the off-diagonal line, from the bottom-left to the top-right corner. To achieve this, the conventions of the image() function have been adopted, in which horizontal coordinates correspond to the rows of the matrix and vertical coordinates correspond to columns, and vertical coordinates are indexed in increasing order from bottom to top. See the package vignette LDheatmap for more examples and details of the implementation. Examples of adding “tracks” of genomic annotation above a flipped heatmap are in the package vignette addTracks.

Value

An object of class "LDheatmap" which contains the following components:

LDmatrix The matrix of pairwise LD measurements plotted in the heat map.

LDheatmapGrob A grid graphical object (grob) representing the produced heat map.

heatmapVP The viewport in which the heat map is drawn. See viewport.

genetic.distances The vector of the supplied physical or genetic map locations, or the vector of equispaced marker distances when no distance vector is supplied.

distances A character string specifying whether the provided map distances are physical or genetic.

color The range of colors used for drawing the heat map.

The grob LDheatmapGrob has three grobs as its children (components). They are listed below along with their own children and respectively represent the color image with main title, genetic map and color key of the heat map: "heatMap" - "heatmap", "title": "genMap" - "diagonal", "segments", "title", "symbols", "SNPnames"; and "Key" - "colorKey", "title", "labels", "ticks", "box".

Note

The produced heat map can be modified in two ways. First, it is possible to edit interactively the grob components of the heat map, by using the function grid.edit; the function will not work
if there is no open graphical device showing the heat map. Alternatively, the user can use the function `editGrob` and work with the grob `LDheatmapGrob` returned by `LDheatmap`. See Examples for usage. `LDheatmap()` uses `Grid`, which does not respond to `par()` settings. Hence modifying `par()` settings of `mfrow` and `mfcol` will not work with `LDheatmap()`. The Examples section shows how to display multiple heat maps on one plot without the use of `par()`.

References


Examples

```r
# Pass LDheatmap a SnpMatrix object
data(geneHapR_test)
plot_LDheatmap(hap = hapResult,
gff = gff,
Chr = hapResult[1,2],
start = 4000, end = 8200)
```

Description

computes a haplotype network with haplotype summary result

Usage

```r
get_hapNet(hapSummary,
AccINFO = AccINFO,
groupName = groupName,
na.label = "Unknown")
```

```r
getHapGroup(
hapSummary,
AccINFO = AccINFO,
groupName = groupName,
na.label = na.label
)
```

Arguments

- `hapSummary` object of `hapSummary` class, generated by `hap_summary()`
- `AccINFO` data.frame, specified groups of each accession. Used for pie plot. If missing, pie will not draw in `plotHapNet`. Or you can supplied a `hap_group` matrix with `plot(hapNet, pie = hap_group)`.
groupName      the group name used for pie plot, should be in AccINFO column names, default as the first column name
na.label        the label of NAs

Value

hapNet class

References

Mark P.J. van der Loo (2014) doi:10.32614/RJ2014011;

See Also

plotHapNet() and hap_summary().

Examples

data("geneHapR_test")
hapSummary <- hap_summary(hapResult)

# calculate haploNet
hapNet <- get_hapNet(hapSummary,
                      AccINFO = AccINFO, # accession types
groupName = colnames(AccINFO)[2])

# plot haploNet
plot(hapNet)

# plot haploNet
plotHapNet(hapNet,
            size = "freq", # circle size
            scale = "log10", # scale circle with 'log10(size + 1)'
cex = 1, # size of hap symbol
            col.link = 2, # link colors
            link.width = 2, # link widths
            show.mutation = 2, # mutation types one of c(0,1,2,3)
            legend = FALSE) # legend position

plink.pedmap2hap

Description

convert p.link format data into hapResult
Usage

plink.pedmap2hap(
    p.link,
    hapPrefix = "H",
    pad = 3,
    hetero_remove = TRUE,
    na_drop = TRUE
)

Arguments

  p.link list contains p.link information
  hapPrefix prefix of haplotype names
  pad The number length in haplotype names should be extend to.
  hetero_remove whether remove accessions contains hyb-sites
  na_drop whether drop accessions contains missing data ("N", NA)

Value

  object of hapSummary class

Examples

pedfile <- system.file("extdata",
    "snp3kvars-CHR8-25947258-25951166-plink.ped",
    package = "geneHapR")
mapfile <- system.file("extdata",
    "snp3kvars-CHR8-25947258-25951166-plink.map",
    package = "geneHapR")
p.link <- import_plink.pedmap(pedfile = pedfile, mapfile = mapfile,
    sep_map = "\t", sep_ped = "\t")
p.link <- filter_plink.pedmap(p.link, mode = "POS",
    Chr = "chr08", start = 25948004,
    end = 25949944)
hapResult <- plink.pedmap2hap(p.link, hapPrefix = "H",
    hetero_remove = TRUE,
    na_drop = TRUE)
Usage

plotEFF(
  siteEFF,
  gff = gff,
  Chr = Chr,
  start = start,
  end = end,
  showType = c("five_prime_UTR", "CDS", "three_prime_UTR"),
  CDS.height = CDS.height,
  cex = 0.1,
  col = c("red", "yellow"),
  pch = 20,
  main = main,
  legend.cex = 0.8,
  gene.legend = TRUE,
  markMutants = TRUE,
  mutants.col = 1,
  mutants.type = 1,
  y = c("pvalue", "effect"),
  ylab = ylab,
  legendtitle = legendtitle,
  par.restore = TRUE
)

Arguments

siteEFF matrix, column name are pheno names and row name are site position
gff gff annotation
Chr the chromosome name
start start position
end end position
showType character vector, eg.: "CDS", "five_prime_UTR", "three_prime_UTR"
CDS.height numeric indicate the height of CDS in gene model, range: [0, 1]
cex a numeric control the size of point
col vector specified the color bar
pch vector controls points type, see \texttt{par()}
main main title
legend.cex a numeric control the legend size
gene.legend whether add legend for gene model
markMutants whether mark mutants on gene model, default as \texttt{TRUE}
mutants.col color of lines which mark mutants
mutants.type a vector of line types
y, ylab, legendtitle

- y: indicate either pvalue or effect should be used as y axix, ylab, legendtitle: character, if missing, the value will be decide by y.
- par.restore: default as TRUE, wether restore the origin par after ploted EFF.

Value

No return value, called for side effects

Examples

data("geneHapR_test")
# calculate site functional effect
# siteEFF <- siteEFF(hapResult, pheno, names(pheno))
# plotEFF(siteEFF, gff = gff, Chr = "scaffold_1")

plotHapNet

Description

plotHapNet

Usage

plotHapNet(
  hapNet,
  size = "freq",
  scale = 1,
  cex = 0.8,
  cex.legend = 0.6,
  col.link = 1,
  link.width = 1,
  show.mutation = 2,
  backGround = backGround,
  hapGroup = hapGroup,
  legend = FALSE,
  show.size_legend = TRUE,
  show_color_legend = TRUE,
  pie.lim = c(0.5, 2),
  main = main,
  labels = TRUE,
  legend_version = 0,
  labels.cex = 0.8,
  labels.col = "blue",
)
plotHapNet

labels.adj = NULL,
labels.font = 2,
...
)

Arguments

hapNet an object of class "haploNet"
size a numeric vector giving the diameter of the circles representing the haplotypes: this is in the same unit than the links and eventually recycled.
scale a numeric indicate the ratio of the scale of the links representing the number of steps on the scale of the circles representing the haplotypes or a character one of c("log10", "log2") indicate the scale method by log10(size) or log2(size), respectively. Default as 1
cex character expansion factor relative to current par("cex")
cex.legend same as cex, but for text in legend
col.link a character vector specifying the colours of the links; eventually recycled.
link.width a numeric vector giving the width of the links; eventually recycled.
show.mutation an integer value:
if 0, nothing is drawn on the links;
if 1, the mutations are shown with small segments on the links;
if 2, they are shown with small dots;
if 3, the number of mutations are printed on the links.
backGround a color vector with length equal to number of Accession types
hapGroup a matrix used to draw pie charts for each haplotype; its number of rows must be equal to the number of haplotypes
legend a logical specifying whether to draw the legend, or a vector of length two giving the coordinates where to draw the legend; FALSE by default. If TRUE, the user is asked to click where to draw the legend.
show_size_legend, show_color_legend whether show size or color legend
pie.lim A numeric vector define the maximum and minmum pie size, which will be avoid the pie to samll or too large
main The main title (on top) using font, size (character expansion) and color par(c("font.main", "cex.main", "col.main"))
labels a logical specifying whether to identify the haplotypes with their labels (default as TRUE)
legend_version the size legened style, default as 0
labels.cex the size of labels
labels.col the labels color
labels.adj a named list contains two length vectors defining the adjustment of labels. The names should be exactly matched with the haplootype names. default as NULL.
labels.font the font of labels, default as 2
... other parameters will pass to plot function
plotHapTable

Details
Additional parameters control the network features: `labels.cex = 1`, `labels.font = 2`, `link.color = "black"`, `link.type = 1`, `link.type.alt = 2`, `link.width = 1`, `link.width.alt = 1`, `haplotype.inner.color = "white"`, `haplotype.outer.color = "black"`, `mutations.cex = 1`, `mutations.font = 1`, `mutations.frame.background = "#0000FF4D"`, `mutations.frame.border = "black"`, `mutations.text.color = 1`, `mutations.arrow.color = "black"`, `mutations.arrow.type = "triangle"`, `mutations.sequence.color = "#BFBFBF4D"`, `mutations.sequence.end = "round"`, `mutations.sequence.length = 0.3`, `mutations.sequence.width = 5`, `pie.inner.segments.color = "black"`, `pie.colors.function = rainbow`, `scale.ratio = 1`, `show.mutation = 2`

Value
No return value

See Also
`hap_summary()` and `get_hapNet()`.

Examples

```r
data("geneHapR_test")
hapSummary <- hap_summary(hapResult)

# calculate haploNet
hapNet <- get_hapNet(hapSummary,
    AccINFO = AccINFO, # accession types
    groupName = colnames(AccINFO)[2])

# plot haploNet
plot(hapNet)

# plot haploNet
plotHapNet(hapNet,
    size = "freq", # circle size
    scale = "log10", # scale circle with 'log10(size + 1)'
    cex = 1, # size of hap symbol
    col.link = 2, # link colors
    link.width = 2, # link widths
    show.mutation = 2, # mutation types one of c(0,1,2,3)
    legend = FALSE) # legend position
```

plotHapTable

Description
display hap result as a table-like figure
plotHapTable(hapSummary,  
    hapPrefix = "H",  
    title = "",  
    geneName = geneName,  
    INFO_tag = NULL,  
    tag_split = tag_split,  
    tag_field = tag_field,  
    tag_name = tag_name,  
    displayIndelSize = 0, angle = c(0,45,90),  
    replaceMultiAllele = TRUE,  
    ALLELE.color = "grey90")

Arguments

hapSummary object of hapSummary class
hapPrefix prefix of haplotype names. Default as "H"
title the main title of the final figure
geneName character, will be used for filter INFO filed of ANN
INFO_tag The annotations in the INFO field are represented as tag-value pairs, where the tag and value are separated by an equal sign, ie "]="", and pairs are separated by colons, ie "];". For more information please see details.
tag_split usually, the value of tag-value contains one information. However, if a tag contains more than one fields, eg "ANN", then tag_split is needed. When INFO_tag was set as "ANN" or "SNPEFF", tag_split will be set as "]|" by default, see details.
tag_field integer, if a tag-value contains more than one fields, user need to specified which field should be display. If tag_field set as 0, the whole contents will be displayed. Default as 0.
tag_name tag name is displayed in Hap figure. If tag_name is missing, will take the value of INFO_tag.
displayIndelSize display indels with max size of displayIndelSize, If set as 0, all indels will convert into "]*" of which "]i" represents "]indel".
angle the angle of coordinates, should be one of 0, 45 and 90
replaceMultiAllele whether to replace MultiAllele with "T*", default as TRUE.
ALLELE.color the color of ALLELE row, default as "grey90"

Details

In VCF files, the INFO field are represented as tag-value pairs, where the tag and value are separated by an equal sign, ie "]="", and pairs are separated by colons, ie "];".

If hapSummary were generated from sequences, INFO row is null. If hapSummary were generated from VCF, INFO was take from the INFO column in the source VCF file. Some tag-values
Values may contain more than one value separated by "|", eg.: "ANN" or "snpeff" added by 'snpeff' or other software. For those fields we need specified value of `tag_field = "ANN"` and `tag_split = "[|]"`, it's suggest specified the value of `tag_name` for display in figure.

'snpeff', a toolbox for genetic variant annotation and functional effect prediction, will add annotations to INFO filed in VCF file under a tag named as "ANN". The annotations contains several fields separated by "|". eg.:

1. Allele
2. Annotation
3. Annotation_Impact
4. Gene_Name
5. Gene_ID
6. Feature_Type
7. Feature_ID
8. Transcript_BioType
9. Rank
10. HGVS.c
11. HGVS.p
12. cDNA.pos/cDNA.length ...

However, the INFO in hapResults may missing annotations that we need. In this case, we can custom INFOs in hapSummarys with `addINFO()`. Once the needed annotations were included in hap results, we can display them with `plotHapTable()` by specify the value of `INFO_tag`.

**Value**

`ggplot2` object

**See Also**

`addINFO()`

**Examples**

data("geneHapR_test")
plotHapTable(hapResult)
seqs2hap  Generate Hap Results from Seqs

Description

generate hapResults from aligned and trimed sequences

Usage

```r
seqs2hap(
  seqs,
  Ref = names(seqs)[1],
  hetero_remove = TRUE,
  na_drop = TRUE,
  maxGapsPerSeq = 0.25,
  hapPrefix = "H",
  pad = 3,
  ...
)
```

```r
trimSeqs(seqs,
  minFlankFraction = 0.1)
```

Arguments

- `seqs`  object of DNAStringSet or DNAMultipleAlignment class
- `Ref`  the name of reference sequences. Default as the name of the first sequence
- `hetero_remove`  whether remove accessions contains hybrid site or not. Default as TRUE
- `na_drop`  whether drop sequeces contain "N" Default as TRUE.
- `maxGapsPerSeq`  value in [0, 1] that indicates the maximum fraction of gaps allowed in each seq after alignment (default as 0.25). Seqs with gap percent exceed that will be dropped
- `hapPrefix`  prefix of hap names. Default as "H"
- `pad`  The number length in haplotype names should be extend to.
- `...`  Parameters not used.
- `minFlankFraction`  A value in [0, 1] that indicates the minimum fraction needed to call a gap in the consensus string (default as 0.1).

Value

object of hapResult class
Examples

data("geneHapR_test")
seqs
seqs <- trimSeqs(seqs,
    minFlankFraction = 0.1)
seqs
hapResult <- seqs2hap(seqs,
    Ref = names(seqs)[1],
    hetero_remove = TRUE, na_drop = TRUE,
    maxGapsPerSeq = 0.25,
    hapPrefix = "H")

Description

Set position of ATG as zero in hap result and gff annotation. The upstream was negative while the gene range and downstream was positive.

Usage

gffSetATGas0(gff = gff, hap = hap,
    geneID = geneID,
    Chr = Chr, POS = POS)

hapSetATGas0(gff = gff, hap = hap,
    geneID = geneID,
    Chr = Chr, POS = POS)

Arguments

gff gene annotations
hap object of hapResult or hapSummary class
geneID geneID
Chr Chromosome name
POS vector consist with start and end position

Details

Filter hap result and gff annotation according to provided information. And then set position of ATG as zero in hap result and gff annotation. The upstream was negative while the gene range and downstream was positive.

Notice: the position of "ATG" after modified was 0, 1 and 2 separately. The site in hap result exceed the selected range will be dropped.
Value

gffSetATGas0: filtered gff with position of ATG was as zero
hapSetATGas0: hap results with position of ATG was set as zero

See Also

displayVarOnGeneModel()

Examples

# load example dataset
data("geneHapR_test")

# set position of ATG as zero in gff
ewgff <- gffSetATGas0(gff = gff, hap = hapResult,
geneID = "test1G0387",
Chr = "scaffold_1",
POS = c(4300, 7910))
data("geneHapR_test")

# set position of ATG as zero in hap results
newhapResult <- hapSetATGas0(gff = gff, hap = hapResult,
geneID = "test1G0387",
Chr = "scaffold_1",
POS = c(4300, 7910))

---

siteEFF

Calculation of Sites Effective

Description

Calculation of Sites Effective

Usage

siteEFF(hap, pheno, phenoNames, quality = FALSE, method = "auto",
p.adj = "none")

Arguments

hap object of "hapResult" class
pheno phenotype data, with column names as pheno name and row name as accessions.
phenoNames pheno names used for analysis, if missing, will use all pheno names in pheno
quality: bool type, indicate whether the type of phenos are quality or quantitative. Length of quality could be 1 or equal with length of phenoNames. Default as FALSE

method: character or character vector with length equal with phenoNames indicate which method should be performed towards each phenotype. Should be one of "t.test", "chi.test", "wilcox.test" and "auto". Default as "auto", see details.

p.adj: character, indicate correction method. Could be "BH", "BY", "none"

Details

The site EFF was determined by the phenotype difference between each site genotype.

The p was calculated with statistical analysis method as designated by the parameter method. If method set as "auto", then chi.test will be selected for quantity phenotype, eg.: color; for quantity phenotype, eg.: height, with at least 30 observations per genotype and fit Gaussian distribution t.test will be performed, otherwise wilcox.test will be performed.

Value

a list containing two matrix names as "p" and "EFF", with column name are pheno names and row name are site position. The matrix names as "p" contains all p-value. The matrix named as "EFF" contains scaled difference between each genotype per site.

Examples

data("geneHapR_test")

# calculate site functional effect
# siteEFF <- siteEFF(hapResult, pheno, names(pheno))
# plotEFF(siteEFF, gff = gff, Chr = "scaffold_1")

description: convert variants stored in table format into hapResult

usage: table2hap(x, hapPrefix = "H", pad = 3, hetero_remove = TRUE, na_drop = TRUE)
Arguments

x  a data.frame contains variants information. The first file column are fix as Chrome name, position, reference nucleotide, alter nucleotide and INFO. Accession genotype should be in followed columns. "-" will be treated as Indel. "." and "N" will be treated as missing data. Heterozygotes should be "A/T", "AAA/A"

hapPrefix  prefix of haplotype names

pad  The number length in haplotype names should be extend to.

hetero_remove  whether remove accessions contains hyb-sites, Character not A T C G

na_drop  whether drop accessions contains missing data ("N", "NA", ".")

Value

object of hapSummary class

Examples

data("geneHapR_test")
hapResult <- table2hap(gt.geno, hapPrefix = "H",
                         hetero_remove = TRUE,
                         na_drop = TRUE)

vcf2hap  

Generat Haps from VCF

Description

Generate hapResult from vcfR object A simple filter by position was provided in this function, however it's prefer to filter VCF (vcfR object) through `filter_vcf()`.

Usage

vcf2hap(
    vcf,
    hapPrefix = "H",
    filter_Chr = FALSE,
    filter_POS = FALSE,
    pad = 3,
    hetero_remove = TRUE,
    na_drop = TRUE,
    ...
)
write.hap

Save Haplotype Results to Disk

Description

This function will write hap result into a txt file.

Usage

write.hap(x, file = file, sep = "\t")

Arguments

- **x**: objec of hapResult or hapSummary class
- **file**: file path, where to save the hap result/summary
- **sep**: the field separator string. Values within each row of x are separated by this string. Default as "\t"

Arguments

- **vcf**: vcfR object imported by import_vcf()
- **hapPrefix**: prefix of hap names, default as "H"
- **filter_Chr**: not used
- **filter_POS**: not used
- **pad**: The number length in haplotype names should be extend to.
- **hetero_remove**: whether remove accessions contains hybrid site or not. Default as TRUE
- **na_drop**: whether remove accessions contains unknown allele site or not Default as TRUE.
- **...**: Parameters not used

Value

object of hapResult class

Author(s)

Zhangrenl

See Also

eXtract genotype from vcf: vcfR::extract_gt_tidy(), import vcf files: import_vcf() (preferred) and vcfR::read.vcfR(), filter vcf according position and annotations: filter_vcf()
Details

The hap result and hap summary result have common features. The common features of these two types are: First four rows contain extra information: CHR, POS, INFO and ALLELE Hap names were in the first column. The differences are: Hap summary result have a freq column while hap result not. Rows represent haplotypes in hap summary result, while rows represent accessions in hap result. In addition, the accessions of each haplotype in hap summary result were separated by ";".

Value

No return value

Examples

```r
oriDir <- getwd()
temp_dir <- tempdir()
if(! dir.exists(temp_dir))
dir.create(temp_dir)
setwd(temp_dir)
data("geneHapR_test")
write.hap(hapResult, file = "hapResult.txt")
setwd(oriDir)
```
Index

* datasets
  DataSet, 5
  AccINFO (DataSet), 5
  addINFO, 3
  addINFO(), 45
  addPromoter, 4
  as.haplotype (ashaplotype), 4
  as.matrix, 27
  ashaplotype, 4
  county, 18
  DataSet, 5
  displayVarOnGeneModel, 5
  displayVarOnGeneModel(), 48
  editGrob, 37
  filter_hap, 9
  filter_hmp, 10
  filter_plink.pedmap, 11
  filter_table, 12
  filter_vcf, 13
  filter_vcf(), 50, 51
  filterLargeP, link (filterLargep, link), 6
  filterLargep, link, 6
  filterLargeVCF, 8
  get_hapNet (network), 37
  get_hapNet(), 43
  getGenePOS, 14
  getGeneRanges, 15
  getHapGroup (network), 37
  gff (DataSet), 5
  gffSetATGas0 (SetATGas0), 47
  ggrepur::ggviolin, 20
  Grid, 37
  grid.edit, 36
  gt.geno (DataSet), 5
  hap2hmp, 16
  hap_summary, 26
  hap_summary(), 38, 43
  hapDistribution, 17
  hapResult (DataSet), 5
  hapSetATGas0 (SetATGas0), 47
  hapVsPheno, 19, 25
  hapVsPhenoPerSite, 23
  hapVsPhenos, 24
  hmp2hap (hap2hmp), 16
  import_AccINFO, 27
  import_bed, 28
  import_gff, 29
  import_hap, 30
  import_MultipleAlignment, 31
  import_plink.pedmap, 31
  import_seqs, 32
  import_vcf, 33
  import_vcf(), 51
  LDheatmap, 34
  make.names, 27
  NA_character_, 3
  network, 37
  par(), 40
  pheno (DataSet), 5
  plink.pedmap2hap, 38
  plot_LDheatmap (LDheatmap), 34
  plotEFF, 39
  plotHapNet, 41
  plotHapNet(), 38
  plotHapTable, 43
  plotHapTable(), 3
  seqs (DataSet), 5
  seqs2hap, 46
  SetATGas0, 47

53
siteEFF, 48
sites (addINFO), 3
state, 18
symnum, 20, 25

table2hap, 16, 49
trimSeqs (seqs2hap), 46

usa, 18

vcf (DataSet), 5
vcf2hap, 50
vcfr::extract_gt_tidy(), 51
vcfr::read.vcfR(), 33, 51
viewport, 36

world, 17
write.hap, 51