Package 'geneHapR'

August 22, 2022

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 1.0.8

RoxygenNote 7.2.0

LazyData True

Imports ape, Biostrings, ggpubr, GenomicRanges, magrittr, maps, methods, muscle, IRanges, pegas, reshape2, rlang, rtracklayer, trackViewer, stats, stringdist, stringr, tibble, tidyr, utils, vcfR

Depends ggplot2, R (>= 4.0.0)

Suggests mapdata, maptools, knitr, rmarkdown

License GPL-3

VignetteBuilder knitr

NeedsCompilation no

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

Repository CRAN

Date/Publication 2022-08-22 14:30:08 UTC

R topics documented:

`addINFO` *Add Infomation 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)
```

---

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 addtional 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(gff,
                      hapSummary,
                      Chr,
                      startPOS, endPOS,
                      type = "pin", cex = 0.7,
                      CDS_h = 0.05, fiveUTR_h = 0.02, threeUTR_h = 0.01)
```

Arguments

`gff`	gff
`hapSummary`	object of hapSummary class
`Chr`	the chromosome name. If missing, the first element in the hapSummary will be used
`startPOS`	If missing, will use the min position in hapSummary
`endPOS`	If missing, will use the max position in hapSummary
`type`	character. Could be "circle", "pie", "pin", "pie.stack" or "flag"
`cex`	a numeric control the size of circle
`CDS_h, fiveUTR_h, threeUTR_h`	The height of CDS 5'UTR and 3'UTR in gene model

Value

No return value

Examples

```
data("geneHapR_test")
hapSummary <- hap_summary(hapResult)
displayVarOnGeneModel(gff, hapSummary,
                      startPOS = 4100,
                      endPOS = 8210,
                      cex = 0.75)
```

`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 orignal 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'.

Value

No return value

Examples

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

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

oldDir <- getwd()
setwd(tempdir())
# extract a single gene/range from large vcf
filterLargeVCF(VCFin = vcfPath, VCFout = "filtered.vcf.gz",
               Chr = "scaffold_1", POS = c(4300,500), 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_vcf_by_gff *Filter VCF by GFF*

Description

filter VCF by GFF annotation or by position or both

Usage

```
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"))
```

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 `custom_type` is needed.
cusTyp	character vector, custom filter type, needed if `type` set to "custom"

Value

vcfR

Examples

```
# filtet hap
data("geneHapR_test")
vcf_f1 <- filter_vcf(vcf, mode = "POS",
                     Chr = "scaffold_1",
                     start = 4300, end = 5890)

vcf_f2 <- filter_vcf(vcf, mode = "type",
                     gff = gff,
                     type = "CDS")
vcf_f3 <- filter_vcf(vcf, mode = "both",
                     Chr = "scaffold_1",
                     start = 4300, end = 5890,
                     gff = gff,
                     type = "CDS")
```

getGenePOS *Get Gene Position*

Description

Get Gene Position

Usage

```
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, defalt is Parent

Value

named vectors contains start, end and strand

Examples

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

getGeneRanges *Get Gene Ranges*

Description

Get Gene Ranges

Usage

```
getGeneRanges(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, defalt is Parent

Value

GRanges

Examples

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

---

hapDistribution *Display of Geography Distribution*

Description

show distribution of intereted haplotypes on maps

Usage

```
hapDistribution(hap, AccINFO, LON.col, LAT.col, hapNames,
                database = "world", regions = ".",
                zColours = zColours,
                legend = TRUE, symbolSize = 1,
                ratio = 1, cex.legend = 0.8,
                lwd.pie = 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 (usa, state, county), and more (type help(package='maps') to see the package index). If the requied 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.
zColours	colours to apply to the pie section for each attribute column
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
ratio	the ratio of Y to N in the output map, set to 1 as default
cex.legend	character expansion factor for legend relative to current par("cex")
lwd.pie	line width of the pies
...	Extra arguments passed to polygon or lines. Of particular interest may be the options border andlty that control the color and line type of the polygon borders when fill = TRUE.

Value

No return value

Examples

```
data("geneHapR_test")
hapDistribution(hapResult,
```

```
                AccINFO = AccINFO,
                LON.col = "longitude",
                LAT.col = "latitude",
                hapNames = c("H001", "H002", "H003"))
```

---

hapVsPheno *hapVsPheno*

Description

hapVsPheno

Usage

```
hapVsPheno(hap,
           pheno,
           phenoName, hapPrefix = "H",
           title = "test1G0387",
           mergeFigs = TRUE,
           minAcc = 5, outlier.rm = TRUE, ...)
```

Arguments

hap	object of hapResult class, generate withvcf2hap() or seqs2hap()
pheno	object of data.frame class, imported by import_pheno()
phenoName	pheno name for plot, should be one column name of pheno
hapPrefix	prefix of hapotypes, default as "H"
title	a charater which will used for figure title
mergeFigs	bool type, indicate whether merge the heat map and box plot or not. Default as TRUE
minAcc	If observations number of a Hap less than this number will not be compared with others or be ploted. Should not less than 3 due to the t-test will meaninglessly. default as 5
outlier.rm	whether remove ouliers, default as TRUE
...	options will pass to ggpubr()

Value

list. A list contains a character vector with Haps were applied student test, a mattrix 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)
```

---

hapVsPhenos *hapVsPhenos*

Description

hapVsPhenos

Usage

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

Arguments

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

`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 charater 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 seperate charcter `filename.sep`, pheno name, a dot and surfix `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 surfix `filename.surfix`

`outlier.rm`
: whether remove ouliers, default as TRUE

`...`
: options will pass to `ggpubr()`

Value

No return value

Examples

```
data("geneHapR_test")

oriDir <- getwd()
setwd(tempdir())
# 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

```
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.

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 delimed table file

Usage

```
import_AccINFO(file, comment.char = "#",
               check.names = FALSE, row.names = 1, ...)
```

Arguments

`file`	file path, this file should be a tab delimed 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

```
oldDir <- getwd()
setwd(tempdir())
data("geneHapR_test")
write.table(pheno, file = "test.pheno.txt", sep = "\t")
pheno <- import_AccINFO("test.pheno.txt")
pheno
setwd(oldDir)
```

`import_gff` *Import Annotations from GFF Format File*

Description

import genome annotations in GFF/GFF3 format

Usage

```
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

```
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 addtion, the accessions of each haplotype in hap summary result were separated by ";".

Value

hapSummary or hapResult

Examples

```
oldDir <- getwd()
setwd(tempdir())
data("geneHapR_test")
write.hap(hapResult, file = "test.pheno.txt", sep = "\t")
hap <- import_hap("test.pheno.txt")
hap
setwd(oldDir)
```

import_MultipleAlignment *Import MultipleAlignment Result*

Description

import sequences algned 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_seqs *Import Sequences*

Description

import DNA sequences in FASTA format

Usage

```
import_seqs(filepath, format = "fasta")
```

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()`

Examples

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

---

network *Generate Haplotype Net Relationshop with Haplotype Result*

Description

computes a haplotype network with haplotype summary result

Usage

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

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 mattrix 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

haplonet class

References

Mark P.J. van der Loo (2014) doi:10.32614/RJ-2014-011;

E. Paradis (2010) doi:10.1093/bioinformatics/btp696

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
```

---

plotEFF *plotEFF*

Description

plotEFF

Usage

```
plotEFF(siteEFF, gff = gff,
        Chr = Chr, start = start, end = end,
        showType = c("five_prime_UTR", "CDS", "three_prime_UTR"),
        CDS.height = 1, cex = 0.1, col = col, pch = 20,
        main = main, legend.cex = 0.8, legend.ncol = legend.ncol,
        markMutants = TRUE, mutants.col = 1, mutants.type = 1,
        ylab = "effect")
```

Arguments

siteEFF	matrix, column name are pheno names and row name are site position
gff	gff
Chr	the chromosome name
start	start postion

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 controls points color, see points()
pch	vector controls points type, see par()
main	main title
legend.cex	a numeric control the legend size
legend.ncol	the number of columns in which to set the legend items
markMutants	whether mark mutants on gene model, default as TRUE
mutants.col	color of lines which mark mutants
mutants.type	a vector of line types
ylab	character, yaxis label

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 *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 = 1,
            backGround = backGround,
            hapGroup = hapGroup,
            legend = FALSE,
            main = main,
            labels = TRUE,
            ...)
```

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.
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)
...	other parameters will pass to `plot` function

Value

No return value

See Also

`hap_summary()` and `get_hapNet()`.

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
```

---

plotHapTable *plotHapTable*

Description

display hap result as a table-like figure

Usage

```
plotHapTable(hapSummary,
             hapPrefix = "H",
             title = "",
             geneName = geneName,
             INFO_tag = INFO_tag,
             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 "i*" 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 hapSummarys were generated from sequences, INFO row is null. If hapSummarys were generated from VCF, INFO was take from the INFO column in the source VCF file. Some tag-values may contains 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

allign imported sequences

Usage

```
seqs2hap(seqs,
         Ref = names(seqs)[1],
         hyb_remove = TRUE, na.drop = TRUE,
         maxGapsPerSeq = 0.25,
         hapPrefix = "H", ...)

allignSeqs(seqs, ...)

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
hyb_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"
...	parameters will pass to muscle::muscle()
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

See Also

muscle::muscle()

Examples

```
data("geneHapR_test")
seqs <- allignSeqs(seqs)
seqs <- trimSeqs(seqs,
         minFlankFraction = 0.1)
hapResult <- seqs2hap(seqs,
                      Ref = names(seqs)[1],
                      hyb_remove = TRUE, na.drop = TRUE,
                      maxGapsPerSeq = 0.25,
                      hapPrefix = "H")
```

SetATGas0 *Set Position of ATG as Zero*

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	Chromsome 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
newgff <- 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 whther 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 determinate by the phenotype difference between each site geno-type.

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 phynotype, eg.: height, with at least 30 observations per geno-type 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 geno-types per site.

Examples

```
data("geneHapR_test")

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

---

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,
        Chr = Chr,
        filter_POS = FALSE,
        startPOS = startPOS,
        endPOS = endPOS,
        hyb_remove = TRUE,
        na.drop = TRUE)
```

Arguments

`vcf`	vcfR object imported by `import_vcf()`
`hapPrefix`	prefix of hap names, default as "H"
`filter_Chr`	logical, whether filter vcf by chromosome or not. Default as `FALSE`. If set as `TRUE`, `Chr` is needed
`Chr`	Chromosome name, needed when `filter_Chr` was set as `TRUE`
`filter_POS`	logical, whether filter vcf by position or not. Default as `FALSE`. If set as `TRUE`, `startPOS` and `endPOS` are needed
`startPOS, endPOS`	start and end position, needed when `filter_POS` was set as `TRUE`. In addition, `startPOS` must less than `endPOS`
`hyb_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`.

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()`

Examples

```
data("geneHapR_test")
hapResult <- vcf2hap(vcf)
```

---

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"

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 addtion, the accessions of each haplotype in hap summary result were separated by ";".

Value

No return value

Examples

```
oriDir <- getwd()
setwd(tempdir())
data("geneHapR_test")
write.hap(hapResult, file = "hapResult.txt")
setwd(oriDir)
```

Index