Package ‘crosshap’

May 2, 2023

Type Package

Title Local Haplotype Clustering and Visualization

Version 1.2.2

Maintainer Jacob Marsh <jake.marsh@live.com.au>

Description A local haplotyping visualization toolbox to capture major patterns of co-inheritance between clusters of linked variants, whilst connecting findings to phenotypic and demographic traits across individuals. ‘crosshap’ enables users to explore and understand genomic variation across a trait-associated region. For an example of successful local haplotype analysis, see Marsh et al. (2022) <doi:10.1007/s00122-022-04045-8>.

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.2.3

Imports cli, clustree, data.table, dbscan, dplyr, ggdist, ggplot2, ggpp, gridExtra, gttable, magrittr, patchwork, rlang, scales, tibble, tidyr

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0), umap, vdiffr

Depends R (>= 4.00)

URL https://jacobimarsh.github.io/crosshap/

License MIT + file LICENSE

Config/testthat/edition 3

NeedsCompilation no

Author Jacob Marsh [aut, cre] (<https://orcid.org/0000-0003-3734-2023>), Brady Johnston [aut] (<https://orcid.org/0000-0001-6301-2269>), Jakob Petereit [aut] (<https://orcid.org/0000-0003-2159-0388>)

Repository CRAN

Date/Publication 2023-05-02 07:50:08 UTC
R topics documented:

- arith_mode
- build_bot_halfeyeplot
- build_left_alleleplot
- build_left_posplot
- build_mid_dotplot
- build_right_clusterplot
- build_right_phenoplot
- build_summary_tables
- build_top_metaplot
- clustree_viz
- crosshap_viz
- HapObject
- LD
- mean_na.rm
- metadata
- pheno
- prepare_hap_umap
- pseudo_haps
- read_LD
- read_metadata
- read_pheno
- read_vcf
- run_haplotyping
- run_hdbscan_haplotyping
- tagphenos
- vcf

Index

| arith_mode | Mode utility function |

Description

Mode utility function

Usage

arithmetic_mode(x)

Arguments

- x: Input vector

Value

Mode numerical values
**Description**

`build_bot_halfeyeplot()` builds a vertical plot displaying the phenotypic scores for each individual, grouped by haplotype, coloured by metadata variable. Metadata groups can be isolated using the `isolate_groups` argument. Makes use of the `$Indfile` information from haplotype object. It is an internal function called by `crosshap_viz()`, though can be called separately to build a stand-alone plot.

**Usage**

```r
build_bot_halfeyeplot(
  HapObject,
  epsilon,
  hide_labels = TRUE,
  isolate_group = NA
)
```

**Arguments**

- **HapObject**: Haplotype object created by `run_haplotyping()`.
- **epsilon**: Epsilon to visualize haplotyping results for.
- **hide_labels**: If TRUE, legend is hidden.
- **isolate_group**: If a Metadata group is provided, all other Metadata groups will be masked from the plot. NOTE: it does change the summary tables or marker group phenotype scores.

**Value**

A `ggplot2` object.

**Examples**

```r
build_bot_halfeyeplot(HapObject, epsilon = 0.6, hide_labels = FALSE)
```
Description

build_left_alleleplot() builds a horizontal plot displaying mean allelic frequencies (reference/alternate/missing/heterozygous) of all SNP loci, grouped by marker group. Makes use of $Varfile information from a HapObject created by run_haplotyping(). This is an internal function called by crosshap_viz(), though can be called separately to build a stand-alone plot.

Usage

build_left_alleleplot(HapObject, epsilon, hide_labels = TRUE)

Arguments

HapObject  Haplotype object created by run_haplotyping.
epsilon    Epsilon matching the haplotype object used for umap_in.
hide_labels If TRUE, legend is hidden.

Value

A ggplot2 object.

Examples

build_left_alleleplot(HapObject, epsilon = 0.6, hide_labels = FALSE)

Description

build_left_posplot() builds a horizontal plot displaying the chromosomal position of each SNP locus, grouped by marker group. Makes use of the $Varfile file from haplotype object. It is an internal function called by crosshap_viz(), though can be called separately to build a stand-alone plot.

Usage

build_left_posplot(HapObject, epsilon, hide_labels = TRUE)
build_mid_dotplot

Arguments

HapObject  Haplotype object created by run_haplotyping().
epsilon    Epsilon matching the haplotype object used for umap_in.
hide_labels If TRUE, legend is hidden.

Value

A ggplot2 object.

Examples

build_mid_dotplot(HapObject, epsilon = 0.6, hide_labels = FALSE)

---

Description

build_mid_dotplot() builds a central dot plot displaying the relationship between haplotype combinations and the characteristic marker group alleles that define them. Makes use of the $Hapfile information from a haplotype object. This is an internal function called by crosshap_viz(), though can be called separately to build a stand-alone plot (can be useful when patched to a peripheral plot).

Usage

build_mid_dotplot(HapObject, epsilon, hide_labels = FALSE)

Arguments

HapObject  Haplotype object created by run_haplotyping
epsilon    Epsilon to visualize haplotyping results for.
hide_labels If TRUE, legend is hidden.

Value

A ggplot2 object.

Examples

build_mid_dotplot(HapObject, epsilon = 0.6, hide_labels = FALSE)
**build_right_clusterplot**

*Right intra-cluster linkage plot*

**Description**

build_right_jitterplot() builds a horizontal plot displaying the mean pairwise $R^2$ linkage between each SNP and all other SNPs in its marker group, grouped by marker group, coloured by alternate allele frequency. Makes use of the $Varfile$ information from haplotyping object. It is an internal function called by crosshap_viz(), though can be called separately to build a stand-alone plot.

**Usage**

```r
build_right_clusterplot(HapObject, epsilon, hide_labels = FALSE)
```

**Arguments**

- `HapObject`: Haplotype object created by run_haplotyping().
- `epsilon`: Epsilon to visualize haplotyping results for.
- `hide_labels`: If TRUE, legend is hidden.

**Value**

A ggplot2 object.

**Examples**

```r
build_right_clusterplot(HapObject, epsilon = 0.6, hide_labels = FALSE)
```

**build_right_phenoplot**

*Right SNP-phenophen plot*

**Description**

build_right_phenoplot() builds a horizontal plot displaying the mean difference in phenotype score between individuals with the alternate vs reference alleles for each SNP locus, grouped by marker group, coloured by the alternate allele frequency of each SNP. Makes use of the $Varfile$ phenotypic information from haplotyping object. It is an internal function called by crosshap_viz(), though can be called separately to build a stand-alone plot.

**Usage**

```r
build_right_phenoplot(HapObject, epsilon, hide_labels = TRUE)
```
**build_summary_tables**

**Arguments**

- **HapObject**  
  Haplotype object created by run_haplotyping().

- **epsilon**  
  Epsilon to visualize haplotyping results for.

- **hide_labels**  
  If TRUE, legend is hidden.

**Value**

A ggplot2 object.

**Examples**

```r
build_right_phenoplot(HapObject, epsilon = 0.6, hide_labels = FALSE)
```

---

**build_summary_tables**  
*Hap/MG summary tables*

**Description**

build_summary_tables() builds summary tables for each haplotype and Marker Group with some of the information shown in the peripheral crosshap plots. It is an internal function called by crosshap_viz(), though can be called separately to build stand-alone grob tables.

**Usage**

```r
build_summary_tables(HapObject, epsilon)
```

**Arguments**

- **HapObject**  
  Haplotype object created by run_haplotyping().

- **epsilon**  
  Epsilon to visualize haplotyping results for.

**Value**

A list containing two TableGrob objects.
**build_top_metaplot**  
*Top metadata-hap bar plot*

Description

build_top_metaplot() builds a vertical stacked bar plot displaying the frequency of each haplotype combination, broken down by each categorical metadata variable provided. Makes use of the $Ind-file information from a haplotype object. This is an internal function called by crosshap_viz(), though can be called separately to build a stand-alone plot.

Usage

```r
build_top_metaplot(HapObject, epsilon, hide_labels = FALSE)
```

Arguments

- `HapObject`: Haplotype object created by run_haplotyping()
- `epsilon`: Epsilon to visualize haplotyping results for.
- `hide_labels`: If TRUE, legend is hidden.

Value

A ggplot2 object.

Examples

```r
build_top_metaplot(HapObject, epsilon = 0.6, hide_labels = FALSE)
```

---

**clustree_viz**  
*Clustering tree*

Description

clustree_viz() builds a clustering tree displaying changes in haplotype assignment between individuals or changes in Marker Group assignment for SNPs, across different epsilon values. This function is a ‘clustree’ wrapper.

Usage

```r
clustree_viz(HapObject, type = "MG")
```
**crosshap_viz**

**Arguments**

- **HapObject**: A haplotyping object with a range of results from different epsilons created by `run_haplotyping()`.  
- **type**: When type = "hap", nodes represent haplotype populations, when type = "MG", nodes represent marker groups.

**Value**

A ggplot2 object.

---

**crosshap_viz**  
**Visualize haplotypes**

**Description**

crosshap_viz() builds five individual plots using various elements of a HapObject created by run_haplotyping(). The central dotplot displays relationship between clusters of linked SNPs (Marker Groups), and distinct haplotypes present within the population. Vertical plots (top/bottom) visualize individuals and populations, grouped by haplotype. Horizontal plots (left/right) visualize SNP information, grouped by Marker Group cluster.

**Usage**

crosshap_viz(  
  HapObject,  
  epsilon,  
  plot_left = "allele",  
  plot_right = "pheno",  
  hide_labels = FALSE,  
  isolate_group = NA  
)

**Arguments**

- **HapObject**: Haplotype object created by run_haplotyping().  
- **epsilon**: Epsilon to visualize haplotyping results for.  
- **plot_left**: When plot_left = "allele", SNP allele frequency information is displayed, when plot_left = "pos", SNP position information is displayed.  
- **plot_right**: When plot_right = "pheno", phenotype associations for SNPs are displayed, when plot_right = "cluster", internal marker group linkage is displayed.  
- **hide_labels**: When TRUE, legends from plots are hidden.  
- **isolate_group**: If one or more Metadata groups are provided, all other Metadata groups will be masked from the plot. NOTE: it does change the summary tables or marker group phenotype scores.
Value

A patchwork object.

---

HapObject

Example Haplotype object

---

Description

A haplotyping object created by run_haplotyping() for example cqProt-003 soy data

Usage

HapObject

Format

A haplotype (S3) object containing results needed for haplotype visualization across five epsilon values (0.2,0.4,0.6,0.8,1)

- **epsilon**: Epsilon value chosen for haplotyping with DBSCAN
- **MGmin**: MGmin value (minPts) chosen for haplotyping with DBSCAN
- **Hapfile**: Summary of Marker Groups defining haplotype combinations
- **Indfile**: Haplotype assignments for individuals
- **Varfile**: Marker Group assignments for SNPs, with additional calculated information

---

LD

Example LD matrix

---

Description

A pairwise $R^2$ linkage matrix generated by PLINK for example cqProt-003 soy data

Usage

LD

Format

A square matrix read in by read_LD()
**mean_na.rm**

---

**Mean utility function**

---

**Description**

Mean utility function

**Usage**

\[\text{mean}_\text{na.rm}(x)\]

**Arguments**

- **x**: Input vector

**Value**

Mean numerical values

---

**metadata**

**Example Domestication metadata**

---

**Description**

Metadata file with level of domestication for each individual in example cqProt-003 soy data

**Usage**

\[\text{metadata}\]

**Format**

A two-column tibble read in by read_metadata()

**Source**

https://doi.org/10.1007/s00122-022-04045-8
### pheno

**Example phenotype data**

**Description**

Seed protein scores for each individual in example cqProt-003 soy data

**Usage**

```r
pheno
```

**Format**

A two-column tibble read in by `read_pheno()`

**Source**

[https://doi.org/10.1007/s00122-022-04045-8](https://doi.org/10.1007/s00122-022-04045-8)

---

### prepare_hap_umap

**UMAP haplotype visualization helper**

**Description**

prepare_hap_umap() builds a large composite ggplot2 object ready for faceting and animation (see vignette) for visualizing SNP alleles (coloured by Marker Group) possessed by individuals with each haplotype. UMAP coordinates for each SNP can be generated using `umap::umap()`, with the LD matrix generated for `run_haplotyping()` as input. When fully rendered and faceted, the resultant GIF intuitively visualizes the shared loci within each Marker Group that are constant within each haplotype combination.

**Usage**

```r
prepare_hap_umap(
  umap_in,
  hetmiss_as = "allele",
  HapObject,
  epsilon,
  vcf,
  nsamples = 25
)
```
**pseudo_haps**

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>umap_in</code></td>
<td>UMAP results produced for a haplotype object at a given epsilon.</td>
</tr>
<tr>
<td><code>hetmiss_as</code></td>
<td>If <code>hetmiss_as = &quot;allele&quot;</code>, heterozygous-missing SNPs <code>./N</code> are recoded as <code>N/N</code>, if <code>hetmiss_as = &quot;miss&quot;</code>, the site is recoded as missing.</td>
</tr>
<tr>
<td><code>HapObject</code></td>
<td>Haplotype object created by <code>run_haplotyping()</code>.</td>
</tr>
<tr>
<td><code>epsilon</code></td>
<td>Epsilon matching the haplotype object used for <code>umap_in</code>.</td>
</tr>
<tr>
<td><code>vcf</code></td>
<td>Input vcf.</td>
</tr>
<tr>
<td><code>nsamples</code></td>
<td>Number of times to sample each haplotype group, will directly translate to the number of frames in animation. Should be the same as the <code>nframes</code> passed to <code>gganimate::animate()</code>.</td>
</tr>
</tbody>
</table>

**Value**

A large `ggplot2` object.

---

**pseudo_haps**

Identify haplotypes from clustered SNPs

**Description**

`pseudo_haps()` calls the most common allelic states for each SNP marker group across individuals, before building dummy SNPs for each marker group that mimic the binary vcf format. This is the step which determines the haplotype combinations, and therefore enables several summaries to be returned - as contained in the `$Hapfile` and preliminary `$Indfile` and finalised `$MGfile`, following marker group smoothing. This is an internal function not intended for external use.

**Usage**

`pseudo_haps(preMGfile, bin_vcf, minHap, LD, keep_outliers)`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>preMGfile</code></td>
<td>SNP clusters from DBscan.</td>
</tr>
<tr>
<td><code>bin_vcf</code></td>
<td>Binary VCF for region of interest reformatted by <code>run_haplotyping()</code>.</td>
</tr>
<tr>
<td><code>minHap</code></td>
<td>Minimum size (<code>nIndividuals</code>) to keep haplotype combinations</td>
</tr>
<tr>
<td><code>LD</code></td>
<td>LD matrix input.</td>
</tr>
<tr>
<td><code>keep_outliers</code></td>
<td>When FALSE, marker group smoothing is performed to remove outliers.</td>
</tr>
</tbody>
</table>

**Value**

Returns intermediate of haplotype object
### read_LD

**Read LD correlation matrix to tibble**

**Description**

If your correlation matrix does not have rownames and column names, a VCF will need to be provided so it can be added with `read_LD()`.

**Usage**

```r
read_LD(LDin, vcf = NULL)
```

**Arguments**

- **LDin** Square correlation matrix
- **vcf** VCF object created by `read_vcf()` that can be used to assign column names

**Value**

A tibble.

---

### read_metadata

**Read metadata to tibble**

**Description**

Requires two column text file without a header (Ind | Metadata)

**Usage**

```r
read_metadata(Metain)
```

**Arguments**

- **Metain** Input phenotype file

**Value**

A tibble.
read_pheno  

**Read phenotype data to tibble**

**Description**

Requires two column text file without a header (Ind | Pheno)

**Usage**

```r
read_pheno(Phenoin)
```

**Arguments**

- **Phenoin**
  - Input phenotype file

**Value**

A tibble.

---

read_vcf  

**Read VCF to tibble**

**Description**

Dashes, '-', in individual names are recoded to '.' for downstream compatibility.

**Usage**

```r
read_vcf(VCFin)
```

**Arguments**

- **VCFin**
  - Input VCF

**Value**

A tibble.
run_haplotyping

Description

run_haplotyping() performs density-based clustering of SNPs in region of interest to identify Marker Groups. Individuals are classified by haplotype combination based on shared combinations of Marker Group alleles. Returns a haplotyping object (HapObject), which can be used as input to build clustering tree for epsilon optimization using clustree_viz(), and can be visualized with reference to phenotype and metadata using crosshap_viz().

Usage

run_haplotyping(
    vcf,
    LD,
    pheno,
    metadata = NULL,
    epsilon = c(0.2, 0.4, 0.6, 0.8, 1),
    MGmin = 30,
    minHap = 9,
    hetmiss_as = "allele",
    het_phenos = FALSE,
    keep_outliers = FALSE
)

Arguments

vcf Input VCF for region of interest.
LD Pairwise correlation matrix of SNPs in region (e.g. from PLINK).
pheno Input numeric phenotype data for each individual.
metadata Metadata input (optional).
epsilon Epsilon values for clustering SNPs with DBscan.
MGmin Minimum SNPs in marker groups, MinPts parameter for DBscan.
minHap Minimum nIndividuals in a haplotype combination.
hetmiss_as If hetmiss_as = "allele", heterozygous-missing SNPs "./N" are recoded as "N/N", if hetmiss_as = "miss", the site is recoded as missing.
het_phenos When FALSE, phenotype associations for SNPs are calculated from reference and alternate allele individuals only, when TRUE, heterozygous individuals are included assuming additive allele effects.
keep_outliers When FALSE, marker group smoothing is performed to remove outliers.

Value

A comprehensive haplotyping S3 object (HapObject) for each provided epsilon value, needed for clustree_viz() and crosshap_viz().
**Description**

`run_hdbscan_haplotyping()` performs HDBSCAN clustering of SNPs in region of interest to identify marker groups. Individuals are classified by haplotype combination based on shared combinations of marker group alleles. Returns a comprehensive haplotyping object (HapObject), which can be visualized with reference to phenotype and metadata using `crosshap_viz()` (set epsilon to 1 as a dummy value).

**Usage**

```r
run_hdbscan_haplotyping(
  vcf,
  LD,
  pheno,
  MGmin,
  minHap = 5,
  hetmiss_as = "allele",
  metadata = NULL,
  keep_outliers = FALSE
)
```

**Arguments**

- `vcf`: Input VCF for region of interest.
- `LD`: Pairwise correlation matrix of SNPs in region (e.g. from PLINK).
- `pheno`: Input numeric phenotype data for each individual.
- `MGmin`: Minimum SNPs in marker groups, MinPts parameter for DBscan.
- `minHap`: Minimum nIndividuals in a haplotype combination.
- `hetmiss_as`: If hetmiss_as = "allele", heterozygous-missing SNPs ’/N’ are recoded as ’N/N’, if hetmiss_as = "miss", the site is recoded as missing.
- `metadata`: Metadata input (optional).
- `keep_outliers`: When FALSE, marker group smoothing is performed to remove outliers.

**Value**

A comprehensive haplotyping S3 object (HapObject) for each provided epsilon value, needed for `clustree_viz()` and `crosshap_viz()`.
tagphenos

**Calculate SNP phenotypic associations**

**Description**

`tagphenos()` reports the frequency of allele types for each SNP and calculates phenotype associations for the different alleles, before returning this information in a $Varfile in a HapObject. This is an internal function that is not intended for external use.

**Usage**

```
tagphenos(MGfile, bin_vcf, pheno, het_phenos = FALSE)
```

**Arguments**

- `MGfile`: SNP marker groups clustered using DBscan.
- `bin_vcf`: Binary VCF for region of interest reformatted by run_haplotyping().
- `pheno`: Input numeric phenotype data for each individual.
- `het_phenos`: When FALSE, phenotype associations for SNPs are calculated from reference and alternate allele individuals only, when TRUE, heterozygous individuals are included assuming additive allele effects.

**Value**

Returns intermediate of haplotype object.

---

vcf

**Example VCF**

**Description**

A VCF containing SNPs for example cqProt-003 soy data

**Usage**

```
vcf
```

**Format**

A VCF read in by read_vcf()

**Source**

https://doi.org/10.1007/s00122-022-04045-8
Index

* datasets
  HapObject, 10
  LD, 10
  metadata, 11
  pheno, 12
  vcf, 18

* data
  HapObject, 10
  LD, 10
  metadata, 11
  pheno, 12
  vcf, 18

arith_mode, 2

build_bot_halfeyeplot, 3
build_left_alleleplot, 4
build_left_posplot, 4
build_mid_dotplot, 5
build_right_clusterplot, 6
build_right_phenoplot, 6
build_summary_tables, 7
build_top_metaplot, 8

clustree_viz, 8
crosshap_viz, 9

HapObject, 10

LD, 10

mean_na.rm, 11
metadata, 11

pheno, 12
prepare_hap_umap, 12
pseudo_haps, 13

read_LD, 14
read_metadata, 14
read_pheno, 15
read_vcf, 15
run_haplotyping, 16
run_hdbscan_haplotyping, 17
tagphenos, 18
vcf, 18