Package 'mapfuser'

October 10, 2017

Type Package

Title Construct Consensus Genetic Maps and Estimate Recombination Rates

Version 0.1.2

Author Dennis van Muijen, Ram Kumar Basnet, Nathalie J. Dek

Maintainer Dennis van Muijen <d.van.muijen@rijkzwaan.nl>

Description Construct consensus genetic maps with LPmerge, see Endelman and Plomion (2014) <doi:10.1093/bioinformatics/btu091> and model the relationship between physical distance and genetic distance using thin-plate regression splines, see Wood (2003) <doi:10.1111/1467-9868.00374>. Perform quality control on input data and visualise intermediate steps.

Depends igraph

Imports dplyr, ggplot2, mgcv, doParallel, parallel, foreach, stringi, plotly, visNetwork, LPmerge, lazyeval, tidyr

License GPL-3

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository CRAN

Date/Publication 2017-10-10 10:13:54 UTC

R topics documented:

`calc_RMSE` *Internal function to calculate the Root Mean Square Error to select the maximum interval size.*

Description

The LPmerge algorithm is called for all linkage groups with an option for multicore processing. mapfuser further adds automatic selection of the lowest max

Usage

```
calc_RMSE(lp_res, max.interval)
```

Arguments

`lp_res`	Result of the mapfuser foreach call or alternatively the result of a call to LPmerge
`max.interval`	A whole number specifying the maximum interval size between bins to included in the objective function for LPmerge

Value

A list of calculated Root Mean Square Error (RMSE) per chromosome with RMSE for each linkage group ID compared to the consensus map. The mean and standard deviation over the linkage group IDs is also provided.

Author(s)

Dennis van Muijen

check_anchors *Replace sub-linkage groups (e.g. 1.1, 1.2) with truncated linkage group number and split per chromosome for easy integration*

Description

Replace sub-linkage groups (e.g. 1.1, 1.2) with truncated linkage group number and split per chromosome for easy integration

Usage

```
check_anchors(MF.obj = MF.obj, anchors = anchors)
```

Arguments

MF.obj	mapfuser object
anchors	Number of uniquely positioned anchor markers between genetic maps

Value

mapfuser object with QC$maps list item containing input maps splitted to unique linkage groups

check_cM *internal position check*

Description

internal position check

Usage

```
check_cM(x)
```

Arguments

x	vector of positions

Value

index of non-numeric items

Author(s)

Dennis van Muijen

genphys_fit *Model the relationship between genetical and physical genome positions*

Description

Fit a penalized thin plate regression spline (Wood, 2003) through genetic position in centiMorgan and physical genome positions in (Mega) base pairs.

Usage

```
genphys_fit(MF.obj, type = c("consensus", "map"), z = 5,
  chromosomes = NULL, map = NULL)
```

Arguments

MF.obj	A mapfuser object with a reference map loaded and either a consensus map created with mapfuser or a genetic loaded with read.maps
type	Fit physical genome positions vs. consensus genetic map made with mapfuser or an individual genetic map
z	discard invididual data points based on z-score threshold for scaled pearson residuals
chromosomes	The chromosomes to fit a P-spline, default to all chromosomes
map	Name of the genetic map to use when the consensus map is not used for fitting a P-spline and recombination rate calculation

Value

The input object is returned with added components recombination rate at a 0.1 Mbp interval, the general additive model fit (gam) with penalized spline fits per chromosome, and predictions of centiMorgan positions used for fitting. Markers that have been removed due to z-threshold are saved to the config slot.

Author(s)

Dennis van Muijen

Examples

```
## Not run:
MF.obj <- genphys_fit(MF.obj, type = "consensus", z = 5, chromosomes = 1:5, map = NULL)
MF.obj <- genphys_fit(MF.obj, type = "map", z = 5, chromosomes = 1:5, map = "Col-0_Blh-1.csv")
# Plot the result
plot(MF.obj, which = "mareymap", maps = "consensus", chr = 1:5)

## End(Not run)
```

LPmerge_par *Wrapper for multicore and multichromosome merging of maps using the LPmerge*

Description

The LPmerge algorithm is called for all linkage groups with an option for multicore processing. The mapfuser package further adds automatic selection of max.interval giving the lowest RMSE

Usage

```
LPmerge_par(MF.obj, n.cores = 2, max.interval = 1:3, max.int_sel = "auto",
  weights = NULL)
```

Arguments

MF.obj	A mapfuser object genetics maps loaded and optionally a reference map
n.cores	number of cores
max.interval	A whole number specifying the maximum interval size between bins to include in the objective function. An array of numbers can be passed to test different values (one consensus map is produced for each value in the array).
max.int_sel	Either automatically select the max.interval with the lowest RMSE for each linkage group or specify manually a vector of values of max.interval to select for the output
weights	Optional vector of length T containing the weights for each map in the objective function (see details). If not passed, the maps are given equal weight.

Value

Three items added to the mapfuser object under the results slot. 1) The consensus map at the selected max.interval, either manually specified or automatically select 2) A list of calculated Root Mean Square Error (RMSE) per chromosome with RMSE for each linkage group ID compared to the consensus map. THe mean and standard deviation over the linkage group IDs is also provided. 3) A list with length equal to the length of the max.interval parameter. Each entry in the list is a data frame containing the consensus map and the component linkage maps.

Author(s)

Dennis van Muijen

References

Endelman, JB, and C Plomion. 2014. LPmerge: An R package for merging genetic maps by linear programming. Bioinformatics 30:1623-1624.

Examples

```
## Not run:
MF.obj <- LPmerge_par(MF.obj = MF.obj, n.cores = 2,
max.interval = 1, max.int_sel = "auto", weights = NULL)
# Plot result
plot(MF.obj, which = "single_map", maps = "consensus")
# Access RMSE table
MF.obj$result$RMSE

## End(Not run)
```

`map_cohesion` *Graph check whether maps can be integrated*

Description

Graph check whether maps can be integrated

Usage

```
map_cohesion(MF.obj)
```

Arguments

`MF.obj`	A mapfuser object

Value

The input mapfuser object if check OK, error otherwise

Author(s)

Dennis van Muijen

`map_export` *Convert a consensus genetic map created with mapfuser to JoinMap format*

Description

Writes the consensus map to the JoinMap ".map" format

Usage

```
map_export(MF.obj, file = NULL)
```

Arguments

MF.obj	The mapfuser object with filled results slot
file	Path to the output file

Author(s)

Dennis van Muijen

Examples

```
## Not run:
## Read maps
fpath <- system.file("extdata", package="mapfuser")
maps <- list.files(fpath, pattern = "Col", full.names = TRUE)
MF.obj <- read_maps(mapfiles = maps, sep = ",", header = TRUE,
mapweights = rep(1,7), type = "delim")

## Run map_qc
MF.obj <- map_qc(MF.obj, anchors = 3)

## Construct consensus map
MF.obj <- LPmerge_par(MF.obj = MF.obj, n.cores = 2,
max.interval = 1:3, max.int_sel = "auto", weights = NULL)

## Export to JoinMap format
file <- paste(tempdir(), "/consensus.map", sep="")
map_export(MF.obj, file)

## End(Not run)
```

---

map_flip *Invert a linkage group*

Description

Invert a linkage group

Usage

```
map_flip(x)
```

Arguments

x	Linkage group to invert

Value

Linkage group with inverted genetic positions

Author(s)

Dennis van Muijen

`map_orient` *Corrects orientation of input genetic maps*

Description

Corrects orientation of input genetic maps

Usage

```
map_orient(MF.obj)
```

Arguments

`MF.obj`	A mapfuser object

Value

A mapfuser object with corrected map orientation

Author(s)

Dennis van Muijen

`map_qc` *Wrapper function of genetic map cleaning per linkage group.*

Description

The raw data is first splitted to separate linkage groups if sublinkage groups exist (e.g LG 1.1 and 1.2). Subsequently a graph is created from the adjacency matrix that counts the number of overlapping markers between the set of genetic maps. Calculations are performed for each chromosome separately. Taken quality control steps are printed to the console and can be visualised using the plot function.

Usage

```
map_qc(MF.obj, anchors = 3)
```

Arguments

`MF.obj`	A mapfuser object genetics maps loaded and optionally a reference map
`anchors`	Number of minimum overlapping anchors marker between at least one other genetic map. At least 3 are required.

Value

The input object is returned with filled QC slot containing genetic maps after quality control. Used parameters and inverted or names of removed data are saved to the config slot.

Author(s)

Dennis van Muijen

Examples

```
## Not run:
MF.obj <- map_qc(MF.obj = MF.obj, anchors = 3)
#Graphical overview of how different genetic maps are connected by overlapping markers
plot(MF.obj, which = "mapnetwork", chr = 1) ## Multiple chromosomes not supported
## A minimal spanning tree using the number of anchors as edge weight,
plot(MF.obj, which = "mst", chr = 1)
#Visualize inverted maps
plot(MF.obj, which = "genetic_maps", maps = c("Col-0_Cvi-0.csv","Col-0_Sha.csv"), chr = 1:3)

## End(Not run)
```

---

map_split *Replace sub-linkage groups (e.g. 1.1, 1.2) with truncated linkage group number and split per chromosome for easy integration*

Description

Replace sub-linkage groups (e.g. 1.1, 1.2) with truncated linkage group number and split per chromosome for easy integration

Usage

```
map_split(MF.obj)
```

Arguments

MF.obj	A mapfuser object

Value

mapfuser object with QC$maps list item containing input maps splitted to unique linkage groups

plot.mapfuser *Visualise mapfuser object data*

Description

Visualise mapfuser object data

Usage

```
## S3 method for class 'mapfuser'
plot(x, which = c("mapnetwork", "mst", "compare_maps",
  "single_map", "mareymap", "recombination_rate"), chr = NULL, maps = NULL,
  ...)
```

Arguments

x	A mapfuser object with a reference map loaded and either a consensus map created with mapfuser or a genetic loaded with read_maps
which	Dataset to visualize: 1) which = "mapnetwork", "mst". Plots the connection in terms of overlapping markers between maps as mapnetwork and the corresponding minimum spanning tree using the inverse of the number of anchors markers as edge weight 2) which = "compare_maps". Plots a simple scatterplot between two genetic maps, 3) which = "single_map". Plot a single map per chromosome in mapchart style 4) which = "marey_map". Plot the relationship between centi Morgan positions and physical genome position along with the fitted Penalized spline for each chromosomes 5) which = "recombination_rate". Plot the recombination rate in centi Morgan per mega base pair along the physical genome.
chr	The chromosomes to plot, mapnetwork and mst is single chromosome only
maps	Name or names of the maps to plot in case which = "mareymap", "single_map", or "compare_maps"
...	ingored in function call

Value

A plot.igraph object in the case of "mst" or "mapnetwork" or an interactive ggplotly object otherwise

Author(s)

Dennis van Muijen

Examples

```
## Not run:
plot(x = MF.obj, which = "mapnetwork", chr = 1)
plot(x = MF.obj, which = "mst", chr = 1)
plot(x = MF.obj, which = "single_map", maps = "consensus")
```

```
plot(x = MF.obj, which = "compare_maps", maps = c("Col-0_Cvi-0.csv","Col-0_Sha.csv"), chr = 1:3)
plot(x = MF.obj, which = "mareymap", maps = "Col-0_Bur-0.csv", chr = 1:5)
plot(x = MF.obj, which = "mareymap", maps = "consensus", chr = 1:5)
plot(x = MF.obj, which = "recombination_rate", chr = 1:5)

## End(Not run)
```

`predict.mapfuser` *Predict centiMorgan positions from fitted gam models on the mapfuser object*

Description

Takes a fitted thin plate regression spline produced by gam() and produces predictions of centi Morgan positions using known physical genome positions and fitted gam models.

Usage

```
## S3 method for class 'mapfuser'
predict(object, to_predict, ...)
```

Arguments

`object`	A mapfuser object with fitted gam models
`to_predict`	A csv file with columns marker, chromosome ID and position in mega base pairs.
`...`	ingored in function call

Value

A data frame with columns "Marker", "Chr", "Position", and "Position_physical"

Author(s)

Dennis van Muijen

Examples

```
## Not run:
# Read a table with positions to interpolate and/or extrapolate
fpath <- system.file("extdata", package="mapfuser")
to_predict <- read.table(paste0(fpath, "/BaySha_physical.csv"), sep = ",", header = TRUE)
MF.obj <- predict(MF.obj, to_predict)
# Write to csv
write.table(MF.obj$predictions, file = "preds.csv", sep = ",", col.names = TRUE, row.names = FALSE)

## End(Not run)
```

read_joinmap *Internal function, read genetic maps in JoinMap format*

Description

Internal function, read genetic maps in JoinMap format

Usage

```
read_joinmap(mapfile)
```

Arguments

mapfile	JoinMap formatted file to read

Value

A dataframe with markers, linkage groups identifiers and genetic position on separate columns

Author(s)

Dennis van Muijen

read_maps *Read genetic maps*

Description

Reads genetic maps in either delimited or JoinMap format. Maps are loaded to a mapfuser object.

Usage

```
read_maps(mapfiles = NULL, sep = NULL, header = TRUE, na.strings = "NA",
  type = c("delim", "JoinMap"), mapweights = NULL)
```

Arguments

mapfiles	List of filenames to read
sep	The field separator character, see read.delim
header	A logical value indicating whether the files contains the names of variable int the first line, see read.delim()
na.strings	How to interpret missing values? See read.delim()
type	Type of input data, either delimited file with columns "Marker", "LG", and "Position" or a JoinMap ".map" file.
mapweights	numeric vector mapweights

Value

A mapfuser object with the genetic maps listed under the raw_data slot. Chromosome identifiers are added to the config slot

Author(s)

Dennis van Muijen

Examples

```
fpath <- system.file("extdata", package="mapfuser")
maps <- list.files(fpath, pattern = "Col", full.names = TRUE)
MF.obj <- read_maps(mapfiles = maps, sep = ",", header = TRUE,
mapweights = rep(1,7), type = "delim")
```

---

read_ref *Load a reference map*

Description

Read a reference map, either a reference genetic map or a map with physical genome positions

Usage

```
read_ref(MF.obj = NULL, ref_file = ref_file, sep = NULL, header = TRUE,
  na.strings = "NA", type = c("delim", "JoinMap"))
```

Arguments

MF.obj	A mapfuser object
ref_file	path to reference file
sep	The field separator character, see read.delim
header	A logical value indicating whether the files contains the names of variable int the first line, see read.delim()
na.strings	How to interpret missing values? See read.delim
type	Type of input data, either delimited file with columns "Marker", "LG", and "Position" or a JoinMap ".map" file.

Value

A mapfuser object with the reference map loaded to the ref_map slot. Name of the reference map is saved to the config.

Author(s)

Dennis van Muijen

Examples

```
fpath <- system.file("extdata", package="mapfuser")
maps <- list.files(fpath, pattern = "Col", full.names = TRUE)
MF.obj <- read_maps(mapfiles = maps, sep = ",", header = TRUE,
mapweights = rep(1,7), type = "delim")
ref_file <- list.files(fpath, pattern = "reference", full.names = TRUE)
MF.obj <- read_ref(MF.obj = MF.obj, ref_file = ref_file, sep = ",",
header = TRUE, na.string = NA, type = "delim")
```

---

remove_manual *Manually remove a linkage group within a specific map and update man networks and minimum spanning tree*

Description

The quality control passed list of genetic maps may be manually curated further with remove_manual() when the map merging process identified a linkage group witihn a map that gives a high Root Mean Square Error. In the case of interspecific crosses one complete map could be better to exclude all together.

Usage

```
remove_manual(MF.obj, to_remove)
```

Arguments

MF.obj	A mapfuser object genetics maps after map_orient() has been performed
to_remove	A list of linkage group ID's to remove

Value

The input object is returned in which the linkage group ID have been removed. Manually removed linkage group ID are saved to the removed_LGIDs config slot

Author(s)

Dennis van Muijen

Examples

```
fpath <- system.file("extdata", package="mapfuser")
maps <- list.files(fpath, pattern = "-1", full.names = TRUE)
MF.obj <- read_maps(mapfiles = maps, sep = ",", header = TRUE, type = "delim")
MF.obj <- map_qc(MF.obj)
## Remove two linkage groups manually
to_remove <- c("Col-0_Blh-1.csv_1","Col-0_Blh-1.csv_2" )
MF.obj <- remove_manual(MF.obj, to_remove)
```

select_result *Select result from mapfuser*

Description

Internal function for mapfuser - Flatten, simplify, and combine to one dataframe the consensus map of all linkage groups for easy inspection and exporting

Usage

```
select_result(lp_res, max.int_sel = "auto", map_RMSE, chr = chr)
```

Arguments

lp_res	Result of the mapfuser foreach call or alternatively the result of a call to LPmerge
max.int_sel	Either automatic selection of the maximum interval size that minimized the mean Root Square Mean Error between the consensus map and invidual linkage maps
map_RMSE	Output of a call to calc_RMSE
chr	chromosomes to perform analysis for

Value

The consensus map at the selected maximum interval size K in a convenient format

Author(s)

Dennis van Muijen

summary.mapfuser *internal position check*

Description

internal position check

Usage

```
## S3 method for class 'mapfuser'
summary(object, ...)
```

Arguments

object	mapfuser object
...	ingored in function call

Value

length and number of mapped marker for each input genetic map

Author(s)

Dennis van Muijen

Index