Package 'GenoTriplo'

December 4, 2024

Type Package

Title Genotyping Triploids (or Diploids) from Luminescence Data

Version 1.1.2

Description Genotyping of triploid individuals from luminescence data (marker probeset A and B). Works also for diploids.
Two main functions: Run_Clustering() that regroups individuals with a same genotype based on proximity and
Run_Genotyping() that assigns a genotype to each cluster. For Shiny interface use: launch_GenoShiny().

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports cowplot, doParallel, dplyr, DT, foreach, ggplot2, htmltools, parallel, processx, rlang, Rmixmod, shiny, shinythemes, tidyr

Depends R (>= 3.5.0), shinyBS

NeedsCompilation no

Author Julien Roche [aut, cre],
Florence Phocas [aut],
Mathieu Besson [aut],
Pierre Patrice [aut],
Marc Vandeputte [aut],
François Allal [aut],
Pierrick Haffray [aut]

Maintainer Julien Roche <julien.roche@inrae.fr>

Repository CRAN

Date/Publication 2024-12-04 13:30:15 UTC

Contents

`Clustering` *Clustering function*

Description

Clustering function to run clustering with no parallelization process nor auto save

Usage

```
Clustering(
  dataset,
  nb_clust_possible,
  n_iter = 5,
  Dmin = 0.28,
  SampleName = NULL
)
```

Arguments

`dataset`	dataset with Contrast and SigStren for each individuals (as SampleName) and each markers (as MarkerName)
`nb_clust_possible`	number of cluster possible (ploidy+1)
`n_iter`	number of iterations to perform for clustering
`Dmin`	minimal distance between two clusters
`SampleName`	vector with all SampleName (important when missing genotype)

Value

list of results of clustering

Examples

```
data(GenoTriplo_to_clust)
ploidy=3
res = Clustering(dataset=GenoTriplo_to_clust,
                 nb_clust_possible=ploidy+1,n_iter=5)
```

Create_Dataset *Create dataset in appropriate format*

Description

Create SigStren and Contrast variables from luminescence values of probeset A and B of each markers and return a dataframe to be used for clustering or save the result if a saving name is given

Usage

```
Create_Dataset(data, save_name = NULL)
```

Arguments

data	dataframe with probeset_id as first variable (markername finishing by -A or -B depending on the probeset) and individuals as variable with luminescence values for each probeset (dataset created by bash code by shiny app)
save_name	saving name

Value

number of individuals and markers (automatically save the dataset)

GenoTriplo_to_clust *Example of dataset for clustering*

Description

Example of dataset for clustering

Usage

```
GenoTriplo_to_clust
```

Format

A dataframe with 500 rows (corresponding to an individual for a given marker) and 4 columns (SigStren,Contrast,SampleName,MarkerName)

GenoTriplo_to_geno *Example of dataset for genotyping*

Description

Example of dataset for genotyping

Usage

```
GenoTriplo_to_geno
```

Format

A list of 10 each element being the result of clustering for a given marker

launch_GenoShiny *Shiny App for genotyping*

Description

Launch a shiny interface to use GenoTriplo. Really easy to use and user friendly, this will help you gain time !

Usage

```
launch_GenoShiny()
```

Value

void : most results are automatically saved

Run_Clustering *Launch parallel clustering*

Description

Launch the clustering phase in parallel from the dataset with SampleName, Contrast and SigStren for each markers (MarkerName).

Usage

```
Run_Clustering(
  data_clustering,
  ploidy,
  save_n = "",
  n_iter = 5,
  D_min = 0.28,
  n_core = 1,
  path_log = ""
)
```

Arguments

data_clustering	dataframe result from create dataset phase
ploidy	ploidy of offspring
save_n	name of the saving file
n_iter	number of iterations of clustering
D_min	threshold distance between two clusters
n_core	number of cores used for parallelization
path_log	path for log file when run by the shiny app

Value

the result of clustering or automatically save a list of objects if a saving name has been provided

Examples

```
data(GenoTriplo_to_clust)
res = Run_Clustering(data_clustering=GenoTriplo_to_clust,
                     ploidy=3,n_iter=5,n_core=1)
# or if you want to automatically save the result
# This will automatically create a folder and save the result in it
# Run_Clustering(data_clustering=GenoTriplo_to_clust,
#                ploidy=3,n_iter=5,n_core=1,save_n='exemple')
```

Run_Genotyping *Launch genotyping phase in parallel*

Description

Function that launch the genotyping phase from the dataset with SampleName, Contrast and SigStren for each markers and the result of the 'Run_clustering' function.

Usage

```
Run_Genotyping(
  data_clustering,
  res_clust,
  ploidy,
  SeuilNoCall = 0.85,
  SeuilNbSD = 2.8,
  SeuilSD = 0.28,
  n_core = 1,
  corres_ATCG = NULL,
  pop = "Yes",
  cr_marker = 0.97,
  fld_marker = 3.4,
  hetso_marker = -0.3,
  save_n = "",
  batch = "",
  ALL = TRUE,
  path_log = ""
)
```

Arguments

data_clustering	dataframe result from create dataset phase
res_clust	object from clustering phase
ploidy	ploidy of offspring
SeuilNoCall	threshold of the probability of belonging to a cluster
SeuilNbSD	threshold for the distance between an individuals and his cluster (x=Contrast)
SeuilSD	threshold for the standard deviation of a cluster (SeuilSD*(1+0.5*abs(mean_contrast_cluster)))
n_core	number of cores used for parallelization
corres_ATCG	dataframe with the correspondence between A/B of AXAS and A/T/C/G (three columns : probeset_id, Allele_A, Allele_B)
pop	Yes or No : are individuals from a same population
cr_marker	call rate threshold
fld_marker	FLD threshold
hetso_marker	HetSO threshold
save_n	name of the saving file. If " no auto save and return value is changed
batch	batch number in case of parallelization else ignore
ALL	TRUE/FALSE whether the dataset has been cut or not (from the shiny app)
path_log	path for log file when run by the shiny app

Value

if save_n != " : 3 objects list : dataframe with call rate by individuals, dataframe with call rate and other metrics of markers and another dataframe – Automatically save results. Else : return list with genotype

Examples

```
data(GenoTriplo_to_clust)
data(GenoTriplo_to_geno)
res = Run_Genotyping(data_clustering=GenoTriplo_to_clust,
                     res_clust=GenoTriplo_to_geno,
                     ploidy=3)
```

Index