Package 'pheno2geno'

July 29, 2014

Version 1.2.1

Date May 2014

Title Pheno2Geno - Generating genetic markers and maps from molecular phenotypes

Author Konrad Zych <k.zych@rug.nl> and Danny Arends <danny.arends@gmail.com>

Maintainer Konrad Zych <k.zych@rug.nl>

Description Pheno2Geno is an R package for the computational generation genetic markers and maps from molecular phenotypes

Depends R (>= 2.14.1), graphics, stats, utils, qtl, VGAM, mixtools

Suggests RankProd

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-29 10:54:13

R topics documented:

---

pheno2geno-package *Pheno2Geno - High-throughput generation of genetic markers and maps from molecular phenotypes*

Description

Pheno2geno is an R package to generate genetic markers and maps out from molecular phenotypes. Currently supported breeding schemes are: Recombinant Inbred Lines (RIL), F2 and backcross (BC).

The most important functions:

- `read.population` - Reads the files into R.
- `find.diff.expressed` - Using Rank Product or student t-test analysis to select differentially expressed genes.
- `scan.qtls` - Scanning population data for qtls for use in cross.saturate function.
- `generate.biomarkers` - Converts continous gene expression measurments into discrete genetic markers.
- `cross.denovo` - Create de novo genetic map or vector showing how chromosomes should be assigned.
- `cross.saturate` - Saturate an existing genetic map with phenotype-derived markers.

Details

Background Genetic markers and maps are instrumental for quantitative trait locus (QTL) mapping in segregating populations. The resolution of QTL localisation depends on the number of informative recombinations in the population and how well these recombinations are tagged by markers. Thus larger populations and denser marker maps do a better job. Ideally there are enough markers to pinpoint all informative recombinations in the population. In practice marker maps are often still too sparse. However, maps can be saturated or even be derived de-novo from high-throughput gene expression, protein or metabolite abundance data. A fraction of these molecular traits may show a clear multimodal distribution due to a major QTL effect and can therefore be converted into useful genetic markers. Results We developed the pheno2geno R package for high-throughput generation of genetic markers and maps from molecular phenotypes. Pheno2geno selects suitable phenotypes that show clear differential expression in the 1 founders. Mixture modelling is used to select phenotypes showing segregation ratios close to the expected mendelian segregation ratios and transform these phenotypes into genetic markers suitable for map construction and/or saturation. We demonstrate our method on 164 individuals from an A. thaliana Recombinant Inbred Line (RIL) population. We show that pheno2geno is able to saturate the existing genetic map decreasing the average distance between markers from 7.1 cM to 0.70 cM, pinpointing all recombinations in the population. Using pheno2geno we created a de-novo map from the gene expression data that is twice as dense as the original genetic map consisting of AFLP markers.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

References

Arends D., Zych K., Li Y., van der Velde K. J., Joosen R. V. L., Ligterink W. and Jansen R. C. (2013): Pheno2Geno - High-throughput generation of genetic markers and maps from molecular phenotypes. **in press**

Breitling, R.; Armengaud, P.; Amtmann, A.; and Herzyk, P.(2004) Rank Products:A simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments, **FEBS Letter**, 57383-92

Broman, K. W. and Sen, Ś. (2009) *A guide to QTL mapping with R/qtl.* **Springer**. http://www.rqtl.org/book

See Also

- read.population - Load genotype, phenotype, genetic map data files into R environment into a population object.
- find.diff.expressed - Using Rank Product or student t-test analysis to select differentially expressed genes.
- scan.qtls - Scanning population data for qtls for use in cross.saturate function.
- cross.denovo - Create de novo genetic map or vector showing how chromosomes should be assigned.
- cross.saturate - Saturate existing map.

`add.to.population` *Add additional data to a population object*

Description

Add additional data to an existing population object. When adding data already present in the population objects the function will issue a warning.

Usage

```
add.to.population(population, dataObject, dataType=c("founders",
"offspring$phenotypes",  "founders$group", "offspring$genotypes",
"maps$genetic","maps$physical","annotations"), verbose=FALSE,debugMode=0)
```

Arguments

`population`	An object of class `population`. See `create.population` for details.
`dataObject`	A matrix of data to be put into the population objects, or a list of matrices.
`dataType`	Specifies what kind of data dataObject contains, if dataObject is a list of matrices to add, dataType should be a list of the same length: • founders - Founders phenotype. • offspring$phenotypes - Offspring phenotype. • founders$group - Specifying groups in founders phenotypes. • offspring$genotypes - Offspring genotype. • maps$genetic - Genetic map. • maps$physical - Physical map. • annotations - Annotations file.
`verbose`	Be verbose.
`debugMode`	Either use 1 or 2, this will modify the amount of information returned to the user. 1) Print out checks, 2) Print additional time information.

Details

This function inputs data into existing population object. It can input single matrix or list of matrices.

Value

An object of class `population`. See `create.population` for details.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- read.population - Loads genotype, phenotype, genetic map data files into R environment into a population object.
- create.population - Create object of class population from data already in R environment.
- fake.population - Simulate basic population object for use in examples.

Examples

```
population <- fake.population()
offspring <- population$offspring$phenotypes
founders <- population$founders$phenotypes
founders_groups <- population$founders$groups
maps_genetic <- population$maps$genetic
population <- create.population(offspring,founders,founders_groups)
population <- add.to.population(population,maps_genetic,"maps$genetic")
```

---

assignChrToMarkers *Function that assigns a chromosome label to a genetic marker*

Description

This functions returns an ordering vector of markers for each marker it shows which chromosome the marker belongs to.

Usage

```
assignChrToMarkers(assignment, cross)
```

Arguments

assignment	Chromosome assignment vector created using cross.denovo with reOrder = FALSE
cross	An object of class cross. See read.cross for details.

Details

When using the cross.denovo function with the parameter reOrder = FALSE, its return value will be a chromosome assignment vector. This chromosome assignment vector shows how chromosomes from the created map are assigned to chromosomes from the original map. By using the assignChrToMarkers function the chromosome assignment vector is transformed into a marker ordering vector, which is used by reorganizeMarkersWithin to reorder markers inside the cross object.

Value

Ordering vector, that can be used by reorganizeMarkersWithin function to reorder the cross object.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

reorganizeMarkersWithin - Apply new ordering on the cross object usign ordering vector. cross.saturate - Saturate existing map. cross.denovo - Create de novo genetic map or vector showing how chromosomes should be assigned.

Examples

```
data(yeastCross)
data(yeastPopulation)
assignment <- cross.denovo(yeastPopulation,n.chr=16,verbose=TRUE,map="physical",
  comparisonMethod=sumMajorityCorrelation, use.orderMarkers=FALSE,reOrder=FALSE)
  assignment
  ordering <- assignChrToMarkers(assignment,yeastCross)
```

---

create.population *Create a population object*

Description

Create a new population object from phenotype data already loaded in the R environment

Usage

```
create.population(offspringPhenotypes, founders, foundersGroups, offspringGenotypes,
 mapsGenetic, mapsPhysical, populationType=c("riself", "f2", "bc", "risib"),
 noWarn=FALSE, verbose=FALSE, debugMode=0)
```

Arguments

offspringPhenotypes	A matrix that contains the phenotype data measured on the offspring (required).
founders	A matrix that contains the phenotype data measured on the founders (required).
foundersGroups	When multiple measurement for the founders are present this is used to group the founders. The format is a matrix that contains the phenotype data measured on the (required).
offspringGenotypes	Matrix containing any known offspring genotype data (optional).
mapsGenetic	Matrix containing a known genetic map (optional).
mapsPhysical	Matrix containing a known physical map (optional).
populationType	Type of population the expression data was obtained from: • riself - Recombinant inbred line by selfing.

- f2 - F2 cross.
- bc - Back cross.
- risib - Recombinant inbred line by sibling mating.

noWarn	If TRUE, no warnings will be produced.
verbose	Be verbose.
debugMode	Either use 1 or 2, this will modify the amount of information returned to the user. 1) Print out checks, 2) Print additional time information.

Details

When all required information is provided an object of class `population` is returned.

Value

An object of class `population`. This is a complex object containing all the information needed for the entire pheno2geno analysis. It's structure looks like below (depending on which optional information was supplied):

- $offspring - Section in the object which contains all data related to the offspring:
  - $phenotypes - Offspring phenotype data, stored as a numeric matrix, Rows: phenotypes, Columns: individuals.
  - $genotypes - (Optional) Offspring genotype data:
    - $real - The original data when a known genetic map is provided by the user - numeric matrix, Rows: markers, Columns: individuals.
    - $simulated - Simulated genetic map produced by `generate.biomarkers` from phenotype data - numeric matrix, Rows: markers, Columns: individuals.
- $founders - Section in the object which contains all data related to the founders:
  - $phenotypes - Founders phenotype data, stored as a numeric matrix, Rows: phenotypes, Columns: individuals.
  - $groups - Groups a founder belong to when replicates are available, storad as a vector of 0s and 1s, specifying per column which founder phenotype data belongs to which group.
  - $RP - Results from the t.test or RankProd analysis on the founders phenotype data, by `find.diff.expressed`.
- $maps - Section in the object which contains all data related to maps:
  - $genetic Genetic map, stored as a numeric matrix, Rows: markers, 2 Columns [1] Chromosome, [2] Position on chromosome in cM.
  - $physical Physical map, stored as anumeric matrix, Rows: markers, 2 Columns [1] Chromosome , [2] Position on chromosome in Mbp.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- read.population - Loads genotype, phenotype, genetic map data files into R environment into a population object.
- add.to.population - Adding data to an existing population object.

Examples

```
### simulating data
population <- fake.population()
offspring <- population$offspring$phenotypes
founders <- population$founders$phenotypes
founders_groups <- population$founders$groups
population <- create.population(offspring,founders,founders_groups)
```

---

cross.denovo *Create a de novo genetic map from a population object.*

Description

Create a de novo genetic map from offspring phenotype data stored in a population object

Usage

```
cross.denovo(population, n.chr, map=c("none", "genetic", "physical"),
  comparisonMethod = c(sumMajorityCorrelation, majorityCorrelation, meanCorrelation,
  majorityOfMarkers), assignFunction=c(assignMaximumNoConflicts, assignMaximum),
   reOrder=TRUE, use.orderMarkers=FALSE, verbose=FALSE, debugMode=0, ...)
```

Arguments

population	An object of class population. See create.population for details.
n.chr	Number of chromosomes expected on the map.
map	Which map (from ones stored in population$maps) should be used for assigning chromosomes on the created map. If none is selected - assigning is not performed.
comparisonMethod	Method used to compare chromosomes from the new map to the original ones while assigning: • sumMajorityCorrelation - For each chromosome in cross for every marker checks the marker it is having highest correlation with. Checks on which chromosome this marker is placed in old map. For each of new chromosomes one or more of chromosomes from old map will be represented. Function sums correlations for each pair of those and for every new chromosomes assigns old chromosome with highest cumulative cor.

- majorityCorrelation - For each chromosome in cross for every marker checks the marker it is having highest correlation with. Checks on which chromosome this marker is placed in old map. For each of new chromosomee, old chromosome with most markers with high correlation is assigned.
- meanCorrelation - Assigning chromosome from new map to old ones using sum of the mean correlation between their markers.
- majorityOfMarkers - For each chromosome in the cross object (either created inside the function or provided by user) chromosome from original map, where most markers from new chromosome are is assigned.

`assignFunction` function used to assign chromosomes on the created map, in both cases for every chromosome from the new map, original chromosome with maximal score is assigned, but if one of the original chromosomes is assigned to more then one of new ones:

- assignMaximumNoConflictsadditional step is performed to make sure each of the original chromosomes is used only once
- assignMaximumthose two are being merged

`reOrder` if TRUE, cross object is returned, FALSE - vector showing how chromosomes should be assigned

`use.orderMarkers` should markers on the newly created map be ordered using R/qtl orderMarkers funtion

`verbose` be verbose

`debugMode` 1: Print our checks, 2: print additional time information

`...` parameters passed directly to the `formLinkageGroups` function

Details

cross.denovo function creates new genetic map using genotypes simulated by `generate.biomarkers` function. Then it uses information provided by user to assign number to newly created chromosomes.

Value

When reordering this will produce an object of class `cross`, otherwise (reOrder=FALSE) a chromosomes assignment vector (See `assignChrToMarkers`) is produced which can be used to manual reorder the markers.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- `reorganizeMarkersWithin` - Apply new ordering on the cross object usign ordering vector.
- `assignChrToMarkers` - Create ordering vector from chromosome assignment vector.

- cross.saturate - Saturate existing map.
- reduceChromosomesNumber - Number of routines to reduce number of chromosomes of cross object.
- generate.biomarkers - Creating genotype markers out of gene expression data.

Examples

```
data(yeastPopulation)
cross <- cross.denovo(yeastPopulation,n.chr=16,verbose=TRUE,map="physical",
  comparisonMethod=sumMajorityCorrelation, use.orderMarkers=FALSE)
```

---

cross.saturate *Saturate an existing genetic map.*

Description

Saturating an existing genetic map using markers derived from phenotype data.

Usage

```
cross.saturate(population, cross, map=c("genetic","physical"), placeUsing=c("qtl",
  "correlation"), flagged = c("remove","warn","ignore"), threshold=3, chr, env,
  use.orderMarkers=FALSE, verbose=FALSE, debugMode=0)
```

Arguments

population	An object of class population. See create.population for details.
cross	An object of class cross. See read.cross for details. If not supplied, it will be created using data from the population object
map	Which map should be used for comparison: • genetic - genetic map from cross$maps$genetic. • physical - physical map from cross$maps$physical.
placeUsing	How should the position of the new markers on the saturated map be determinated: • qtl - position the new markers between / next to markers with high LOD score (see threshold). • correlation - position the new markers on the locations with the highest correction to markers on the physical map from cross$maps$physical.
flagged	How to handle the markers influenced by epistatic or environmental interactions: • remove - warn about every marker affected and remove them. • warn - warn about every marker affected but leave them in. • ignore - leave them in.
threshold	Specifies a threshold for the selection of new phenotype markers (see markerPlacementPlot).

chr	When specified the algorithm only saturates a subset of chromosomes. If not specified, all the chromosomes will be saturated.
env	Vector of environmental conditions - for each of the individuals specifies a condition. Ignored if missing.
use.orderMarkers	If true the algorithm (after initial saturation) performs an orderMarkers on the newly created map.
verbose	Be verbose.
debugMode	Either use 1 or 2, this will modify the amount of information returned to the user. 1) Print out checks, 2) Print additional time information.

Details

This function saturates an existing map with markers derived from the phenotype data provided inside either the cross or population object. A correlation matrix between those two sets of markers is made, and new markers are assigned to the 'optimal' location on the map.

Value

An object of class population. See create.population for details.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- reorganizeMarkersWithin - Apply new ordering on the cross object usign ordering vector.
- assignChrToMarkers - Create ordering vector from chromosome assignment vector.
- cross.denovo - Create de novo genetic map or chromosome assignment vector.
- reduceChromosomesNumber - Functions to reduce the number of chromosomes in a cross object.
- markerPlacementPlot - Plot showing how many markers will be selected for map saturation with different thresholds.

Examples

```
data(yeastPopulation)
cross <- cross.saturate(yeastPopulation,map="physical",verbose=TRUE,debugMode=2)
```

`fake.population` *Simulate a population object.*

Description

Simulates a basic population object for use in examples.

Usage

```
fake.population(n.founders = 4, n.offspring = 100, n.markers=100,n.chromosomes=10,
  type = c("riself", "f2", "bc", "risib"), n.mixups=0, verbose=FALSE,...)
```

Arguments

`n.founders`	Number of founders to be simulated.
`n.offspring`	Number of offspring individuals to be simulated.
`n.markers`	Number of markers individuals to be simulated.
`n.chromosomes`	Number of chromosomes individuals to be simulated.
`type`	Type of the cross to be faked: • riself - RILs by selfing. • f2 - f2 cross. • bc - back cross. • risib - RILs by sibling mating.
`n.mixups`	Number of mixups to be faked.
`verbose`	Be verbose.
`...`	To be passed to `sim.cross`.

Details

This function simulates a population object that can be used for further analysis.

Value

An object of class `population`. See `create.population` for details.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

References

Copenhaver, G. P., Housworth, E. A. and Stahl, F. W. (2002) Crossover interference in arabidopsis. *Genetics* **160**, 1631–1639.

Foss, E., Lande, R., Stahl, F. W. and Steinberg, C. M. (1993) Chiasma interference as a function of genetic distance. *Genetics* **133**, 681–691.

Zhao, H., Speed, T. P. and McPeek, M. S. (1995) Statistical analysis of crossover interference using the chi-square model. *Genetics* **139**, 1045–1056.

Broman, K. W. (2005) The genomes of recombinant inbred lines *Genetics* **169**, 1133–1146.

Teuscher, F. and Broman, K. W. (2007) Haplotype probabilities for multiple-strain recombinant inbred lines. *Genetics* **175**, 1267–1274.

See Also

- `read.population` - Load genotype, phenotype, genetic map data files into R environment into a population object.
- `add.to.population` - Add data to existing population object.
- `sim.cross` - Function from R/qtl package used to simulate genotypic data.
- `create.population` - Create object of class population from data already in R environment.

Examples

```
population <- fake.population()
```

`find.diff.expressed` *Finding differentially expressed genes.*

Description

Using Rank Product or student t-test analysis to select differentially expressed genes.

Usage

```
find.diff.expressed(population,use=c("ttest","rankprod"),verbose=FALSE,
debugMode=0,...)
```

Arguments

`population`	An object of class `population`. See `create.population` for details.
`use`	Which method should be used for selecting differentially expressed probes: • ttest - student t-test. • rankprod - Rank Product using RP function from RankProd package. *RankProd package from Bioconductor has to be installed before this option is enabled.*
`verbose`	Be verbose.
`debugMode`	1: Print out checks, 2: print additional time information.
`...`	Additional arguments passed to RP function.

Details

This function finds probes differentially expressed between founders using either student t.test or RankProd (*RankProd package from Bioconductor has to be installed before this option is enabled.*).

Value

Object of class population.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

References

Hong F, Breitling R, McEntee CW, Wittner BS, Nemhauser JL, Chory J.(2006) RankProd: a bioconductor package for detecting differentially expressed genes in meta-analysis. *Bioinformatics*, **15**;22(22):2825-7.

See Also

- RP - Perform rank product method to identify differentially expressed genes.
- read.population - Load genotype, phenotype, genetic map data files into R environment into a population object.
- generate.biomarkers - Creating genotypes from children phenotypes.
- showRPpval- Printing out p-values calculated by the find.diff.expressed function.
- plotRPpval - Plotting p-values calculated by the find.diff.expressed function.

Examples

```
data(yeastPopulation)
yeastPopulation <- find.diff.expressed(yeastPopulation)
```

---

find.mixups *Find sample mix-ups*

Description

Finding possible sample mix-ups in the data.

Usage

```
find.mixups(population,map=c("genetic","physical"),n.qtls=50,threshold=15,verbose=FALSE)
```

Arguments

population	An object of class population. See create.population for details.
map	Which map should be used to determine the ordering / positions of original markers.
n.qtls	Number of qtls that we use for scanning for mix-ups.
threshold	When an individual is not matching the expected genotype more the x % of the time. The individual should be considered as being a mix-up.
verbose	Be verbose.

Details

After scanning for the requested number of QTLs, each individual is checked if their genotype is matching the expected genotype. If an individuals expression value is not in the range of the expected genotype (mean - 3*sd, mean+3*sd), it's receives a penaltie. After which the individuals above the threshold are being printed with a warning about possible mix-up.

Value

An vector with for each individual a percentage that shows how many times an individual didn't match the expected genotype.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- read.population - Load genotype, phenotype, genetic map data files into R environment into a population object.
- cross.denovo - Create de novo genetic map or vector showing how chromosomes should be assigned.
- cross.saturate - Saturate existing map.
- find.diff.expressed - Using Rank Product or student t-test analysis to select differentially expressed genes.

Examples

```
data(yeastPopulation)
  scores <- find.mixups(yeastPopulation,map="physical",n.qtls=10,threshold=5,verbose=FALSE)
  plot(scores[[2]])
```

`generate.biomarkers` *Generate discrete biomarkers from the continuous phenotypes*

Description

Creating genotype markers out of gene expression data.

Usage

```
generate.biomarkers(population, threshold=0.05, overlapInd = 10,
  proportion = c(50,50), margin = 15, pProb=0.8, n.cluster=1, env,
  verbose=FALSE, debugMode=0)
```

Arguments

`population`	An object of class `population`. See `create.population` for details.
`threshold`	If the pvalue for differential expression of this phenotype (see `find.diff.expressed`) is lower that the set threshold, the phenotype is kept in the analysis as being differentially expressed.
`overlapInd`	The number of individuals that are allowed in the overlap (undecided region) when assigning genotype encodings.
`proportion`	The expected proportion of individuals expected to carrying a certain genotype (e.g. c(50,50) in a recombinant inbred line).
`pProb`	Threshold posterior probability used to assign expression values to the genotypes. If not crossed - empty genotype is assigned.
`n.cluster`	Number of cores to be used .
`env`	Vector of environmental conditions - for each of the individuals specifies a condition. Ignored if missing.
`margin`	This specifies how much deviation from the expected proportion is allowed (2 sided).
`verbose`	Be verbose.
`debugMode`	Either use 1 or 2, this will modify the amount of information returned to the user. 1) Print out checks, 2) Print additional time information.

Details

This function, using the results from mixture modeling splits the continuous offspring phenotype data into discrete genotype markers, infering the direction from the founders expression data.

Value

An object of class `cross`. See `read.cross` for details

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- read.population - Load genotype, phenotype, genetic map data files into R environment into a population object.
- cross.denovo - Create de novo genetic map or vector showing how chromosomes should be assigned.
- cross.saturate - Saturate existing map.
- find.diff.expressed - Using Rank Product or student t-test analysis to select differentially expressed genes.

Examples

```
#Example for F2 population
set.seed(102)
population <- fake.population(type="f2")
population <- find.diff.expressed(population)
population <- generate.biomarkers(population,proportion=c(25,50,25),threshold=0.01)
## Not run:
#Example for BC population
set.seed(102)
population <- fake.population(type="bc")
population <- find.diff.expressed(population)
population <- generate.biomarkers(population,proportion=c(25,75),threshold=0.01)

#Example for BC population
set.seed(102)
population <- fake.population(type="riself")
population <- find.diff.expressed(population)
population <- generate.biomarkers(population,proportion=c(50,50),threshold=0.01)

## End(Not run)
```

---

map.functions *Functions to provide some descriptive statistics on genetic maps*

Description

Functions to provide some descriptive statistics on genetic maps

Usage

```
avg_map_distances(m)
map_distances(m)
```

Arguments

m	An object of class cross or map, See read.cross or pull.map for details.

Value

A list with per chromosomes either the average map distance or the total distance

Author(s)

Danny Arends <Danny.Arends@gmail.com>, Konrad Zych <k.zych@rug.nl> Maintainer: Danny Arends <Danny.Arends@gmail.com>

See Also

- read.population - Load genotype, phenotype, genetic map data files into R environment into a population object.
- cross.denovo - Create de novo genetic map or vector showing how chromosomes should be assigned.
- cross.saturate - Saturate existing map.
- find.diff.expressed - Using Rank Product or student t-test analysis to select differentially expressed genes.

Examples

```
data(yeastCross)
  avg_map_distances(yeastCross)
  map_distances(yeastCross)
```

---

markerPlacementPlot *Plot number of markers selected.*

Description

Plot number of markers selected with different thresholds.

Usage

```
markerPlacementPlot(population, placeUsing=c("qtl","correlation"),
  thrRange=c(1,5,1),cross,verbose=FALSE)
```

Arguments

population	An object of class population. See create.population for details.
placeUsing	How position of the new markers on the saturated map should be determinate: • qtl - placed between two markers with highest . • correlation - physical map from cross$maps$physical.
thrRange	Range of the threshold to be checked. Specified in a format(start,stop,step).
cross	An object of class cross. See read.cross for details.
verbose	Be verbose.

Details

This plot is really useful while saturating existing map (using cross.saturate). It helps choose best threshold for marker selection, showing how much markers will be selected with different threshold values.

Value

None.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- cross.saturate - Saturate existing map.
- reorganizeMarkersWithin - Apply new ordering on the cross object usign ordering vector.
- assignChrToMarkers - Create ordering vector from chromosome assignment vector.
- reduceChromosomesNumber - Number of routines to reduce number of chromosomes of cross object.
- generate.biomarkers - Creating genotype markers out of gene expression data.

Examples

```
data(yeastCross)
data(yeastPopulation)
markerPlacementPlot(yeastPopulation,placeUsing="qtl",cross=yeastCross)
markerPlacementPlot(yeastPopulation,placeUsing="correlation",cross=yeastCross)
```

`markersCorPlot` *Plotting correlation between markers on two maps.*

Description

Plotting correlation between two maps together with markers placement (comparison of coverage).

Usage

```
markersCorPlot(cross, population, map=c("genetic","physical"), cmBetween=25,
  comparisonMethod = c(sumMajorityCorrelation,majorityCorrelation,
  meanCorrelation,majorityOfMarkers), chr, show.legend=FALSE, verbose=TRUE)
```

Arguments

`cross`	R/qtl cross type object.
`population`	An object of class `population`.
`map`	Which map (from ones stored in population$maps) should be used fo assigning chromosomes on the created map.
`cmBetween`	Offset between chromosomes specified in cM.
`comparisonMethod`	Method used to compare chromosomes from the new map to the original ones while assigning: • sumMajorityCorrelation - For each chromosome in cross for every marker checks the marker it is having highest correlation with. Checks on which chromosome this marker is placed in old map. For each of new chromosomes one or more of chromosomes from old map will be represented. Function sums correlations for each pair of those and for every new chromosomes assigns old chromosome with highest cumulative cor. • majorityCorrelation - For each chromosome in cross for every marker checks the marker it is having highest correlation with. Checks on which chromosome this marker is placed in old map. For each of new chromosomee, old chromosome with most markers with high correlation is assigned. • meanCorrelation - Assigning chromosome from new map to old ones using sum of the mean correlation between their markers. • majorityOfMarkers - For each chromosome in the cross object (either created inside the function or provided by user) chromosome from original map, where most markers from new chromosome are is assigned.
`chr`	Specifies subset of chromosomes to be shown.
`show.legend`	Shall the legend be shown on the plot.
`verbose`	Be verbose.

Details

Plots markers from moth old and new map as points and in the background - comparison between them done using selected comparison method.

Value

Matrix of comparisons between chromosomes obtained using comparison method.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- plotMapComparison - Plotting routine for comparison of two genetic maps.
- projectOldMarkers - Plotting routine for showing how markers from original map are placed on saturated map.
- cross.saturate - Saturate existing map.
- cross.denovo - Create de novo genetic map or chromosome assignment vector.

Examples

```
data(yeastPopulation)
data(yeastCross)
markersCorPlot(yeastCross,yeastPopulation,map="physical")
```

---

modify number of chromosomes

Change the number of chromosomes in a cross object

Description

Methods to manually modify the number of chromosomes inside an cross object

Usage

```
reduceChromosomesNumber(cross, numberOfChromosomes, verbose=FALSE)
removeChromosomes(cross, chromosomesToBeRmv, verbose=FALSE)
removeTooSmallChromosomes(cross, minNrOfMarkers, verbose=FALSE)
```

Arguments

cross	An object of class cross. See read.cross for details.
numberOfChromosomes	How many chromosomes should stay (remove all but 1:numberOfChromosomes).
chromosomesToBeRmv	NAMES of chromosomes to be removed.
minNrOfMarkers	Specify minimal number of markers chromosome is allowed to have (remove all that have less markers than that).
verbose	Be verbose.

Details

There are three functions in pheno2geno to allow the user to manually reduce number of resulting chromosomes.

reduceChromosomesNumber This functions removes all chromosomes from the cross object excluding chromosome 1 to numberOfChromosomes. It depends on the ordering of chromosomes inside the cross object (which is based on the length of the chromosomes).

removeChromosomes This function removes chromosomes from the cross object by name. Because of this it does not depend on the ordering of the chromosomes inside the cross object.

removeTooSmallChromosomes This function is used to clean a cross object after using `formLinkageGroups`. FormLinkageGroups can introduce small chromosomes as artifacts. These linkage groups consist of only a few markers with poor quality and should be removed from the cross object.

Value

An object of class `cross`. See `read.cross` for details.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

`reorganizeMarkersWithin` - Apply new ordering on the cross object usign ordering vector. `assignChrToMarkers` - Create ordering vector from chromosome assignment vector. `cross.saturate` - Saturate existing map. `cross.denovo` - Create de novo genetic map.

Examples

```
data(yeastCross)
plot.rf(yeastCross, main="riself generate.biomarkers example")
cross_ <- reduceChromosomesNumber(yeastCross,5,verb=TRUE)
plot.rf(cross_, main="Leaving only 5 chromosomes")
cross_ <- removeChromosomes(yeastCross,1,verb=TRUE)
plot.rf(cross_, main="Removing chromosome 1")
cross_ <- removeTooSmallChromosomes(yeastCross,5,verb=TRUE)
plot.rf(cross_, main="Leaving only chromosomes with more than 5 markers")
```

`plotChildrenExpression`

Plotting routine for children expression data.

Description

Plots offspring gene expression data in comparison with founders data.

Usage

```
plotChildrenExpression(population, markers=1:100)
```

Arguments

population	An object of class population. See read.population for details.
markers	Numbers of markers to be plotted.

Details

Plots offspring expression data (boxplot) max value of parental expression (red triangle) min (blue triangle) and mean(line) for selected markers.

Value

None.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- plotParentalExpression - Plotting routine for parental gene expression data.
- plotMarkerDistribution - Plotting gene expression data for a single marker.

Examples

```
data(yeastPopulation)
### plotting only 10 markers for clearer image
plotChildrenExpression(yeastPopulation,10:20)
```

---

plotMapComparison *Plotting routine for comparison of two genetic maps.*

Description

Plotting routine for comparison of two genetic maps.

Usage

```
plotMapComparison(cross, population, map=c("genetic","physical"), chr)
```

Arguments

cross	An object of class cross. See read.cross for details.
population	An object of class population. See create.population for details.
map	Which map (from ones stored in population$maps) should be used fo assigning chromosomes on the created map.
chr	Specifies subset of chromosomes to be shown.

Value

None.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- markersCorPlot - Plotting correlation between two maps together with markers placement (comparison of coverage).
- projectOldMarkers - Plotting routine for showing how markers from original map are placed on saturated map.
- cross.saturate - Saturate existing map.
- cross.denovo - Create de novo genetic map or chromosome assignment vector.

Examples

```
data(yeastPopulation)
data(yeastCross)
plotMapComparison(yeastCross,yeastPopulation,map="physical")
```

---

plotMarkerDistribution

plotMarkerDistribution

Description

Plotting distribution of gene expression values of a single marker.

Usage

```
plotMarkerDistribution(population,marker,nrDistributions,logarithmic=FALSE)
```

Arguments

population	An object of class population. See create.population for details.
marker	Number or name of the marker to be printed.
nrDistributions	Number of normal distributions to be fitted.
logarithmic	TRUE - log(data) is used instead of raw data.

Details

Plotting histogram out of gene expression data for a single marker and fitting specified number of normal distribution curves, using EM algorithm.

Value

None.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- plotParentalExpression - Plotting routine for parental gene expression data.
- plotChildrenExpression - Plotting routine for children gene expression data.

Examples

```
data(yeastPopulation)
plotMarkerDistribution(yeastPopulation,2,2)
```

---

plotParentalExpression

Plotting routine for parental expression data.

Description

Plots parental gene expression data.

Usage

```
plotParentalExpression(population, markers=1:100, groupLabels=c(0,0,1,1))
```

Arguments

population	An object of class population. See create.population for details.
markers	Numbers of markers to be plotted.
groupLabels	Specify which column of parental data belongs to group 0 and which to group 1.

Details

Plots parental gene expression data in two colors (two parental groups) and mean of values for each marker.

Value

None.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- plotChildrenExpression - Plotting routine for children gene expression data.
- plotMarkerDistribution - Plotting gene expression data for a single marker.

Examples

```
data(yeastPopulation)
### plotting
plotParentalExpression(yeastPopulation)
```

---

power.plot *Comparison of power of qtl detection.*

Description

Plots maximal values of QTL peak measured on the same phenotypes in two crosses.

Usage

```
power.plot(cross1,cross2,scores,qtlThr=5,nPheno=500,verbose=FALSE,...)
```

Arguments

cross1	An object of class cross. See read.cross for details.
cross2	An object of class cross. See read.cross for details.
scores	An object of class scores (result of running of this function). This allows for not recalculating QTL scores everytime user wants to plot them.
qtlThr	Threshold for assessing the significance of the QTL peak.
nPheno	Nr of phenotypes that will be scanned for QTLs. Phenotypes are selected randomly.
verbose	Be verbose.
...	Arguments passed to scanone function (see scanone).

Details

Plots maximal values of QTL peak measured on the same phenotypes in two crosses. This give a good comparison of power to detect the QTLs between crosses, if the number of phenotypes scanned is large enough.

Value

An object of class scores containing all the QTL scores calculated during the run of this function. This can be plugged back into the function to avoid unnecessary recalculation of the scores.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- plotMapComparison - Plotting routine for comparison of two genetic maps.
- projectOldMarkers - Plotting routine for showing how markers from original map are placed on saturated map.
- cross.saturate - Saturate existing map.
- cross.denovo - Create de novo genetic map or chromosome assignment vector.

Examples

```
data(yeastCross)
  power.plot(yeastCross,yeastCross,nPheno=50)
```

`projectOldMarkers` *Plotting routine which shows where markers from original map are located on saturated map.*

Description

Plotting routine which shows where markers from original map are located on saturated map.

Usage

```
projectOldMarkers(cross,population,map=c("genetic","physical"),
  label=c("positions","names","no"),...)
```

Arguments

`cross`	An object of class `cross`. See `read.cross` for details.
`population`	An object of class `population`. See `create.population` for details.
`map`	Which map (from the ones stored in the population$maps) should be used to assigning chromosomes on the created map
`label`	Should the old markers be labeled in the plot (options: position, name or off).
`...`	Parameters passed to `plot.qtl`.

Value

None.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- `plotMapComparison` - Plotting routine for comparison of two genetic maps.
- `markersCorPlot` - Plotting correlation between two maps together with markers placement (comparison of coverage).

Examples

```
data(yeastPopulation)
data(yeastCross)
projectOldMarkers(yeastCross,yeastPopulation,map="physical")
```

pull.biomarkers *Extract the detected biomarkers from a population object.*

Description

Extract the detected biomarkers from a population object, or select biomarkers that best match a certain pattern.

Usage

```
pull.biomarkers(population, pattern, verbose=FALSE)
```

Arguments

population	An object of class population. See create.population for details.
pattern	Vector containg pattern to be matched in markers.
verbose	Be verbose.

Details

After running generate.biomarkers function, biomarkers are stored inside population class object. Use the pull.biomarkers function to extract them from the popuklation object into a matrix. This will return a matrix will all the markers or when pattern is specified a vector with biomarkers best matching the pattern.

Value

Matrix of all markers / vector with markers best matching the specified pattern.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- generate.biomarkers - Create genotype markers out of gene expression data.
- read.population - Load genotype, phenotype, genetic map data files into R environment into a population object.
- cross.denovo - Create de novo genetic map or vector showing how chromosomes should be assigned.
- cross.saturate - Saturate existing map.
- find.diff.expressed - Using Rank Product or student t-test analysis to select differentially expressed genes.

Examples

```
data(yeastPopulation)
markers <- pull.biomarkers(yeastPopulation,verbose=TRUE)
bestMarker <- pull.biomarkers(yeastPopulation,round(runif(109)),verbose=TRUE)
```

`qtl.comparison.plot` *Comparison of qtl profiles.*

Description

Plots comparison between the qtl profiles of two cross objects.

Usage

```
qtl.comparison.plot(cross1, cross2, chr, ...)
```

Arguments

`cross1`	An object of class `cross`. See `read.cross` for details.
`cross2`	An object of class `cross`. See `read.cross` for details.
`chr`	Specifies the chromosome to be shown (only one chromosome can be plotted at a time.
`...`	Arguments passed to scanone function (see `scanone`).

Details

Plots markers from moth old and new map as points and in the background - comparison between them done using selected comparison method.

Value

Matrix of comparisons between chromosomes obtained using comparison method.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- `plotMapComparison` - Plotting routine for comparison of two genetic maps.
- `projectOldMarkers` - Plotting routine for showing how markers from original map are placed on saturated map.
- `cross.saturate` - Saturate existing map.
- `cross.denovo` - Create de novo genetic map or chromosome assignment vector.

Examples

```
data(yeastCross)
qtl.comparison.plot(yeastCross,yeastCross)
```

`read.population` *Loading genotype and phenotype data*

Description

Loads genotype, phenotype, genetic map data files into R environment into a population object.

Usage

```
read.population(offspring = "offspring", founders = "founders", map = "map",
foundersGroups, populationType = c("riself", "f2", "bc", "risib"),
readMode = c("normal","HT"), threshold=0.05, verbose = FALSE, debugMode = 0,
n.cluster=1, ...)
```

Arguments

`offspring`	Core used to specify names of children phenotypic ("core_phenotypes.txt") genotypic ("core_genotypes.txt") and annotations ("core_annotations.txt") files.
`founders`	Core used to specify names of parental phenotypic ("core_phenotypes.txt") file.
`map`	Core used to specify names of genetic ("map_genetic.txt") and physical ("map_physical.txt") map files.
`foundersGroups`	Specify groups of individuals in founders data, see description below and RP for more details
`populationType`	Type of the population data was obtained from: • riself - RILs by selfing. • f2 - f2 cross. • bc - back cross. • risib - RILs by sibling mating.
`readMode`	HT, or High-Throughput mode should be used when the very large dataset is processed (at least 10000 probes). Then files are read in chunks intead of at once. To avoid R memory limits, only probes showing differential expression between parent are selected. Size of the chunk and threshold for assesing significance can be specified (see description of ... parameter).
`threshold`	- threshold for assesing probes that are differentially expressed between parents. 0.05 by default.
`verbose`	Be verbose
`debugMode`	1: Print out checks, 2: print additional time information
`n.cluster`	number of cores used for calcuations
`...`	Parameters passed to high-throughtput function: • transformations - how should the data be transformed (see `transformation`) • sliceSize - number of lines to be read at once byt HT function. 5000 by default.

Details

Function is working on tab delimited files. Phenotype files, both for founders and offspring, should have header, containing column names (so names of individuals). All the other rows should start with rowname (unique). Rownames and colnames are only values allowed to be not numeric. After file is read into R, check is performed and rows and columns containing values that are not numeric and not convertable to numeric, will be removed from dataset. Rownames should match between founders and offspring. After loading founders file in, all non-matching rows are removed. Example of phenotype file structure:

	"individual1"	"individual2"	"individual3"	"individual4"	"individual5"
"marker"	8.84494695336781	9.06939381429179	9.06939381429179	7.72431126650435	6.04480152688572
"marker2"	9.06939381429179	7.85859536346299	8.84494695336781	6.04480152688572	7.72431126650435
"marker3"	6.04480152688572	6.04480152688572	7.85859536346299	7.72431126650435	7.85859536346299
"marker4"	6.04480152688572	7.85859536346299	6.04480152688572	8.84494695336781	7.85859536346299
"marker5"	7.72431126650435	7.72431126650435	17.85859536346299	7.85859536346299	7.85859536346299

Genotype file should have basically the same structure as the phenotype file. The genotypes codes are exactly the same as in r/qtl - for F2 populations: AA - 1, AB - 2, BB - 3, not BB - 4, not AA - 5, missing - NA and for BC and RILs: AA - 1, BB - 2, missing - NA (see `read.cross` for details.) Example of genotype file structure:

	"individual1"	"individual2"	"individual3"	"individual4"	"individual5"
"marker"	1	1	2	1	2
"marker2"	NA	1	2	1	2
"marker3"	1	1	1	1	2
"marker4"	1	NA	1	1	2
"marker5"	NA	1	1	1	2

Map files should have really simple structure, always three columns, no header. First column contains rownames, second - chromosome number and third - position on chromosome (in cM for genetic or Mbp for physical map). Secodn and third column can contain only numbers (any NA, Inf, etc, will cause dropping of file). Rownames should match either ones from genotype file or ones from phenotype file, depending which one you want to use map with (see generate.biomarkers for more information). Example of map file structure:

"marker"	1	0
"marker2"	1	1.2
"marker3"	1	1.2
"marker4"	1	2
"marker5"	1	3

You have also to specify groups ion founders file, so which columns come from which parent. Let's imagine, you have measured both parents in triplo and data for first parent is in columns 1,3 and 5, for second parent - columns 2,4,6. Founders groups should be c(0,1,0,1,0,1) then. Always use only

0 and 1 to specify groups.

Value

An object of class population.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- add.to.population - Adding data to existing population object.
- create.population - Create new object of class population.

Examples

```
## Not run:
### simplest call possible
population <- read.population(founders_groups=c(0,0,0,1,1,1))
### more informative one
population <- read.population(founders_groups=c(0,0,0,1,1,1),verbose=TRUE,debugMode=1)
### imagine you prefer parents and children instead of founders and offspring:
population <- read.population(offspring="children",founders="parents",
  founders_groups=c(0,0,0,1,1,1),verbose=TRUE,debugMode=1)
### etc.. when you load it, you may want to inspect it:
population$founders$phenotypes[1:10,]

## End(Not run)
```

---

reorganizeMarkersWithin

Reorganize markers within cross object.

Description

Reorder markers within cross object using supplied marker ordering vector.

Usage

```
reorganizeMarkersWithin(cross, ordering)
```

Arguments

cross	An object of class cross. See read.cross for details.
ordering	Ordering vector specifying for every marker in the cross object (by name) which new chromosome this marker will be moved to.

Details

Functions reorders an object of class cross using the supplied marker ordering vector. This vector contains for each marker the chromosome number that this marker will be moved to.

Value

An object of class cross. See read.cross for details.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- cross.denovo - Creating a de novo genetic map or a chromosome assignment vector.
- cross.saturate - Saturate an existing genetic map by using phenotype markers.
- assignChrToMarkers - Create ordering vector from chromosome assignment vector.

Examples

```
data(yeastCross)
data(yeastPopulation)
assignment <- cross.denovo(yeastPopulation,n.chr=16,verbose=TRUE,map="physical",
comparisonMethod=sumMajorityCorrelation, use.orderMarkers=FALSE,reOrder=FALSE)
assignment #boring,but expected
ordering <- assignChrToMarkers(assignment,yeastCross)
yeastCross <- reorganizeMarkersWithin(yeastCross, ordering)
```

RPpval *Visualize the outcome of a Rank product analysis*

Description

Visualize the outcome of a Rank product analysis, this function will print/plot p-values calculated by the find.diff.expressed function.

Usage

```
showRPpval(population,markers=1:10)
plotRPpval(population,thresholdRange=c(0.01,0.1,0.01))
```

Arguments

population	An object of class population. See create.population for details.
markers	Numbers of markers to be printed
thresholdRange	Specifies in which range threshold will be checked (start, stop, step).

Details

Those are two helper functions of `find.diff.expressed`. showRPpval is printing to the screen p-values for specified markers, while plotRPpval is showing how many markers will be selected using different thresholds.

Value

An object of class population, (see `create.population` for more details) with object of class RP saved into population$founders$RP.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- `RP` - Perform rank product method to identify differentially expressed genes.
- `find.diff.expressed` - Select differentially expressed genes using Rank Product or student t-test analysis.
- `generate.biomarkers` - Creating genotypes from children phenotypes.
- `showRPpval`- Printing out p-values calculated by the find.diff.expressed function.
- `plotRPpval` - Plotting p-values calculated by the find.diff.expressed function.

Examples

```
data(yeastPopulation)
showRPpval(yeastPopulation)
plotRPpval(yeastPopulation)
```

---

save.gff *Saving gff files.*

Description

Saving gff files.

Usage

```
save.gff(cross, map.physical, ind, gffFileCore="population", verbose=FALSE)
```

Arguments

cross	An object of class cross. See read.cross for details. If not supplied, it will be created using data from the population object
map.physical	Map with physical locations of the markers. A matrix with three columns - chromosome/ start position/ end position. Just like physical map in population object, see: read.population
ind	Which individuals should be saved. Numeric vector. If missing - all individuals will be selected.
gffFileCore	Name of the gff files core. For each of the individuals a separated file is created with a name: "core" +"ind_x.gff".
verbose	Be verbose.

Details

This function saves gff files, that can be visualised using most of the genome viewers. The files contain physical location of markers and recombination breakpoints. Therefore, physical map should be stored in an object of class population. Redundant markers are the markers having the same loaction on the genomic map, but different on the physical map. These may be produced e.g. by cross.saturate function (markers that have QTL exactly on the position where an original marker is located). Also, as a result of smoothing genotypping errors some markers may be put on the same position on the genetic map.

Value

None.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- reorganizeMarkersWithin - Apply new ordering on the cross object usign ordering vector.
- assignChrToMarkers - Create ordering vector from chromosome assignment vector.
- cross.denovo - Create de novo genetic map or chromosome assignment vector.
- reduceChromosomesNumber - Functions to reduce the number of chromosomes in a cross object.
- markerPlacementPlot - Plot showing how many markers will be selected for map saturation with different thresholds.

Examples

```
data(yeastPopulation)
cross <- cross.saturate(yeastPopulation,map="physical",verbose=TRUE,debugMode=2)
```

scan.qtls *Scan qtls*

Description

Scanning population data for qtls for use in cross.saturate function.

Usage

```
scan.qtls(population,map=c("genetic","physical"), env, epistasis = c("scan","ignore"),
 step=0.1,verbose=FALSE)
```

Arguments

population	An object of class population. See create.population for details.
map	Which map (from ones stored in population$maps) should be used fo assigning chromosomes on the created map
env	Vector of environmental conditions - for each of the individuals specifies a condition. Ignored if missing.
epistasis	Should the two markers epistasis be scanned for. It is a heavy procedure!
step	Maximum distance (in cM) between positions at which the genotype probabilities are calculated, though for step = 0, probabilities are calculated only at the marker locations. See calc.genoprob for more information .
verbose	Be verbose.

Details

This function takes care about qtl scan that is used by cross.saturate function. It was made separated function, since process itself takes a long time and before running cross.saturate function one should run markerPlacementPlot to assess the optimal threshold.

Value

An object of class population. See create.population for details.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- read.population - Load genotype, phenotype, genetic map data files into R environment into a population object.
- cross.denovo - Create de novo genetic map or vector showing how chromosomes should be assigned.

- cross.saturate - Saturate existing map.
- find.diff.expressed - Using Rank Product or student t-test analysis to select differentially expressed genes.

Examples

```
data(yeastPopulation)
yeastPopulation <- scan.qtls(yeastPopulation,verbose=TRUE,map="physical",step=0)
```

---

set.geno.from.cross *Pull genotype from an object of class cross.*

Description

Pulling genotypes with a map from cross and putting into population object.

Usage

```
set.geno.from.cross(cross,population,map=c("genetic","physical"))
```

Arguments

cross	An object of class cross. See read.cross for details. If not supported, it will be created using data stored in population
population	An object of class population. See create.population for details.
map	In which map ofd an population object shall map pulled from cross be stored: • genetic - genetic map - population$offspring$maps$genetic. • physical - physical map - population$offspring$maps$physical.

Details

This function pull genotypes with a map from the cross object and puts them into provided population object. This is useful if the same cross is saturated multiple times.

Value

An object of class population. See create.population for details.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- reorganizeMarkersWithin - Apply new ordering on the cross object usign ordering vector.
- assignChrToMarkers - Create ordering vector from chromosome assignment vector.
- cross.denovo - Create de novo genetic map or chromosome assignment vector.
- reduceChromosomesNumber - Number of routines to reduce number of chromosomes of cross object.

Examples

```
data(yeastPopulation)
data(yeastCross)
yeastPopulation <- set.geno.from.cross(yeastCross,yeastPopulation)
```

---

transformation *Basic functions to do transformation / normalization of phenotypes.*

Description

Basic functions to do transformation / normalization of phenotypes.

Usage

```
transformation(x, transformations=c("nothing","log","sqrt","reciprocal","probit",
  "logit"), ..., verbose=TRUE)
```

Arguments

x	data matrix with measurements, Rows: Traits/Phenotypes columns: Individuals.
transformations	which function should be used to transform the data: • nothing - no data transformation performed. • log - log(data) • sqrt - sqrt(data) • reciprocal - 1/(data) • probit - probit transformation • logit - logit transformation
...	Passed to the underlying test function.
verbose	Be verbose.

Value

List with matrices.

Author(s)

Danny Arends <Danny.Arends@gmail.com> Maintainer: Danny Arends <Danny.Arends@gmail.com>

See Also

- cross.saturate - Saturate existing map.
- cross.denovo - Create de novo genetic map or chromosome assignment vector.

Examples

```
data <- matrix(runif(1000),10,100)
  resA <- transformation(data, c("log","logit"))
  resB <- transformation(data, c("reciprocal","probit"))
```

---

write.population *Writes a population object to file.*

Description

Writes a population object to file, for easy loading of intermediate data later.

Usage

```
write.population(population, offspring = "offspring", founders = "founders",
  map = "map", verbose = FALSE, debugMode = 0)
```

Arguments

population	An object of class population. See create.population for details.
offspring	Core used to specify names of children phenotypic ("core_phenotypes.txt") genotypic ("core_genotypes.txt") and annotations ("core_annotations.txt") files.
founders	Core used to specify names of parental phenotypic ("core_phenotypes.txt") file.
map	Core used to specify names of genetic ("map_genetic.txt") and physical ("map_physical.txt") map files.
verbose	Be verbose.
debugMode	1: Print out checks, 2: print additional time information

Details

This function writes an object of class population into a file.

Value

None.

Author(s)

Konrad Zych <k.zych@rug.nl>, Danny Arends <Danny.Arends@gmail.com> Maintainer: Konrad Zych <k.zych@rug.nl>

See Also

- add.to.population - Adding data to existing population object.
- create.population - Create new object of class population.
- read.population - Create new object of class population.

Examples

```
## Not run:
  population <- fake.population()
  write.population(population,verbose=TRUE)

## End(Not run)
```

---

yeastCross *Test cross object*

Description

R/qtl cross object made out of yeast data (see references) using pheno2geno package.

Usage

```
data(yeastCross)
```

Format

An object of class cross. See read.cross for details.

Author(s)

Smith EN, Kruglyak L

References

Smith EN, Kruglyak L. Gene-environment interaction in yeast gene expression. PLoS Biol 2008 Apr 15;6(4):e83

yeastPopulation *Test population object*

Description

pheno2geno population object made out of yeast data (see references).

Usage

```
data(yeastPopulation)
```

Format

An object of class population. See create.population for details.

Author(s)

Smith EN, Kruglyak L

References

Smith EN, Kruglyak L. Gene-environment interaction in yeast gene expression. PLoS Biol 2008 Apr 15;6(4):e83

Index