Package 'MRaQTL'

April 26, 2023

Type Package

Title Expression regulator activity inference with master regulator and activity QTL analyses

Version 1.0.0

Author Jason Hoskins, ORCID:0000-0001-6944-1996

Maintainer Jason Hoskins <jason.hoskins@gmail.com>

Description This package includes all functions required to apply the master regulator (MR) analysis and activity QTL analyses as first demonstrated in Hoskins et al., 2021 (https://doi.org/10.1371/journal.pcbi.1009563) and then developed into a user-friendly pipeline for STAR Protocols (currently under review).
The pipeline is divided into 4 general steps.
1. Tissue-specific gene co-expression network inference with ARACNe (not implemented in R).
2. Expression regulators activity inferences and data preparation (implemented in this package).
3. Phenotypic master regulator (MR) inference (implemented in this package).
4. Expression and activity QTL (eQTL and aQTL) analyses (implemented in this package).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

biocViews

Imports aracne.networks,
Biobase,
caret,
doParallel,
foreach,
future,
graphics,
grDevices,
MatrixEQTL,
MatrixGenerics,
parallel,
preprocessCore,
randomForest,
rlang (>= 0.4.11),
RNOmni,
stats,
utils,
viper

RoxygenNote 7.2.3

Suggests roxygen2

URL https://github.com/hoskinsjw/MRaQTL

BugReports https://github.com/hoskinsjw/MRaQTL/issues

R topics documented:

`corDensPlot` *corDensPlot*

Description

This function generates a png image file of a density plot of regulators' activity-to-expression Pearson correlations.

Usage

```
corDensPlot(expression, activity, file_prefix)
```

Arguments

`expression`	matrix or data.frame of expression values
`activity`	matrix or data.frame of inferred activities
`file_prefix`	string for the file name prefix for the output png file

Value

Writes the density plot for all regulators' matched activities and expression values Pearson correlations to a png image file.

Examples

```
## Not run:
corDensPlot(reg_expression,reg_activity,"Regulators")
# Generates the file "Regulators_activity_expression_correlation.png" in
# your working directory.

## End(Not run)
```

`finalRF` *finalRF*

Description

This function carries out the final master regulator (MR) analysis using the workspace image file from the random forest cross-validation analysis and compares phenotype prediction based on the final MR random forest model to the actual phenotypes in both the training and test sets.

Usage

```
finalRF(rfcv_workspace, exp_file, phenotype, prefix, mr_count = 100)
```

Arguments

`rfcv_workspace`	string indicating the path and file name of the .RData workspace image file from the random forest cross-validation analysis generated by the rfCrossVal function.
`exp_file`	string indicating the path and file name of the normalized expression data file.
`phenotype`	string indicating the column name labeling the phenotype of interest.
`prefix`	string indicating a prefix to be added to all output file names.
`mr_count`	numeric indicating how many representative master regulators should be inferred, which is suggested by the random forest cross-validation plot. Defaults to 100.

Value

Writes seven output files:

1. A log file recording the standard outputs generated by the function.
2. A pdf file for the importance plot of the final MR random forest model that indicates the relative importance of each inferred MR.
3. A text file with the summary statistics for phenotype prediction accuracy based on the final MR random forest model in the training and test sets. For binary phenotypes, these results are from caret::confusionMatrix, while for continuous phenotypes they are from stats:lm linear models.
4. A text file listing the inferred representative phenotypic MRs.
5. A tab-delimited text file containing the input normalized expression data filtered to include only the representative phenotypic MRs.
6. A tab-delimited text file containing the activity data (which was already present in the input workspace file) filtered to include only the representative phenotypic MRs.

7. An .RData workspace image file containing all environment variables from this analysis. This may be useful if one wants to validate the final MR random forest model in an independent dataset by again comparing actual phenotype values to those predicted with the MR random forest model.

Examples

```
## Not run:
finalRF("Toy_data_rfcv_workspace.RData","Sim_pheno","Toy_data")

## End(Not run)
```

---

grabMRs *grabMRs*

Description

This function will grab the expression and activity data for anyn list of genes, though it is implemented here for the purpose of grabbing inferred phenotype MRs.

Usage

```
grabMRs(exp_data, act_data, mr_list, prefix)
```

Arguments

`exp_data`	string, data.frame or matrix. If a string, it must indicate the path and file name for the expression data file. Otherwise, the given, pre-existing data.frame or matrix containing the expression data will be used.
`act_data`	string, data.frame or matrix. If a string, it must indicate the path and file name for the activity data file. Otherwise, the given, pre-existing data.frame or matrix containing the activity data will be used.
`mr_list`	string or vector. If a string, it must indicate the path and file name for the gene (MR) name list. Otherwise, the given, pre-existing vector of gene (MR) names will be used.
`prefix`	string indicating a prefix to be added to all output file names.

Value

Writes to text file the expression and activity data for the given list of genes/MRs.

Examples

```
## Not run:
# Run with objects already in the workspace.
grabMRs(gene_exp,gene_act,mrs,"Toy_data")

# Run with names of files to be read.
grabMRs("./Toy_data_expressed_genes_QNorm_INT_expression.txt",
"Toy_data_regulators_activities.txt",
"Toy_data_representative_Sim_pheno_MRs_list.txt",
"Toy_data")

## End(Not run)
```

`importancePlot` *importancePlot*

Description

This function writes a pdf file of the importance plot for the final random forest model with the representative phenotypic master regulators.

Usage

```
importancePlot(rndFor, phenotype, mr_count, prefix)
```

Arguments

`rndFor`	random forest model object generated by the randomForest function from the randomForest package.
`phenotype`	string indicating the column name of the phenotype of interest.
`mr_count`	numeric indicating how many representative master regulators are being inferred by this analysis.
`prefix`	string indicating a prefix to be added to the output file name.

Value

Writes the importance plot from the final random forest model with the representative phenotypic master regulators to a pdf file.

Examples

```
## Not run:
importancePlot(final_rndFor,"Sim_pheno",100,"Toy_data")

## End(Not run)
```

`invNormTrans` *invNormTrans*

Description

This function applies an inverse normal transformation (INT) on the given gene metrics using the RankNorm() function from the RNOmni package, but is more rapid due to the use of multi-threading when possible.

Usage

```
invNormTrans(genes)
```

Arguments

`genes`	matrix or data.frame containing gene metrics with genes in rows and samples in columns. Gene and sample identifiers should be in rownames and colnames, respectively.

Value

Returns a data.frame in the same format as the input gene data, but with inverse normal transformed values.

Examples

```
## Not run:
int_genes <- invNormTrans(genes)

## End(Not run)
```

---

matrixQTL *matrixQTL*

Description

This is the wrapper function for the eQTL/aQTL analyses using the MatrixEQTL package, which was authored and is maintained by Andrey A Shabalin and is under an LGPL-3 license. For MatrixEQTL troubleshooting and bug reports, visit https://github.com/andreyshabalin/MatrixEQTL.

Usage

```
matrixQTL(
  snp_dose,
  snp_map,
  gene_data,
  gene_map,
  covars = NULL,
  prefix,
  cisP = 1,
  transP = 0.05
)
```

Arguments

`snp_dose`	string indicating the path and file name for the SNP dosage file.
`snp_map`	string indicating the path and file name for the SNP map file.
`gene_data`	string indicating the path and file name for the gene exression/activities file.
`gene_map`	string indicating the path and file name for the gene map file.
`covars`	string indicating the path and file name for the covariables file. If you do not wish to include covariables in the QTL analysis (not typically recommended), do not assign anything to this argument. Defaults to NULL.
`prefix`	string indicating a prefix to be added to all output file names.
`cisP`	numeric indicating the P-value threshold for cis-QTLs to be written to the output file. Defaults to 1 (all cis-QTL results are output).
`transP`	numeric indicating the P-value threshold for trans-QTLs to be written to the output file. Defaults to 0.05 (only nominally significant QTLs are output).

Value

Writes four output files:

1. A log file recording the standard outputs generated by the function.
2. A pdf file for the QQ-plot of all tested cis and trans-QTLs.
3. A text file with the cis-QTL summary statistics that includes only those that passed the cis P threshold set by the cisP argument.
4. A text file with the trans-QTL summary statistics that includes only those that passed the trans P threshold set by the transP argument.

Examples

```
## Not run:
matrixQTL(
"GEUVADIS_filtered_samples_WHR_significant_bi-allelic_SNPs.dosage",
"GEUVADIS_WHR_significant_bi-allelic_SNPs_locations_in_GRCh37.map",
"Toy_data_regulators_activities.txt",
"GEUVADIS_expressed_genes_locations_in_GRCh37.map",
"GEUVADIS_filtered_samples_select_covars.txt",
prefix="Toy_data")

## End(Not run)
```

`phenoAssoc` *phenoAssoc*

Description

This function determines the association between the given phenotype and all input training set activities and then returns a list of the top Bonferroni-significant regulators (within a given min/max).

Usage

```
phenoAssoc(
  activities,
  phenotype,
  binary = FALSE,
  min_regs = 100,
  max_regs = 500,
  Bonf_thresh = 0.05
)
```

Arguments

`activities`	data.frame containing the training set activities
`phenotype`	data.frame containing the training set phenotype data
`binary`	logical indicating whether the phenotype is binary. If FALSE, the phenotype is assumed to be continuous. Categorical data with two levels will be treated as binary, but more than two levels will not work. Defaults to FALSE.

`min_regs`	numeric indicating the minimum number of best associated regulators to use in the cross-validation random forest analysis. Defaults to 100.
`max_regs`	numeric indicating the maximum number of best associated regulators to use in the cross-validation random forest analysis. Defaults to 500.
`Bonf_thresh`	numeric indicating the Bonferroni threshold for significance that will determine how many of the best associated regulators to use in the cross-validation random forest analysis as long as the number lies within the given min/max. Defaults to 0.05.

Value

Returns a vector of names for the best associated regulators that will be used in the cross-validation random forest analysis

Examples

```
## Not run:
phenoAssoc(train_act,train_pheno,prefix="Toy_data")

## End(Not run)
```

`prepActExp` *prepActExp*

Description

This is the wrapper function that infers regulator activities with the VIPER package, upper quantile normalizes and inverse normal transforms the expression data, and ensures the covariables, activity and expression data are properly formatted for downstream master regulator and eQTL/aQTL analyses.

Usage

```
prepActExp(gene_exp, covariables, network, prefix)
```

Arguments

`gene_exp`	string for the file name of the expression data file (should be appropriately normalized and pre-filtered for expression).
`covariables`	string for the file name of the covariables data file.
`network`	string for the file name of an ARACNe network file. IMPORTANT: this file cannot have a header!!!
`prefix`	string indicating a prefix to be added to all output file names.

Value

Writes six output files:

1. A log file recording the standard outputs generated by the function.
2. A tab-delimited file for the interactome/regulon generated by the aracne2regulon() function from the input ARACNe network and expression data.
3. A png image file of a density plot for all regulators' matched activities and expression values Pearson correlations.
4. A tab-delimited file containing the quantile normalized and inverse normal transformed expression values for all input genes.
5. A tab-delimited file containing the quantile normalized and inverse normal transformed expression values for only expression regulator genes for which activities were successfully inferred.
6. A tab-delimited file containing the activities inferred for expression regulator genes by VIPER.

Examples

```
## Not run:
prepActExp(
"GEUVADIS_filtered_samples_expressed_genes_logRPKMs_with_symbols.txt",
"GEUVADIS_filtered_samples_select_covars.txt",
"GEUVADIS_100boots_ARACNe_network_no_header.txt",
"Toy_data")

## End(Not run)
```

`rfCrossVal` *rfCrossVal*

Description

This is the main function for the random forest cross-validation analysis wherein top phenotype-associated regulators are used in random forest models over a range of feature counts to predict the phenotype of interest. By inspecting the output plot, one may gain a reasonable estimation for the number of features (in this case regulators) required to minimize prediction error.

Usage

```
rfCrossVal(
  act_file,
  pheno_file,
  phenotype,
  prefix,
  seed = 123,
  train_pct = 0.7,
  min_regs = 100,
  max_regs = 500,
  Bonf_thresh = 0.05,
  replicates = 12
)
```

Arguments

act_file	string indicating the path and file name for the regulator activities previously inferred by VIPER using the prepActExp function in this package.
pheno_file	string indicating the path and file name for the phenotype data with the same sample in the same order as the regulator activities file.
phenotype	string indicating the column name for the phenotype of interest in the phenotype file.
prefix	string indicating a prefix to be added to all output file names.
seed	numeric indicating the RNG seed. Defaults to 123
train_pct	numeric indicating the proportion of samples to use in the training set, while the remaining samples will go to the test set. Defaults to 0.7.
min_regs	numeric indicating the minimum number of regulators to use in the random forest cross-validation testing for feature counts. This number of regulators will be used in rfcv if the number of regulators that associate with the phenotype at a Bonferroni-adjusted P less than the given threshold is less than min_regs. Defaults to 100.
max_regs	numeric indicating the maximum number of regulators to use in the random forest cross-validation testing for feature counts. This number of regulators will be used in rfcv if the number of regulators that associate with the phenotype at a Bonferroni-adjusted P less than the given threshold is greater than max_regs. Defaults to 500.
Bonf_thresh	numeric indicating the significance threshold for Bonferroni-adjusted P used to identify regulators best associated with the phenotype of interest. Defaults to 0.05.
replicates	numeric indicating the number of independent random forest cross-validation analyses to run. Higher numbers will tend to generate cleaner plots (up to a point), but will also add to the analysis time. Defaults to 12.

Value

Writes four output files:

1. A log file recording the standard outputs generated by the function.
2. A png file of the phenotype distributions for the training and test sets.
3. A png file of the random forest cross-validation plot depicting the mean cross-validation error as a function of the number of regulators used as predictors of the phenotype of interest. Error bars represent the standard deviation across all replicates, which by default is set to 12.
4. An .RData workspace image file containing all environment variables from this analysis, which should be used as an input for the final MR analysis with the finalRF() function.

Examples

```
## Not run:
rfCrossVal("Toy_data_regulators_activities.txt",
"GEUVADIS_simulated_phenotype.txt",
"Sim_pheno",
"Toy_data")

## End(Not run)
```

`rfcv.wrapper` *rfcv.wrapper*

Description

This is a wrapper function for calling the rfcv function in the randomForest package with a given seed. This used for multi-threading the rfcv analysis.

Usage

```
rfcv.wrapper(train_x, train_y, seed)
```

Arguments

`train_x`	data.frame containing the activities for the top phenotype associated regulators for the training set.
`train_y`	data.frame containing the phenotype data for the training set.
`seed`	numeric indicating the RNG seed.

Value

Returns a list with the following components: list(n.var=n.var,error.cv=error.cv,predicted=cv.pred)

Examples

```
## Not run:
cv_rndFor <- rfcv.wrapper(train_zact,train_pheno,seed=1)

## End(Not run)
```

`rfcvPlot` *rfcvPlot*

Description

This function determines summary statistics for a list of rfcv objects across all different feature counts, and from them, creates a png image of the random forest cross-validation plot showing prediction error as a function of feature counts.

Usage

```
rfcvPlot(rfcv_list, phenotype, prefix)
```

Arguments

`rfcv_list`	list of length 2 or more of rfcv outputs.
`phenotype`	string indicating phenotype of interest, which is just used in plot title.
`prefix`	string indicating the output file prefix.

Value

Writes a png file of the random forest cross-validation plot that can indicate how many regulators are required to minimize prediction error.

Examples

```
## Not run:
rfcvPlot <- rfcvPlot(cv_rndFor,"Sim_pheno","Toy_data")

## End(Not run)
```

---

trainTestPlot *trainTestPlot*

Description

This function generates a plot that compares the phenotype distributions of the training and test sets to confirm they are comparable. A barplot is used for binary phenotypes while a density plot is used for continuous phenotypes.

Usage

```
trainTestPlot(train_pheno, test_pheno, phenotype, binary = FALSE, prefix)
```

Arguments

`train_pheno`	data.frame containing the training set phenotype data.
`test_pheno`	data.frame containing the test set phenotype data.
`phenotype`	string indicating the column name for the phenotype of interest.
`binary`	logical indicating whether the phenotype is binary. If FALSE, the phenotype is assumed to be continuous. Categorical data with two levels will be treated as binary, but more than two levels will not work. Defaults to FALSE.
`prefix`	string indicating the output file prefix that will be added to the file name along with the column name for the phenotype of interest.

Value

Writes the phenotype distributions plot to png file.

Examples

```
## Not run:
trainTestPlot(train_phenos,test_phenos,"Sim_pheno",prefix="Toy_data")

## End(Not run)
```

write.regulon3 *write.regulon3*

Description

Write a regulon object to text file. This is a modification of the write.regulon() function included in the aracne.networks package that is much faster due to multi-threading.

Usage

```
write.regulon3(
  regulon,
  file = "",
  sep = "\t",
  header = TRUE,
  n = Inf,
  regulator = NULL,
  cpus = future::availableCores(),
  toScreen = F
)
```

Arguments

regulon	A regulon object generated from the aracne2regulon() function in the VIPER package.
file	string for the file name to which the regulon should be written.
sep	string for the character(s) or regular expression escaped character to be used as the field delimiter in the text file. Default is tab, which is the tab-delimiter.
header	logical indicating whether the header should be printed. Default is TRUE.
n	numeric indicating the number of interactions to print. Default is Inf.
regulator	string specifying a particular regulator to which the output pairwise interactions will be restricted. Default is NULL.
cpus	numeric indicating the number of CPUs to use in parallel processing. Defaults to all detected available cores.
toScreen	logical indicating whether the output should also be printed in the console. Defaults to FALSE.

Value

Write the regulon object (aka interactome when all regulons are included) to the indicated file and returns a data.frame version of the regulon formatted as in the output file.

Examples

```
## Not run:
regulon<-aracne2regulon(tissue_net,genes_set,format="3col")
# NOTE: The aracne network file cannot have a header!!!
regTable<-write.regulon3(regulon,file="tissue_interactome.txt",sep=""))

## End(Not run)
```

zscale.genes *zscale.genes*

Description

This function applies Z-scaling for all genes across samples very rapidly using multi-threading when possible.

Usage

```
zscale.genes(genes, exprSet = F)
```

Arguments

`genes`	matrix or data.frame containing gene metrics with genes in rows and samples in columns. Gene and sample identifiers should be in rownames and colnames, respectively.
`exprSet`	logical indicating whether the object returned should be an ExpressionSet object (when TRUE), or a data.frame (when FALSE). Defaults to FALSE.

Value

Returns either an ExpressionSet object (when exprSet=TRUE) or a data.frame (when exprSet=FALSE) with the gene metrics Z-scaled across samples.

Examples

```
## Not run:
z_genes <- zscale.genes(genes)

## End(Not run)
```

Index