Package ‘corTools’

February 19, 2015

Type Package

Title Tools for processing data after a Genome Wide Association Study

Version 1.0

Date 2013-08-22

Author Angela Fan

Maintainer Angela Fan <angela.h.fan@gmail.com>

Description Designed for analysis of the results of a Genome Wide Association Study. Includes tools to pull lists of Chromosome number and SNP position below a certain significance threshold, refine gene networks (including data I/O for Cytoscape), and check SNP base pair changes.

License Artistic License 2.0

NeedsCompilation no

Repository CRAN

Date/Publication 2013-08-23 18:47:41

R topics documented:

corTools-package *Tools for processing data after a Genome Wide Association Study*

Description

Designed for analysis of the results of a Genome Wide Association Study. Includes tools to pull lists of Chromosome number and SNP position below a certain significance threshold, and refine gene networks (including data I/O for Cytoscape), and check SNP base pair changes.

Details

Package:	postGWAS
Type:	Package
Version:	1.0
Date:	2013-08-22
License:	Artistic License 2.0

This package contains two functionalities. The first is to process GWAS data: 1) candpull will find the SNP positions (and Chromosome number) of significant hits. 2) syncheck will check to see if these SNPs cause base pair changes. The second is to understand and visualize the relationships between traits analyzed in GWAS, by refining data to be visualized with Cytoscape: 1) cytosub will subset the traits to reduce the number of interaction lines in Cytoscape. 2) edgecount, dist, and edgecutoff will return the names of traits (nodes) with the most interactions (edges) and the number of those edges in order to find key hubs.

These functions rely on genetic annotation data that is normally downloaded from the web, and read into R as a dataframe. Note that many of the functions require the user to input as an argument the name of the column of the dataframe (such as the column that holds the SNP position information), so the dataframe should have headers.

This package was originally built for use analyzing GWAS results using the model organism Arabidopsis thaliana, but theoretically the same tools can be applied to other datasets.

Author(s)

Angela Fan

Maintainer: Angela Fan <angela.h.fan@gmail.com>

References

www.cytoscape.org

Krouk G, Lingeman J, Colon AM, Coruzzi G, Shasha D. (2013) Gene regulatory networks in plants: learning causality from time and perturbation. Genome Biol 14(6):123.

Katari MS, et al. (2010) VirtualPlant: A software platform to support systems biology research. Plant Physiol 152(2) 500-515.

Krouk G, Mirowski P, LeCun Y, Shasha DE, Coruzzi GM (2010) Predictive network modeling of the high-resolution dynamic plant transcriptome in response to nitrate. Genome Biol 11(12):R123.

Atwell S, et al. (2010) Genome-wide association study of 107 phenotypes in Arabidopsis thaliana inbred lines. Nature 465(7298):627-631.

Manolio TA, et al. (2009) Finding the missing heritability of complex diseases. Nature 461(747).

See Also

GWASTools <http://www.bioconductor.org/packages/2.12/bioc/html/GWASTools.html>

postgwas <http://cran.r-project.org/web/packages/postgwas/vignettes/postgwas.pdf>

`candpull` *Finds Candidate Genes*

Description

Finds candidate genes with p-values less than a user-inputted threshold.

Usage

```
candpull(setnum, setname, traitnum, traitname, threshold)
```

Arguments

`setnum` Number of GWAS dataset results. Must already be read into R.

`setname` Name of GWAS dataset results, as read into R. Datasets must be read in setname#, but only the setname is a required input for this function.

`traitnum` Number of traits analyzed in each GWAS dataset. Number of traits must be consistent across all datasets.

`traitname` Name of trait. Traits must be inputted into dataset columns as traitname#, but only the traitname is a required input for this function.

`threshold` Significance threshold. Function will return list of sets and traits with p-values less than this inputted threshold.

Details

This function provides a high-throughput way to scan multiple files of GWAS results to identify potential candidate genes of interest. This function's output can be exported and used to scan gene annotation data to find the genes corresponding to the chromosome number and SNP position, such as BedTools.

Datasets should be read in as datasetname#, with a number incrementing by 1. Trait columns should be labeled as traitname#, with a number incrementing by 1. Trait names must be consistent across all datasets. Function loops through the datasets and then through each trait column.

Value

Returns a list of sets and traits with p-value less than the specified threshold. If no SNPs have a p-value less than the specified threshold, function will return <0 found> to indicate so.

Author(s)

Angela Fan

References

Atwell S, et al. (2010) Genome-wide association study of 107 phenotypes in Arabidopsis thaliana inbred lines. Nature 465(7298):627-631.

Examples

```
# Create two sample datasets
set1ID <- c(1, 2, 3, 4, 5)
Trait1 <- c(0.005, 0.09, 0.98, 0.767, 0.004)
Trait2 <- c(0.6, 0.89, 0.92, 0.008, 0.4)
Trait3 <- c(0.98, 0.232, 0.53, 0.321, 0.0012)
set1 <- cbind(set1ID, Trait1, Trait2, Trait3)

set2ID <- c(1, 2, 3, 4, 5)
Trait1 <- c(0.43, 0.934, 0.41, 0.43, 0.009)
Trait2 <- c(0.23, 0.423, 0.543, 0.78, 0.99)
Trait3 <- c(0.3423, 0.53, 0.63, 0.765, 0.0053)
set2 <- cbind(set2ID, Trait1, Trait2, Trait3)

candpull(2, "set", 3, "Trait", 0.05)
# 2 denotes the are 2 sets of GWAS datasets
# "set" denotes the dataset name (i.e. set1, set2)
# 3 denotes the number of traits in each dataset- must be the same number
# "Trait" denotes the labels of the columns with trait p-values
# 0.05 is the significance threshold chosen
# Function returns set ID and trait number if the trait in that set has a
# value lower than the inputted threshold, 0.05
```

`cytosub` *Dataset Subsetter*

Description

Subsets a dataset based on text common to the traits that should be removed. Built with Cytoscape datasets in mind, to reduce the complexity of Cytoscape networks.

Usage

```
cytosub(dat, col1, col2, text)
```

Arguments

`dat` Dataset containing the traits

`col1` Column 1 of the dataset, containing trait names, such as a source interaction in Cytoscape.

`col2` Column 2 of the dataset, containing trait names, such as a target interaction in Cytoscape.

`text` Text common to the traits that need to be edited out of the dataset. Regular expressions can also be entered.

Details

This function is built to help interpret GWAS result data by considering relationships between associated genes or traits. It requires the basic input of a network of binary relationships between traits, with two columns of trait names and a third column of the trait interaction. This interaction could be multiple things, such as trait-trait relationships in a pathway, or trait correlation data. This data can be inputted directly into Cytoscape for the visualization of these interaction networks, however for a large network it is useful to pare down the number of interactions. This function subsets the user-inputted data.

Function will preserve columns (such as trait values), but delete rows corresponding to unwanted traits. Regular expressions can also be entered in addition to regular text.

Function uses grepl to text-match to find traits that the user would like to subset out, so regular expressions can also be entered.

Value

Returns a subset of the original dataset, with the unwanted traits edited out, as a dataframe. The returned dataframe will have the same number of columns as the originally inputted dataframe, but with unwanted trait rows removed. The returned dataframe will keep the same header column names as the original dataframe.

Author(s)

Angela Fan

References

http://www.cytoscape.org/

Krouk G, Mirowski P, LeCun Y, Shasha DE, Coruzzi GM (2010) Predictive network modeling of the high-resolution dynamic plant transcriptome in response to nitrate. Genome Biol 11(12):R123.

Examples

```
# Create sample dataset
source.interaction <- c("R1", "R2", "R3", "E1", "E2")
target.interaction <- c("L1", "L2", "L4", "E6", "E7")
values.interaction <- c(1.42, 14.34, 6.43, 32.1, 15.8)
dataset <- cbind(source.interaction, target.interaction, values.interaction)
```

```
cytosub(dataset, source.interaction, target.interaction, E)
# dataset indicates the data we are working with
# source.interaction and target.interaction denote the columns of the dataset
# E indicates the text that we want to search for and edit out of the dataset
# function will return the dataset without "E1", "E2", "E6", "E7"
```

`dist` *Distribution Generator*

Description

Generates a distribution based on a user-inputted list of values, and returns values above or below user-inputted percentages of the distribution.

Usage

```
dist(dat, small, large)
```

Arguments

`dat`	Either a list of numeric values, or a numeric column of a dataframe.
`small`	Smaller percentage, written as a decimal value.
`large`	Larger percentage, written as a decimal value.

Details

If dat is entered as a columnname, function will preserve other columns of the dataframe and return those columns in the function output. Function uses quantiles to determine the cutoff values.

Value

Returns list of values of the dataset that are greater than the larger user-inputted percentage, or smaller than the smaller user-inputted percentage.

Author(s)

Angela Fan

References

http://www.cytoscape.org/

Krouk G, Mirowski P, LeCun Y, Shasha DE, Coruzzi GM (2010) Predictive network modeling of the high-resolution dynamic plant transcriptome in response to nitrate. Genome Biol 11(12):R123.

Krouk G, Lingeman J, Colon AM, Coruzzi G, Shasha D. (2013) Gene regulatory networks in plants: learning causality from time and perturbation. Genome Biol 14(6):123.

Examples

```
# Create some sample data, as a dataframe with a numeric column
col1 <- c("L1", "L2", "L4", "E6", "G1")
col2 <- c(1.42, 14.34, 6.43, 32.1, 15.8)
dat <- as.data.frame(cbind(col1, col2))
dat$col2 <- as.numeric(as.character(dat$col2))

dist(dat$col2, 0.05, 0.95)
# dat$col2 denotes the column of the data that the distribution will be based on
# 0.05 and 0.95 indicate that the function will return values that are smaller
# than 5% of the values, or greater than 95% of the values
# function will return values 1.42 and 32.10
```

`edgecount` *Edge counter*

Description

Counts the number of relationships any given trait makes, to find central hubs of traits with many relationships. Built with Cytoscape in mind, where the function counts edges and returns the names of nodes with the most edges.

Usage

```
edgecount(dat, col1, col2)
```

Arguments

`dat`	Dataframe name, containing the traits entered as columns.
`col1`	First column of trait names
`col2`	Second column of trait names

Details

This function helps identify key nodes in an interaction network by identifying the nodes with the most interactions through an incremented counter. This function will edit out redundant interactions and return only a unique list of traits- this means that interactions that are repeated will not be redundantly counted, and each trait will only be listed once in the returned frequency table.

Value

Returns a dataframe where column 1 is the name of the trait and column 2 is the number of times the trait appeared in the dataset, non redundantly.

Author(s)

Angela Fan

References

http://www.cytoscape.org/

Krouk G, Mirowski P, LeCun Y, Shasha DE, Coruzzi GM (2010) Predictive network modeling of the high-resolution dynamic plant transcriptome in response to nitrate. Genome Biol 11(12):R123.

Examples

```
# Create a sample dataset
Interaction1 <- c("L1", "L2", "L3", "L4", "L19", "R7", "L2")
Interaction2 <- c("L1", "L9", "R1", "R2", "R7", "L4", "R10")
dat <- as.data.frame(cbind(Interaction1, Interaction2))

edgecount(dat, Interaction1, Interaction2)
# dat denotes the name of the data frame
# Interaction1 and Interaction2 denote the column names of the dataframe that contain the
# traits whose interactions you want to count
# function returns a list of unique traits and their frequency of appearance
# Example: L1 only appears once, so it is L1   1
# L4 appears twice, so it is L4   2
```

edgecutoff *Identifies Hubs*

Description

Provides a list of traits with the most edges, identified as traits whose values are one standard deviation greater than the mean. If outliers exist, the function uses the interquartile range and median instead. Built with Cytoscape in mind, so returns the list of nodes with the most edges.

Usage

```
edgecutoff(dat, col)
```

Arguments

dat	Dataset name
col	Column name in the dataset that contains the edgecounts. Edgecounts can be generated by running the edgecount function first, and providing the edgecount function's output as input into edgecutoff.

Details

This function helps to determine which traits or genes could potentially be hubs in an interaction network.

Run edgecount first to generate the edge counts. This function uses boxplot to check for outliers and will generate a boxplot of edgecount data.

Value

Returns a boxplot of the edgecounts, which will be displayed in the R graphics window. Also returns a dataframe that can be exported. Column 1 is the name of the trait, with the header "Var" for variable, and column 2 is the trait's frequency, with the header "Freq."

Author(s)

Angela Fan

References

http://www.cytoscape.org/

Krouk G, Mirowski P, LeCun Y, Shasha DE, Coruzzi GM (2010) Predictive network modeling of the high-resolution dynamic plant transcriptome in response to nitrate. Genome Biol 11(12):R123.

Examples

```
# Create some sample data
traits <- c("Trait1", "Trait2", "Trait3", "Trait4", "Trait5")
edgecount <- c(5, 6, 7, 4, 23)
example <- cbind(traits, edgecount)

edgecutoff(example, edgecount)
# example denotes the data
# edgecount denotes the column of the data that you want to cutoff at
# returns Trait5   23 and a boxplot of the data into the graphics window
```

`syncheck` *Identifies Base Pair Change*

Description

Checks against user-inputted data to see if the SNP causes a basepair change.

Usage

```
syncheck(dat, chrom, pos, col1, col2)
```

Arguments

`dat` Dataset name of the data of chromosome number, SNP position, and base information.

`chrom` Chromosome number you would like to check, corresponding to the SNP position.

`pos` SNP position you would like to check, corresponding to the chromosome number.

`col1` Name of the column of the dataset that holds the chromosome number information.

`col2` Name of the column of the dataset that holds the SNP position information.

Details

This function requires SNP basepair change information that is normally retrieved from the web. This data can also be read into R as a dataframe, but must be read in with a header, as the column names are used as arguments for this function. The information returned from this function can be used to check if the basepair change at that SNP position leads to an amino acid change (synonymous or nonsynonymous) using TAIR and Expasy.

Value

Returns the row of the user-inputted data containing the SNP basepair information of the SNP position on the specificed chromosome.

Author(s)

Angela Fan

References

Atwell S, et al. (2010) Genome-wide association study of 107 phenotypes in Arabidopsis thaliana inbred lines. Nature 465(7298):627-631.

See Also

candpull, to identify SNP positions of interest

Examples

```
# Create sample dataset
chromosome <- c(1, 1, 1, 2, 5)
position <- c(1432, 1542, 6834, 4642, 6435)
bp1 <- c("A", "G", "A", "T", "C")
bp2 <- c("A", "G", "T", "T", "G")
bp3 <- c("A", "C", "A", "G", "C")
bp4 <- c("A", "G", "A", "G", "C")
bp5 <- c("C", "G", "T", "G", "G")
snplist <- cbind(chromosome, position, bp1, bp2, bp3, bp4, bp5)

syncheck(snplist, 1, 6834, chromosome, position)
# snplist is the name of the dataset
# 1 and 6834 represent user query for a SNP hit on that chromosome and at that position
# chromosome and position are the names of the dataset columns that hold the information
# of chromosome and position.
# function returns the information that on chromosome 1, position 6834, the pattern is
# "A" "T" "A" "A" "T"
```

`traitcombos` *Trait combination calculator*

Description

Calculates all simple combinations of traits (addition, subtraction, multiplication, division) and outputs the values

Usage

```
traitcombos(dat, ID)
```

Arguments

`dat`	Dataframe name, containing traits in each column. Trait names must be inputted as a header of the dataframe.
`ID`	Name of the ID column of the dataframe.

Details

ID name must be inputted in quotes, as in "id." Trait combinations are found using the expand.grid function.

Value

Returns a dataframe that includes the original dataframe, but has the trait combinations entered in columns after the original dataframe ends. The combination is specified as a header of the column, and the values are inputted into the columns corresponding to the rows by ID.

Author(s)

Angela Fan

References

Manolio TA, et al. (2009) Finding the missing heritability of complex diseases. Nature 461(747).

Examples

```
# Create a sample dataset
ids <- c(1, 2, 3, 4, 5)
Trait1 <- c(23, 43, 46, 74, 42)
Trait2 <- c(32, 56, 72, 56, 97)
Trait3 <- c(42, 54, 77, 92, 40)
dat <- as.data.frame(cbind(ids, Trait1, Trait2, Trait3))

traitcombos(dat, ”ids”)
# dat denotes the name of the dataframe
# ids is the name of the ID column
# function returns dataset with additional columns added, where the column names
# are the trait combinations and the column values are the appropriate trait calculations
```

Index