# Package 'genomicper'

July 5, 2016

**Type** Package

**Title** Circular Genomic Permutation using Gwas p-Values of Association

**Version** 1.6

**Date** 2016-07-05

**Author** Claudia P. Cabrera, Pau Navarro, Chris S.Haley

**Maintainer** Claudia Cabrera <c.cabrera@qmul.ac.uk>

**Imports** stats,grDevices,utils,graphics,DBI,

**Suggests** KEGG.db,reactome.db,AnnotationDbi

**Description** Circular genomic permutation approach uses GWAS results to establish the significance of pathway/gene-set associations whilst accounting for genomic structure. All SNPs in the GWAS are placed in a 'circular genome' according to their location. Then the complete set of SNP association p-values are permuted by rotation with respect to the SNPs' genomic locations. Two testing frameworks are available: permutations at the gene level, and permutations at the SNP level. The permutation at the gene level uses fisher's combination test to calculate a single gene p-value, followed by the hypergeometric test. The SNP count methodology maps each SNP to pathways/gene-sets and calculates the proportion of SNPs for the real and the permutated datasets above a predefined threshold. Genomicper requires a matrix of GWAS association p-values. The SNPs annotation and pathways annotations can be performed within the package or provided by the user.

**License** GPL-2

**NeedsCompilation** no

**Depends** R (>= 2.10)

**Repository** CRAN

**Date/Publication** 2016-07-05 23:42:23

## R topics documented:

---

`genomicper-package` *Circular Genomic Permutations*

---

## Description

Description: Circular genomic permutation approach uses GWAS results to establish the significance of pathway/gene-set associations whilst accounting for genomic structure. All SNPs in the GWAS are placed in a 'circular genome' according to their location. Then the complete set of SNP association p-values are permuted by rotation with respect to the SNPs' genomic locations. Two testing frameworks are available: permutations at the gene level, and permutations at the SNP level. The permutation at the gene level uses fisher's combination test to calculate a single gene p-value, followed by the hypergeometric test. The SNP count methodology maps each SNP to pathways/gene-sets and calculates the proportion of SNPs for the real and the permutated datasets above a pre-defined threshold. Genomicper requires a matrix of GWAS association p-values. The SNPs annotation and pathways annotations can be performed within the package or provided by the user.

## Details

| | |
|---|---|
| Package: | genomicper |
| Type: | Package |
| Version: | 1.5 |
| Date: | 2016-07-04 |
| License: | GPL-2 |

## Author(s)

Claudia P. Cabrera, Pau Navarro, Chris S. Haley
Maintainer: Claudia Cabrera <c.cabrera@qmul.ac.uk>

## References

SNP-level Permutations:
Genomicper: genome-wide association SNP-set analysis
Claudia P. Cabrera*, Pau Navarro*, Jennifer E. Huffman, Alan F. Wright, Caroline Hayward,Harry Campbell, James F. Wilson, Igor Rudan, Nicholas D. Hastie, Veronique Vitart, Chris S. Haley*

Gene-level Permutations:
Uncovering Networks from Genome-Wide Association Studies via
Circular Genomic Permutation. G3: Genes|Genomes|Genetics 2, 1067-1075.
Claudia P. Cabrera*, Pau Navarro*, Jennifer E. Huffman, Alan F. Wright, Caroline Hayward,Harry Campbell, James F. Wilson, Igor Rudan, Nicholas D. Hastie, Veronique Vitart, Chris S. Haley*

## See Also

Genomicper functions: 1) read_pvals, 2) genome_order, 3) get_pathways, 4) read2_paths, 5A) snps_permutation, 5B) genes_permutation, 6) get_results, 7) plot_results

## Examples

```
##############################################################################
#  Genomicper functions                                              ##########
# 1)  read_pvals(data_name="",snps_ann="")
# 2)  genome_order(all_data="")
# 3)  get_pathways(source="",all_paths="")
# 4)  read2_paths(ordered_alldata="",gs_locs="",sets_from="",sets_prefix="",level="")
# 5A) snps_permutation(ordered_alldata="",pers_ids="",ntraits="",nper="",saveto="",
# threshold="",gs_locs=gs_locs,gper.env = gper.env)
# 5B) genes_permutation(ordered_alldata="",pers_ids="",pathways="",
# ntraits="",nper="",threshold="",saveto="",gs_locs=gs_locs,gper.env = gper.env)
# 6)  get_results(res_pattern="Permus",level="snp",from="workspace",
# threshold=0.05, gper.env = gper.env)
# 7) plot_results(results = "", by = "", plot_all = TRUE, var = "", save_plot = TRUE,
#  plot_name = "", bf = FALSE, save_qq = TRUE)
##############################################################################
############## DEMO: ##########################################################

#### SNP-level  ##############################################################
# SNPs annotation and Pathways provided by user
# all data stored at the WORKSPACE

#library(genomicper)
### Load files for analysis
data(demo,SNPsAnnotation)
# load pathways
data(hsa00100,hsa00120,hsa00130,hsa00140,hsa00190,hsa02010)


# Read & format GWAS pvalues
all_data <- read_pvals(data_name=demo,snps_ann=SNPsAnnotation)
# Order data according to the genome
genome_results <-genome_order(all_data=all_data)
```

```
# Results from genome_order
ordered_alldata <- genome_results$ordered_alldata
gs_locs <- genome_results$gs_locs
# Map SNPs to pathways
paths_res <- read2_paths(ordered_alldata=ordered_alldata,
gs_locs=gs_locs,sets_from="workspace",sets_prefix="hsa",
level="snp",envir=.GlobalEnv)
# Results from read2_paths:
  pers_ids <- paths_res$per_ors
  pathways<- paths_res$pathways

# Create new environment to save the permutations to:
gper.env <- new.env()

# Perform permutations:
snps_permutation(ordered_alldata=ordered_alldata,
pers_ids=pers_ids,ntraits=c(7:13),nper=10,saveto="workspace",
threshold=0.05,gs_locs=gs_locs,envir = gper.env)
# Get results
results <- get_results(res_pattern="Permus",level="snp",
from="workspace",threshold=0.05,envir = gper.env)
# Plot results
## Not run:
#saves plots to working directory
qq <- plot_results(results=results,by="set",plot_all=TRUE)
qq <- plot_results(results=results,by="trait",
plot_all=FALSE,var="trait1")
# Displays interactive plot. Select a trait/set to plot and
# set arguments save_plot=FALSE, plot_all = FALSE
# IMPORTANT: to EXIT interactive plot, RIGHT CLICK on the
# plot and STOP.
qq <- plot_results(results=results,by="set",plot_all=FALSE,
var="hsa00100",save_plot=FALSE)

## End(Not run)
# -- END OF DEMO
################################################
```

---

demo *GWAS p_values demo data*

---

## Description

GWAS p-values (tab delimited file). First Column must contain the SNP ids and the column name = "name"

## Usage

```
data(demo)
```

## Format

A data frame with SNPs identifiers and gwas p-values of association

`name` a character vector

`abpi` a numeric vector

`abpilba` a numeric vector

`abpildfa` a numeric vector

`abpilpta` a numeric vector

`abpirba` a numeric vector

`abpirdfa` a numeric vector

`abpirpta` a numeric vector

`alb` a numeric vector

`avdbp` a numeric vector

```
name       abpi      abpilba    abpildfa  abpilpta  abpirba   abpirdfa
rs10000010 0.9122360 0.30088096 0.2332038 0.5193068 0.1255104 0.07253145
rs10000023 0.8642906 0.52064064 0.9243443 0.7177759 0.9512171 0.81716250
rs10000030 0.2832705 0.99021664 0.8359339 0.9662707 0.8491221 0.50208681
```

## Examples

```
# data(demo)
## use: input file for "read_pvals" function
```

---

`genes_permutation` *Gene-level Permutations*

---

## Description

Performs gene-level circular genomic permutations. In each permutation,the complete set of SNP association p-values are permuted by rotation with respect to the SNPs' genomic locations. Once these 'simulated' p-values are assigned,the joint gene p-values are calculated using Fisher's combination test,and pathways' association tested using the hypergeometric test

## Usage

```
genes_permutation(ordered_alldata = "", pers_ids = "", pathways = "",
ntraits = "", nper = 100, threshold = 0.05, saveto = "workspace",
gs_locs="", envir = "")
```

## Arguments

| | |
|---|---|
| ordered_alldata | Return variable from "genome_order". Ordered genome and trait p-values |
| gs_locs | Return variable from "genome_order". SNP indexes |
| pers_ids | Return variable "per_ors" from "read2_paths". Gene indexes |
| pathways | Return variable "pathways" from "read2_paths" |
| ntraits | Traits INDEX to be analysed. Index according to "ordered_alldata". Trait Columns index must start at 7. Example: ntraits=c(7:9), ntraits=7 |
| nper | Number of permutations.Example: nper=1000 |
| threshold | Threshold to be set by the hypergeometric test. threshold=0.05 |
| saveto | Save permutation results to "workspace" OR "directory" |
| envir | R environment to save the data to when saveto is set to "workspace" |

## Value

Returns "Permus_trait" variables or files (permutation datasets).

## References

Imports phyper (from stats)

## See Also

snps_permutation

## Examples

```
# library(genomicper)

# GWAS DATA
data(demo,SNPsAnnotation)


all_data <- read_pvals(data_name=demo,snps_ann=SNPsAnnotation)
# Prepare Genome
genome_results <-genome_order(all_data=all_data)
# Results from genome_order
ordered_alldata <- genome_results$ordered_alldata
gs_locs <- genome_results$gs_locs

# Load pathway data and details
data(hsa00100,hsa00120,hsa00130,hsa00140,hsa00190,hsa02010)

# Map Genes to pathways
paths_res <- read2_paths(ordered_alldata=ordered_alldata,gs_locs=gs_locs,
sets_from="workspace",sets_prefix="hsa",level="gene",envir=.GlobalEnv)
pers_ids <- paths_res$per_ors
pathways<- paths_res$pathways
```

```
# Create new environment to save data:
gper.env <- new.env()

# Perform Permutations:
genes_permutation(ordered_alldata=ordered_alldata,
pers_ids=pers_ids,pathways=pathways,ntraits=c(7:9),
nper=10,threshold=0.05, saveto="workspace",
gs_locs=gs_locs,envir = gper.env)

# Results
results <- get_results(res_pattern="Permus",level="gene",
from="workspace",threshold=0.05,envir= gper.env)
```

---

genome_order *Genome Order*

---

## Description

Orders the SNPs according to their genomic location

## Usage

```
genome_order(all_data = "")
```

## Arguments

| | |
|---|---|
| all_data | SNPs to Genes Annotation and Trait Pvalues of Association<br>all_data = (read_pvals output) OR matrix/dataframe. |

## Details

```
Input Columns with "*" must be included for analysis

NOTE: Trait p-values must start at Column #7

# *Column 1: "name" (SNP_IDs - any SNP ID as character)
# *Column 2: Chromosome Location
# *Column 3: SNP Location
# *Column 4: Gene ID
# Column 5: Symbol (OR Annotation Field 1)
# Column 6: Annotaiton Field 2
# *Column 7: First trait pvalues of association
# Column N: Next trait pvalues of association
# Example Input Data:
name       Chromosome  Location GENE_ID Symbol Orientation abpi
```

```
rs10000010          4  21618674   80333 KCNIP4          - 0.91
rs10000023          4  95733906     658 BMPR1B          + 0.86
rs10000092          4  21895517   80333 KCNIP4          - 0.20
rs1000022          13  100461219 171425 CLYBL           + 0.26
rs10000300          4  40466547   54502 RBM47           - 0.58
```

## Value

`ordered_alldata`
: SNPs annotated to Genes and Trait p-values

`gs_locs`
: Gene annotations, location indexes and number of observations

## Format

```
SNPs annotated to Genes and Trait p-values
#ordered_alldata[1:5,1:8]
name   Chromosome Location GENE_ID  Symbol Orientation abpi abpilba
rs3934834 1   1005806  NA     <NA>        <NA>    0.97  0.92
rs3737728 1 1021415  54991  C1orf159      -   0.91  0.69
rs6687776 1 1030565  54991  C1orf159      -   0.71  0.45
rs9651273 1 1031540  54991  C1orf159      -   0.22  0.60
rs4970405 1 1048955  54991  C1orf159      -   0.77  0.56

Gene annotations, location indexes and number of observations
#gs_locs[1:5,]
#      Symbol     Chromosome Location    Gene_ID Start_Indx Observations
# [1,] "A1BG"     "19"       "58864479"  "1"     "293976"   "1"
# [2,] "A2M"      "12"       "9232268"   "2"     "215264"   "5"
# [3,] "NAT1"     "8"        "18077310"  "9"     "151804"   "1"
# [4,] "NAT2"     "8"        "18257280"  "10"    "151831"   "2"
# [5,] "SERPINA3" "14"       "95080803"  "12"    "249519"   "2"
```

## See Also

read2_paths

## Examples

```
## DEMO / WORKSPACE #####################################
data(demo,SNPsAnnotation)
all_data<-read_pvals(data_name=demo,snps_ann=SNPsAnnotation)
# GENOME ORDER
genome_results <- genome_order(all_data=all_data)

ordered_alldata <- genome_results$ordered_alldata
gs_locs <- genome_results$gs_locs
#######################################################
```

get_pathways *Pathways*

## Description

Helper function to download pathways and their gene identifiers. KEGG.db and reactome.db are used for pathway annotations.

## Usage

```
get_pathways(source="reactome",all_paths=TRUE,envir = "")
```

## Arguments

| | |
|---|---|
| source | "reactome" or "kegg" |
| all_paths | TRUE or FALSE. If FALSE a subset will be asked by the function |
| envir | R environment to save Pathways to |

## Value

Returns "Pathways description" All downloaded pathways are saved in the workspace If reactome is selected as the source a prefix will be prompt to be set by user. When kegg is selected the organism identifier is set automatically as the prefix (e.g."hsa").

## See Also

read2_paths

## Examples

```
## Not run:
# get pathways source = "kegg"
## library(KEGG.db)

# Create new environment to save data:
gper.env <- new.env()

# paths <- get_pathways(source="kegg",all_paths=FALSE,envir = gper.env)
# when prompted introduce species as listed
# hsa
# if all_paths set to TRUE. All pathways are downloaded automatically
# if all_paths set to FALSE. Introduce list of pathways separated by ","
#hsa00010,hsa00020,hsa04670,hsa04672,hsa04710,hsa04720,hsa04722,hsa04730


# get pathways source = "reactome"
## library(reactome.db)
#paths <- get_pathways(source="reactome",all_paths=FALSE,envir=".GlobalEnv")
```

```
# when prompted introduce species as listed
# Homo sapiens
# when prompted introduce prefix to be assigned to pathways
#HSA
# if all_paths set to TRUE. All pathways are downloaded automatically
# IF all_paths set to FALSE, select a subset of pathway identifiers from
# list. Separated by ","
1500931,1299503,...

## End(Not run)
```

---

`get_results` *Circular Permutation Results*

---

## Description

Creates a summary dataframe of the genomic permutations datasets

## Usage

```
get_results(res_pattern="Permus",level="snp",from="workspace",
threshold=0.05,envir = "")
```

## Arguments

| | |
|---|---|
| `res_pattern` | Pattern of the Permutation files/variable. eg. res=pattern="Permus" |
| `level` | Permutation level performed.level values "snp" or "gene" |
| `from` | Location of the permutation datasets.from values "workspace" or "directory" |
| `threshold` | Threshold of significance set |
| `envir` | R environment where save the data to |

## Value

| | |
|---|---|
| `results` | Data frame with Pathway ID, Trait, Threshold set by permutations,<br>Gene results include the theoretical hypergeometric p-value and the,<br>observed (Empirical Hypergeometric p-values)<br>SNP results include the count of significan SNPs and the overall score<br>Score is the proportion of tests observed with more significant results |

## Format

```
## SNP level results
     PathID    Trait Threshold RealCount Score
1  hsa00010     abpi         0         0 0.037
2  hsa00010 abpildfa         0         0 0.040
3  hsa04720     abpi         2         0 0.311
```

```
## Gene level results
     PathID Trait   Threshold     P-Value  Observed
1  hsa00010  abpi 0.040441176 0.058823529 1.0000000
2  hsa00020  abpi 0.000000000 0.000000000 0.1666667
3  hsa00030  abpi 0.040441176 0.058823529 1.0000000
```

## Examples

```
#library(genomicper)
data(demo,SNPsAnnotation)
all_data <- read_pvals(data_name=demo,snps_ann=SNPsAnnotation)
genome_results <-genome_order(all_data=all_data)
# Results from genome_order
ordered_alldata <- genome_results$ordered_alldata
gs_locs <- genome_results$gs_locs

data(hsa00100,hsa00120,hsa00130,hsa00140,hsa00190,hsa02010)

paths_res <- read2_paths(ordered_alldata=ordered_alldata,gs_locs=gs_locs,
sets_from="workspace",sets_prefix="hsa",level="snp",envir=.GlobalEnv)
pers_ids <- paths_res$per_ors
pathways<- paths_res$pathways

# Create new environment to save data
gper.env <- new.env()

snps_permutation(ordered_alldata=ordered_alldata,pers_ids=pers_ids,
ntraits=c(7,9),nper=10,saveto="workspace",threshold=0.05,
gs_locs=gs_locs,envir= gper.env)

results <- get_results(res_pattern="Permus",level="snp",
from="workspace",threshold=0.05,envir = gper.env)
```

---

# hsaXXXXX *KEGG pathways examples*

---

## Description

Each file "hsaXXXXX" contains the gene indetifiers of the pathway

## Usage

```
data(hsa02010)
```

## Format

10327 124 125 126 127 ...

### Pathways:

hsa00100,hsa00120,hsa00130,hsa00140,hsa00190,hsa02010

### Source

http://www.genome.jp/kegg/

### Examples

```
## Not run:
data(hsa02010)
data(hsa00100,hsa00120,hsa00130,hsa00140,hsa00190,hsa02010)

## End(Not run)
```

---

## hyprbg *Hypergeometric Test (phyper)*

---

### Description

Performs Hypergeometric test (phyper() from R)

### Usage

```
hyprbg(Sig_in_Paths, uniSig, gns_in_Paths, universe)
```

### Arguments

| | |
|---|---|
| Sig_in_Paths | Number of significant genes in the pathway |
| uniSig | Number of significant genes in the dataset |
| gns_in_Paths | Number of genes in the pathway |
| universe | Number of genes in the dataset |

### Value

Returns hypergeometric test

### References

hyprbg Imports phyper() (from stats)

plot_results *Plot Results Circular Permutation*

## Description

QQ plots

## Usage

```
plot_results(results="",by="",plot_all=TRUE, var = "", save_plot=TRUE, plot_name="",
bf= FALSE , save_qq = TRUE)
```

## Arguments

| | |
|---|---|
| results | Results datarame from "get_results()" |
| by | Visualize results by "trait" OR by "set" |
| plot_all | plot_all = TRUE plots all the variables in the results dataframe and saves a pdf file in the working directory. Setting plot_all to FALSE plots a single variable(trait or set). The argument "var" must be declared. |
| var | Variable name to plot |
| save_plot | save_plot = TRUE saves the plots in the working directory. save_plot = FALSE the plot is visualized at the console. save_plot = FALSE can be used only when plot_all is set to FALSE. The plot displayed at the console is interactive, clicking on a point displays the points name. |
| plot_name | Argument used to save the file name for the plots. Default value = Results_genomicper_[set/trait] |
| bf | Displays the bonferroni correction |
| save_qq | TRUE returns the qq plot values |

## Value

| | |
|---|---|
| qq | Data frame with qq plot values |

## See Also

get_results

## Examples

```
#library(genomicper)
data(demo,SNPsAnnotation)
all_data <- read_pvals(data_name=demo,snps_ann=SNPsAnnotation)
genome_results <-genome_order(all_data=all_data)
# Results from genome_order
ordered_alldata <- genome_results$ordered_alldata
gs_locs <- genome_results$gs_locs
```

```
data(hsa00100,hsa00120,hsa00130,hsa00140,hsa00190,hsa02010)


paths_res <- read2_paths(ordered_alldata=ordered_alldata,gs_locs=gs_locs,
sets_from="workspace",sets_prefix="hsa",level="snp",envir=.GlobalEnv)
pers_ids <- paths_res$per_ors
pathways<- paths_res$pathways

# Create new environment to save the permutations to:
gper.env <- new.env()

snps_permutation(ordered_alldata=ordered_alldata,pers_ids=pers_ids,
ntraits=c(7,9),nper=10,saveto="workspace",threshold=0.05,
gs_locs=gs_locs,envir = gper.env)

results <- get_results(res_pattern="Permus",level="snp",
from="workspace",threshold=0.05,envir = gper.env)
## Not run:
#saves plots to working directory
qq <- plot_results(results=results,by="set",plot_all=TRUE)
qq <- plot_results(results=results,by="trait",plot_all=FALSE,var="trait1")
qq <- plot_results(results=results,by="set",
plot_all=FALSE,var="hsa00100",
save_plot=FALSE) ## IMPORTANT: to EXIT interactive plot
## right click on the plot to stop

## End(Not run)
```

---

`read2_paths` *Read to SNPs to sets; Map SNPs to gene-sets/pathways*

---

## Description

Reads the sets/pathways, map the SNPs and genes to the gene-sets/pathways read2_paths uses the "genome_order" output(ordered_alldata, gs_locs) to assign genomic location indexes to each element in the gene-set. The permutation method must be defined (i.e. level = "snp" OR level = "gene").

## Usage

```
read2_paths(ordered_alldata="",gs_locs="",sets_from="workspace",
sets_prefix="hsa",level="snp",envir="")
```

## Arguments

`ordered_alldata`
: Ordered data according to the SNPs genomic location. Traits start at column 7
Return variable from:
genome_results <-genome_order(all_data=all_data)

ordered_alldata <- genome_results$ordered_alldata

| | |
|---|---|
| gs_locs | Gene annotation,indexes and number of observations<br>Return variable from genome_order():<br>genome_results <-genome_order(all_data=all_data)<br>gs_locs <- genome_results$gs_locs |
| sets_from | Location of the gene-sets. Default set to "workspace"<br>sets_from="workspace" OR sets_from="directory"<br>"directory", only will search for information in the working directory. |
| sets_prefix | Prefix of the gene-set variables or files.<br>Default set to sets_prefix= "hsa" e.g. Variables "hsa00010","hsa00020". OR files "hsaXXXXX.txt"<br>each variable/file contains the list of gene identifiers part of that pathway |
| level | The level at which the permutations will be performed. Assigns the indexes according to snps or genes<br>Default value "snp" level values = "snp" OR "gene" |
| envir | R environment where pathway data is stored. e.g(envir=.GlobalEnv, envir=gper.env) |

## Value

| | |
|---|---|
| pathways | Pathway Id, Description,Number of Genes in the pathway, Number of genes found in the dataset, Number of SNPs found in the dataset |
| per_ors | A list of identifiers mapped to each pathway |

## Format

```
Input: Ordered_alldata
name     Chromosome  Location GENE_ID   Symbol Orientation abpi abpilba
rs1001567          1 9194614     <NA>     <NA>        <NA> 0.96 0.89
rs1000313          1 15405489   23254 KIAA1026           + 0.93 0.57
rs1002365          1 19797248    <NA>     <NA>        <NA> 0.68 0.58
rs1002706          1 25051153    <NA>     <NA>        <NA> 0.71 0.02
rs1002487          1 26865971    6195  RPS6KA1           + 0.98 0.78

Input:gs_locs
      Symbol   Chromosome Location    Gene_ID Start_Indx Observations
 [1,] "ACYP2"  "2"        "54399633"  "98"    "35"       "1"
 [2,] "AMPD3"  "11"       "10514707"  "272"   "898"      "1"
 [3,] "ANK2"   "4"        "113830885" "287"   "479"      "4"

Input:pathway example
hsa04720
 [1]  10411    107  11261    114   1387 163688   ....

Output:pathways
ID         Name                          GenesInPath GenesFound SNPsInPath
```

```
"hsa00010" "Glycolysis / Gluconeogenesis" "  66"      "1"        "1"
"hsa00020" "Citrate cycle (TCA cycle)"    "  31"      "0"        "0"
"hsa00030" "Pentose phosphate pathway"    "  27"      "1"        "1"
```

## See Also

genes_permutation snps_permutation genome_order

## Examples

```
## DEMO - SNP Level data stored in workspace ######################
# library(genomicper)
data(demo,SNPsAnnotation)
all_data <- read_pvals(data_name=demo,snps_ann=SNPsAnnotation)
genome_results <-genome_order(all_data=all_data)
ordered_alldata <- genome_results$ordered_alldata
gs_locs <- genome_results$gs_locs
data(hsa00100,hsa00120,hsa00130,hsa00140,hsa00190,hsa02010)

paths_res <- read2_paths(ordered_alldata=ordered_alldata,
gs_locs=gs_locs,sets_from="workspace",sets_prefix="hsa",
level="snp",envir=.GlobalEnv)
pers_ids <- paths_res$per_ors
pathways<- paths_res$pathways
#################################################################
```

---

# read_pvals *Read GWAS p-values of association and Merge with SNP annotations*

---

## Description

Read GWAS p-values of association and Merge with SNP annotations for analysis

## Usage

```
read_pvals(data_name="",snps_ann="",from="workspace")
```

## Arguments

| | |
|---|---|
| data_name | GWAS p_values (tab delimited file)(SNP_IDs Trait1 Trait2 ...TraitN) |
| snps_ann | SNPs Annotation (SNPsAnnotation). Genomicper uses entrez gene ids to annotate associate SNPs-to genes-pathways<br>The annotation MUST match your data input (coordinates and chromosome format)<br>Any SNP ID is valid, as long the ID is set as character<br>The examples below show an option on how to annotate the SNPs prior the use of genomicper |
| from | Datasets location. Values "workspace" OR "directory" |

## Value

Dataframe: name; chromosome; Location; GeneID; Symbol; Orientation; Trait1; TraitN

## Formats

```
GWAS p_values (tab delimited file)(SNP_IDs Trait1 Trait2 ...TraitN)
name       abpi      abpilba    abpildfa
rs10000010 0.9122360 0.30088096 0.2332038
rs10000023 0.8642906 0.52064064 0.9243443
rs10000030 0.2832705 0.99021664 0.8359339

SNPs Annotation (SNPsAnnotation)
name        Chromosome  Location  GENE_ID   Symbol    Orientation
rs1000313   1           15405489  23254     KIAA1026  +
rs1000533   1           168282491 9095      TBX19     +
rs1000731   1           231963491 27185     DISC1     +

Output:
name       Chromosome  Location GENE_ID Symbol Orientation      abpi
rs10000010          4  21618674   80333 KCNIP4           - 0.9122360
rs10000023          4  95733906     658 BMPR1B           + 0.8642906
rs10000030          4 103374154      NA   <NA>        <NA> 0.2832705
```

## See Also

genome_order

## Examples

```
## DEMO // WORKSPACE
data(demo,SNPsAnnotation)
all_data <- read_pvals(data_name=demo,snps_ann=SNPsAnnotation)

## Not run:
##
##  Below is an example on how to annotate the SNPs prior the use of genomicper
##  using UCSC table browser and intersectBed from bedtools:

##  The function intersectBed from bedtools can be used to annotate SNPs to genes.
##  This function needs the locations to be annotated as input, and a reference file
##  to annotate to. Genomicper uses entrez gene ids to annotate associate SNPs-to genes-pathways.

# prepare locations INPUT: chr position position other-info

# 1       10763241        10763241        1_10763241_C_T_1
# 1       10764465        10764465        1_10764465_T_C_1
# 1       10767685        10767685        1_10767685_C_T_1

# Prepare the file to annotate to. Using UCSC table browser.
# clade:Mammal genome:Human assembly: Feb2009(GRCh37/hg19)
```

```
# group: All tables database:hg19 Table: knownToLocusLink
# output format: selected fields from primary and related tables
# click on "get output"
# Next select Linked Tables: kgXref and knownGene
# click on "allow filtering using fields in checked tables"
# Select fields for output:
# Entrez Gene ID from hg19.knownToLocusLink
# Gene Symbol from hg19.kgXref
# Reference sequence chromosome or scaffold from hg19.knownGene
# + or - for strand from hg19.knownGene
# Transcription start position from hg19.knownGene
# Transcription end position from hg19.knownGene
# click on "get output"
# Table will include more than one mapping, to avoid results bias  decrease/increase
# the min and max according to the wished annotations for a single gene
# (eg. take min and max of all isoforms or desiered kb distance)

# Reformat Table to intersectBed accepted formats (eg.GTF/BED/VCF)
# awk 'BEGIN{FS="\t";OFS="\t"}{print $3,$5,$6,$1,$2,$4}' Genes_hg19_TableBrowser.txt |
# sed 's/chr//g' | awk 'BEGIN{FS="\t";OFS="\t"}{if($1 !~ /[:alnum:]/) print $0}' > Genes_TEMP.txt

# R >
# x <- read.table("Genes_TEMP.txt",sep="\t",header=F,stringsAsFactors=F)
# genes <- unique(sort(x[,5]))
# gene_table <- matrix(data=NA,ncol=6,nrow=0)
# for(i in genes){
# grids <- which(x[,5] == i)
# min <- x[grids[which.min(x[grids,2])],2]
# max <- x[grids[which.max(x[grids,3])],3]
# gene_table <- rbind(gene_table,c(x[grids[1],1],min,max,
#           x[grids[1],4],x[grids[1],5],x[grids[1],6]))
# }
# write.table(gene_table,file="Gene_Table.txt",col.names=F,row.names=F,sep="\t",quote=F)
# /exit R

## If you are trying to intersect very large files and are having trouble
## with excessive memory usage, please presort your data by chromosome
## and then by start position e.g.: sort -k1,1 -k2,2n in.bed > in.sorted.bed
## for BED files) and then use the -sorted option
## sort -k1,1 -k2,2n Gene_Table.txt > Gene_Table_sorted.txt

## Intersect command:
# intersectBed -a inp.txt -b Gene_Table_sorted.txt -wa -wb -sorted > temp
# Select Columns : SNP_ID,CHR,SNP_Location,GeneID,OtherAnnotation1,OtherAnnotation2
# awk 'BEGIN{FS="\t";OFS="\t"}{print $4,$5,$2,$8,$9,$10}' temp > SNP_Table_Annotation.txt

# data ready for genomicper:
# head SNP_Table_Annotation.txt
# rs1000313       1       15405489        23254   KAZN    +
# rs1002365       1       19797248        832     CAPZB   -
# rs1002487       1       26865971        6195    RPS6KA1 +
# rs1002358       1       53753718        7804    LRP8    -
# rs1001160       1       76358591        4438    MSH4    +
```

```
# rs1002784       1       76824595        256435  ST6GALNAC3      +
# rs1001193       1       147166377       400818  NBPF9   +
# rs1001193       1       147166377       728841  NBPF8   +
# rs1001193       1       147166377       728855  LINC00623       +
# rs1001193       1       147166377       653505  PPIAL4B +

## End(Not run)
```

---

SNPsAnnotation *SNPs-Genes annotation to Distance 0 (SNPs within a gene)*

---

## Description

SNPs annotated to genes. Annotation only when the SNPs fall within start and end of transcription of the genes.

## Usage

```
data(SNPsAnnotation)
```

## Format

Sample data frame with 339096 SNP observations on the following 6 variables.

`name` a character vector

`Chromosome` a character vector

`Location` a numeric vector of the SNP location

`GENE_ID` a numeric vector with entrez geneID

`Symbol` a character vector ; other annotation slot 1

`Orientation` a character vector; other annotation slot 2

```
name      Chromosome  Location  GENE_ID    Symbol     Orientation
rs1000313          1  15405489    23254    KIAA1026             +
rs1000533          1 168282491     9095    TBX19                +
rs1000731          1 231963491    27185    DISC1                +
```

## Source

NCBI Gene database,(http://www.ncbi.nlm.nih.gov/gene ; Build.37.1).

## Examples

```
# data(SNPsAnnotation)
```

snps_permutation *SNP-level permutations*

## Description

Performs SNP-level circular genomic permutations. In each permutation, the complete set of SNP association p-values are permuted by rotation with respect to the SNPs' genomic locations.
Once these 'simulated' p-values are assigned,the proportion of SNPs per set above a pre-defined threshold is calculated

## Usage

```
snps_permutation(ordered_alldata = "", pers_ids = "", ntraits = "",
nper = 100, threshold = 0.05, saveto = "workspace",
gs_locs = "",envir ="")
```

## Arguments

| | |
|---|---|
| ordered_alldata | Return variable from "genome_order". Ordered genome and trait p-values |
| gs_locs | Return variable from "genome_order". SNP indexes |
| pers_ids | Return variable "per_ors" from "read2_paths". SNP indexes |
| ntraits | Traits INDEX to be analysed. Index according to "ordered_alldata". Trait Columns index must start at 7. Example: ntraits=c(7:9), ntraits=7 |
| nper | Number of permutations.Example: nper=1000 |
| threshold | Threshold to be set by the hypergeometric test. threshold=0.05 |
| saveto | Save permutation results to "workspace" OR "directory" |
| envir | R environment to save the Permutations to when saveto is set to "workspace" |

## Value

Returns "Permus_genesetsname" variables or files (permutation datasets).

## See Also

genes_permutation

## Examples

```
# library(genomicper)
data(demo,SNPsAnnotation)
all_data <- read_pvals(data_name=demo,snps_ann=SNPsAnnotation)
genome_results <-genome_order(all_data=all_data)
# Results from genome_order
ordered_alldata <- genome_results$ordered_alldata
```

```
gs_locs <- genome_results$gs_locs
data(hsa00100,hsa00120,hsa00130,hsa00140,hsa00190,hsa02010)
paths_res <- read2_paths(ordered_alldata=ordered_alldata,gs_locs=gs_locs,
sets_from="workspace",sets_prefix="hsa",level="snp",envir=.GlobalEnv)
pers_ids <- paths_res$per_ors
pathways<- paths_res$pathways

# Create new environment to save the permutations to:
gper.env <- new.env()

# permutations
snps_permutation(ordered_alldata=ordered_alldata,pers_ids=pers_ids,
ntraits=c(7,9),nper=10,saveto="workspace",threshold=0.05,
gs_locs=gs_locs,envir = gper.env)
```

# Index