# GGtools

April 19, 2010

---

`aafSNP-class` *Class "aafSNP" – container for HTML rendering of SNP metadata*

---

### Description

Class "aafSNP" – container for HTML rendering of SNP metadata

### Objects from the Class

Objects can be created by calls of the form `new("aafSNP", ...)`.

### Slots

`.Data`**:** Object of class `"character"` will typically hold rs ids from dbSNP

### Extends

Class `"character"`, from data part.

### Methods

The constructor has the same name, and operates on a list of character vectors. It is expected that you would have a vector of rs numbers for each gene, thus a list of vectors with elements corresponding to genes.

getURL, getHTML are defined; see `getURL`, for example, Apes the handling of UniGene links.

### Examples

```
showClass("aafSNP")
```

`cisSnpTests` *perform tests for eQTL cis to specified genes*

## Description

perform tests for eQTL cis to specified genes

## Usage

```
cisSnpTests(fmla, smls, radius, ...)
```

## Arguments

| | |
|---|---|
| `fmla` | standard formula. LHS can be a GeneSet with AnnotationIdentifier geneIdType. RHS can be predictor formula component using variables in pData of `smls` |
| `smls` | instance of smlSet |
| `radius` | numeric value: number of bases up and downstream from probe CHRLOC to be examined for SNP |
| `...` | not in use |

## Value

a list of cwSnpScreen instances

## Note

Getting SNP locations is slow for the first event while metadata are brought into scope. Subsequent calls are faster.

## Author(s)

VJ Carey <stvjc@channing.harvard.edu>

## Examples

```
library(GSEABase)
# two genes on chr 20
gs1 = GeneSet(c("CPNE1", "ADA"), geneIdType = SymbolIdentifier())
gs2 = gs1
organism(gs2) = "Homo sapiens"
geneIdType(gs2) = AnnotationIdentifier("illuminaHumanv1.db")
if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
cc = cisSnpTests(gs2~male, hmceuB36.2021, radius=1e5)
lapply(cc, function(x) length(p.value(x@.Data[[1]])))
cc = cisSnpTests(gs2~male, hmceuB36.2021, radius=1e6)
lapply(cc, function(x) length(p.value(x@.Data[[1]])))
```

GGtools-package *GGtools Package Overview*

## Description

GGtools Package Overview

## Details

This package provides facilities for analyzing relationships between gene expression distributions (singly or in groups) and SNP genotype series (chromosome-specific or genome-wide). The gwSnpTests method is the primary interface.

Important data classes in use: smlSet-class, gwSnpScreenResult-class, defined in GGBase package.

Main data sets: hmceuB36.2021, an excerpt based on chromosomes 20 and 21, with genotypes for all phase II HapMap SNP and full expression data for 90 CEU HapMap cohort members.

Introductory information is available from vignettes, type openVignette().

Full listing of documented articles is available in HTML view by typing help.start() and selecting GGtools package from the Packages menu or via library(help="GGtools").

## Author(s)

V. Carey

gwSnpTests *methods for iterating association tests (expression vs SNP) across genomes or chromosomes*

## Description

methods for iterating association tests (expression vs SNP) across genomes or chromosomes

## Usage

```
gwSnpTests(sym, sms, cnum, cs, ...)
```

## Arguments

| | |
|---|---|
| sym | genesym, probeId, or formula instance |
| sms | smlSet instance |
| cnum | chrnum instance or missing |
| cs | chunksize specification |
| ... | ... |

## Details

invokes snpMatrix package test procedures (e.g., snp.rhs.tests as appropriate

chunksize can be specified to divide task up into chunks of chromosomes; gc() will be run between each chunk – this may lead to some benefits when memory capacity is exceeded

The dependent variable in the formula can have class genesym (chip annotation package used for lookup), probeId (direct specification using chip annotation vocabulary), or phenoVar (here we use a phenoData variable as dependent variable). If you want to put expression values on the right-hand side of the model, add them to the phenoData and enter them in the formula.

## Value

gwSnpScreenResult-class or cwSnpScreenResult-class instance

## Author(s)

Vince Carey <stvjc@channing.harvard.edu>

## Examples

```
if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
# condense to founders only
hmFou = hmceuB36.2021[, which(hmceuB36.2021$isFounder)]
# show basic formula fit
f1 = gwSnpTests(genesym("CPNE1")~male, hmFou, chrnum(20))
f1
plot(f1)
# show how to avoid adjusted fit
f1b = gwSnpTests(genesym("CPNE1")~1-1, hmFou, chrnum(20))
# show gene set modeling on chromosome
library(GSEABase)
gs1 = GeneSet(c("CPNE1", "ADA"))
geneIdType(gs1) = SymbolIdentifier()
f2 = gwSnpTests(gs1~male, hmFou, chrnum(20))
f2
names(f2)
plot(f2[["ADA"]])
# show 'smlSet-wide' fit
f3 = gwSnpTests(gs1~male, hmFou)
f3
# now use a phenoVar
f3b = gwSnpTests(phenoVar("persid")~male, hmFou, chrnum(20))
topSnps(f3b)
## Not run:
# in example() we run into a problem with sys.call(2); works
# in interpreter
f4 = gwSnpTests(gs1~male, hmFou, snpdepth(250), chunksize(1))
f4

## End(Not run)
```

`hbTestResults-class`

*Class "hbTestResults" holds results of tests of association of expression levels with haplotype within haplotype block*

## Description

Class "hbTestResults" holds results of tests of association of expression levels with haplotype within haplotype block

## Objects from the Class

Objects can be created by calls of the form `new("hbTestResults", ...)`.

## Slots

`formula`: specify gene of interest and covariates from pData

`hscores`: Object of class `"list"` series of haplo.stats:::haplo.score results for blocks

`locs`: Object of class `"numeric"` locations at which blocks were found (mean location within each block)

`chrnum`: Object of class `"chrnum"` chromosome being analyzed

`smlSetName`: Object of class `"character"` name of the `smlSet-class` harboring data in use

`rsid`: Object of class `"ANY"` can be a dbSNP id to use as an anchor, or a number constituting absolute chromosomal location at which blocks will be sought

`rad`: Object of class `"numeric"` radius in base pairs around the `rsid` to be searched for blocks

`ldStruc`: Object of class `"ANY"` the result of the mapLD:::mapLD function

## Methods

**pvals** `signature(x = "hbTestResults")`: extracts p-values for global score tests, one per block

**locs** `signature(x = "hbTestResults")`: extracts locations of haplotype blocks found (average SNP location within block)

**hscores** `signature(x = "hbTestResults")`: extracts `haplo.score` results as a list, for all blocks

## Author(s)

VJ Carey <stvjc@channing.harvard.edu>

## Examples

```
showClass("hbTestResults")
```

## hbTests-methods *haplotype-block based tests for structured expression variation*

### Description

haplotype-block based tests for structured expression variation

### Methods

**fmla = "genesym", sms = "smlSet", cnum = "chrnum", rsid = "numeric", rad = "numeric"** expression data for gene identified by `genesym` is extracted from `sms`, and genotype data within `rad` base pairs of `rsid` are obtained and processed by `mapLD` to define haplotype blocks and the SNP tagging these blocks. Score tests are then computed for the association of expression of the gene identified by `genesym` with haplotype copy number (additive model by default, but options captured by ... are passed to `haplo.score`.)

### Examples

```
library(GGtools)
data(hmceuB36.2021)
hmFou = hmceuB36.2021[, hmceuB36.2021$isFounder==TRUE]
hh = hbTests(genesym("CPNE1"), hmFou, chrnum(20), 33600000, 2e4 )
hh
pvals(hh)
plot(locs(hh), -log10(pvals(hh)))
hscores(hh)[[which.min(pvals(hh))]]
```

## hla2set *a gene set of 9 genes from human HLA2 locus*

### Description

a gene set of 9 genes from human HLA2 locus

### Usage

```
data(hla2set)
```

### Format

The format is: Formal class 'GeneSet' [package "GSEABase"] with 13 slots

..@ geneIdType :Formal class 'SymbolIdentifier' [package "GSEABase"] with 2 slots

.. .. ..@ type :Formal class 'ScalarCharacter' [package "Biobase"] with 1 slots

and so on.

See `GeneSet-class` for additional information.

### Details

This set of 9 genes related to human HLA2 locus was used in the 2009 Bioinformatics Application Note by Carey, Davis et al.

### Examples

```
data(hla2set)
geneIds(hla2set)
```

---

`hmceuB36.2021` *two chromosomes of genotype data and full expression data for CEPH CEU hapmap data*

---

### Description

two chromosomes of genotype data and full expression data for CEPH CEU hapmap data

### Usage

```
data(hmceuB36.2021)
```

### Format

The format is: Formal class 'smlSet' [package "GGBase"] with 9 slots

..@ smlEnv :<environment: 0x3902e98>

..@ annotation : chr "illuminaHumanv1.db"

..@ chromInds : num [1:2] 20 21

..@ organism : chr "Hs"

..@ assayData :<environment: 0x3c96504>

..@ phenoData :Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots

..@ featureData :Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots

..@ experimentData :Formal class 'MIAME' [package "Biobase"] with 13 slots

..@ ...classVersion..:Formal class 'Versions' [package "Biobase"] with 1 slots

### Examples

```
#data(hmceuB36.2021)
```

---

`invokePhase-methods` *~~ Methods for Function invokePhase in Package 'GGtools' ~~*

---

### Description

~~ Methods for function `invokePhase` in Package 'GGtools' ~~

### Methods

**x = "snp.matrix", cnum = "chrnum", parmstring = "character", globpname = "character", where2run = "chara**
transform snp.matrix entity to phaseInput (uses tempfile()) and invokes PHASE

**x = "phaseInput", cnum = "chrnum", parmstring = "character", globpname = "character", where2run = "chara**
for prepared 'phaseInput' structure, invoke PHASE

### Examples

```
## Not run:
data(smtest)
invokePhase(smtest, chrnum(20), "", Sys.getenv("PHASE_LOC"),
   ".", TRUE)

## End(Not run)
```

---

## make_smlSet *create an smlSet instance from components*

---

### Description

create an smlSet instance from components

### Usage

```
make_smlSet(es, sml, organism = "Homo sapiens")
```

### Arguments

| | |
|---|---|
| es | ExpressionSet instance |
| sml | list of snp.matrix instances |
| organism | string naming organism |

### Details

combines snp.matrix instances with expression data

### Value

instance of smlSet class

### Author(s)

VJ Carey <stvjc@channing.harvard.edu>

### Examples

```
data(hmceuB36.2021) # here we just show the mechanics from a working smlSet
hh = hmceuB36.2021
sl = smList(hh)
ex = exprs(hh)
pd = phenoData(hh)
ed = experimentData(hh)
fd = featureData(hh)
es = new("ExpressionSet", exprs=ex, phenoData=pd, experimentData=ed,
   featureData=fd)
mm = make_smlSet(es, sl)
mm
```

masterSnps *visualize a multiGwSnpScreenResult*

## Description

visualize a multiGwSnpScreenResult

## Usage

```
masterSnps(mgw, n = 50, auto = TRUE, orgdb = "org.Hs.eg.db", minl10 = 5,
gstart = 0, gend = 3e+09,
genomesize = 3e+09, pcex = 1, pal = rainbow(20), numxax=FALSE, ...)
```

## Arguments

| | |
|---|---|
| mgw | a multiGwSnpScreenResult, for example from gwSnpTests with a GeneSet on lhs of formula |
| n | number of best snps to retain per gene |
| auto | restrict attention to autosomes? |
| orgdb | an annotation library like org.Hs.eg.db |
| minl10 | threshold of -log10 p above which we keep SNP for plotting |
| gstart | position at which genome-wide SNP locations begin |
| gend | position at which genome-wide SNP locations end |
| genomesize | number of bases over which plotting will be conducted (e.g., ylim=c(0, genome-size) ) |
| pcex | cex setting for pch of plot |
| pal | a palette to differentiate gene coloring |
| numxax | logical: if TRUE, x axis labels genomic coordinates, otherwise chromosome |
| ... | args passed to plot() |

## Details

experimental display with snp location as ordinate and gene location as abscissa – point plotted if snp is associated with gene at p smaller than the threshold specified

## Value

a list with self-describing elements

## Author(s)

VJ Carey <stvjc@channing.harvard.edu>

## Examples

```
if (require("GGdata")) {
 data(fheadFits)
 mm7 = masterSnps(fheadFits, minl10=7, pal=rainbow(10))
 }
```

## plot-methods *Methods for Function plot in Package 'GGtools'*

### Description

Methods for function `plot` in Package 'GGtools'

### Methods

**x = "cwSnpScreenResult", y = "missing"** shows results of chromosome-wide screen for expression-associated SNP

**x = "filteredGwSnpScreenResult", y = "ANY"** shows results of genome-wide screen for expression-associated SNP

**x = "filteredMultiGwSnpScreenResult", y = "ANY"** fails, need to pick gene at this time

## snpm2mapLD *prepare input to mapLD function for haplotype block identification*

### Description

prepare input to mapLD function for haplotype block identification

### Usage

```
snpm2mapLD(x, chrnum, runMAP=TRUE, ...)
```

### Arguments

| | |
|---|---|
| `x` | snp.matrix instance |
| `chrnum` | chromosome number |
| `runMAP` | logical indicating whether or not to run mapLD |
| `...` | additional arguments to mapLD |

### Details

sets up a data frame suitable for mapLD, and will invoke with appropriate arguments identifying columns for alleles and other identifiers if runMAP is TRUE (default).

smtest is a small snp.matrix instance

### Value

a list with element struc holding the data frame, and mapLD output if requested. Note that mapLD writes an eps file to disk *sigh*.

### Author(s)

Vince Carey <stvjc@channing.harvard.edu>

### Examples

```
data(smtest)
ss = snpm2mapLD(smtest, chrnum=20, runMAP=FALSE)
ss
# you could run mapLD on ss[[1]]
```

---

`snpm2phase` *convert information in a snp.matrix to PHASE input format; invokePhase can run a suitably installed version of PHASE*

---

### Description

convert information in a snp.matrix to PHASE input format; invokePhase can run a suitably installed version of PHASE

### Usage

```
snpm2phase(snpm, cnum, outfilename)
parsePh.out(fn)
personalHap(x)
```

### Arguments

| | |
|---|---|
| `snpm` | snp.matrix instance |
| `cnum` | chromosome number as chrnum instance |
| `outfilename` | character name of file to write |
| `fn` | character name of PHASE .out file to read |
| `x` | output of parsePh.out |

### Details

follows phase 2.1 documentation for input format

a phaseInput container class can store relevant metadata

### Value

writes to a file and gives a message

### Author(s)

Vince Carey <stvjc@channing.harvard.edu>

## Examples

```
data(smtest)
tt = tempfile()
pin = snpm2phase(smtest, chrnum(20), tt)

class(pin)
getClass("phaseInput")
pin
readLines(tt)
unlink(tt)
pp = parsePh.out(system.file("phaseOut/cpne1_20k.out", package="GGtools"))
pp[[1]][1:3]
personalHap(pp)
```

---

`strMultPop` *serialization of a table from Stringer's multipopulation eQTL report*

---

## Description

serialization of a table from Stringer's multipopulation eQTL report

## Usage

```
data(strMultPop)
```

## Format

A data frame with 39649 observations on the following 12 variables.

`rsid` a factor with levels rs...

`genesym` a factor with levels `37865` `39692` `ABC1` `ABCD2` `ABHD4` `ACAS2` ...

`illv1pid` a factor with levels `GI_10047105-S` `GI_10092611-A` `GI_10190705-S` `GI_10567821-S` `GI_10835118-S` `GI_10835186-S` ...

`snpChr` a numeric vector

`snpCoordB35` a numeric vector

`probeMidCoorB35` a numeric vector

`snp2probe` a numeric vector

`minuslog10p` a numeric vector

`adjR2` a numeric vector

`assocGrad` a numeric vector

`permThresh` a numeric vector

`popSet` a factor with levels `CEU-CHB-JPT` `CEU-CHB-JPT-YRI` `CHB-JPT`

## Details

imported from the PDF(!) distributed by Stranger et al as supplement to PMID 17873874

## Source

PMID 17873874 supplement

## References

PMID 17873874 supplement

## Examples

```
data(strMultPop)
strMultPop[1:2,]
```

---

`topSnps-methods` *report on most significant SNP with gwSnpTests results*

---

## Description

report on most significant SNP with gwSnpTests results

## Methods

**x = "cwSnpScreenResult"** also takes argument n for number to report

**x = "gwSnpScreenResult"** also takes argument n for number to report

---

`GGtools-RangedData` *Transform results of gwSnpTests to browser tracks*

---

## Description

Create a browser track from a chromosome-wide SNP screen

## Coercion

`as(object, "RangedData")`: Coerce a `cwSnpScreenResult`, `object`, to a `RangedData` instance, with the genomic coordinates -log10 p-values for each SNP

# Index


∗Topic **classes**
aafSNP-class, 1
hbTestResults-class, 5
∗Topic **datasets**
hla2set, 6
hmceuB36.2021, 7
strMultPop, 12
∗Topic **methods**
GGtools-RangedData, 13
hbTests-methods, 6
invokePhase-methods, 7
plot-methods, 10
topSnps-methods, 13
∗Topic **models**
cisSnpTests, 2
gwSnpTests, 3
make_smlSet, 8
masterSnps, 9
snpm2mapLD, 10
snpm2phase, 11
∗Topic **package**
GGtools-package, 3

aafSNP *(aafSNP-class)*, 1
aafSNP-class, 1

character, *1*
chunksize *(gwSnpTests)*, 3
chunksize-class *(gwSnpTests)*, 3
cisSnpTests, 2
coerce,cwSnpScreenResult,RangedData-method *(GGtools-RangedData)*, 13
cwSnpScreenResult, *13*
cwSnpScreenResult-class, *4*

GeneSet-class, *6*
getURL, *1*
GGtools *(GGtools-package)*, 3
GGtools-package, 3
GGtools-RangedData, 13
gwSnpScreenResult-class, *3, 4*
gwSnpTests, 3, *3*
gwSnpTests,formula,smlSet,cnumOrMissing,missing-method *(gwSnpTests)*, 3
gwSnpTests,formula,smlSet,cnumOrMissing,missing-method *(gwSnpTests)*, 3
gwSnpTests,formula,smlSet,cnumOrMissing-method *(gwSnpTests)*, 3
gwSnpTests,formula,smlSet,snpdepth,ANY-method *(gwSnpTests)*, 3
gwSnpTests,formula,smlSet,snpdepth,chunksize-method *(gwSnpTests)*, 3
gwSnpTests,formula,smlSet,snpdepth,missing-method *(gwSnpTests)*, 3
gwSnpTests,formula,smlSet,snpdepth-method *(gwSnpTests)*, 3

haplo.score, *5, 6*
hbTestResults-class, 5
hbTests *(hbTests-methods)*, 6
hbTests,genesym,smlSet,chrnum,numeric,numeric-method *(hbTests-methods)*, 6
hbTests-methods, 6
hla2set, 6
hmceuB36.2021, *3*, 7
hscores *(hbTestResults-class)*, 5
hscores,hbTestResults-method *(hbTestResults-class)*, 5

invokePhase *(invokePhase-methods)*, 7
invokePhase,phaseInput,chrnum,character,character-method *(invokePhase-methods)*, 7
invokePhase,snp.matrix,chrnum,character,character-method *(invokePhase-methods)*, 7
invokePhase-methods, 7

locs *(hbTestResults-class)*, 5
locs,hbTestResults-method *(hbTestResults-class)*, 5

make_smlSet, 8
mapLD, *6*
masterSnps, 9

parsePh.out *(snpm2phase)*, 11
pg,ANY-method *(snpm2phase)*, 11
phaseInput-class *(snpm2phase)*, 11

plot,cwSnpScreenResult,missing-method
(plot-methods), 10
plot,filteredGwSnpScreenResult,ANY-method
(plot-methods), 10
plot,filteredMultiGwSnpScreenResult,ANY-method
(plot-methods), 10
plot,snp.reg.imputation,missing-method
(plot-methods), 10
plot-methods, 10
pvals (hbTestResults-class), 5
pvals,hbTestResults-method
(hbTestResults-class), 5

RangedData, 13
residTests (gwSnpTests), 3
residTests,cwSnpScreenResult,smlSet,formula,missing-method
(gwSnpTests), 3

smlSet, 3
smlSet-class, 3, 5
smtest (snpm2mapLD), 10
snp.rhs.tests, 4
snpm2mapLD, 10
snpm2phase, 11
strMultPop, 12

topSnps (topSnps-methods), 13
topSnps,cwSnpScreenResult-method
(topSnps-methods), 13
topSnps,gwSnpScreenResult-method
(topSnps-methods), 13
topSnps-methods, 13