GGBase

April 19, 2009

`genesym-class` *Class "genesym" and other casting classes*

Description

classes that help establish symbol semantics for dispatching

Objects from the Class

Objects can be created by calls of the form `new("genesym", ...)`. and generally just extend character or numeric so that vector operations are straightforward

Currently, `genesym` is used to allow HUGO symbols to be passed to `[`; `chrnum` identifies numerals or numeric constants as indices into the set of chromosomes (no chr prefix is allowed); `rsid` identifies dbSNP identifiers; `probeId` identifies a string as a microarray probe identifier.

`snpdepth` identifies a number that will be used as the number of chromosome-specific test results to be retained in any genome-wide screen

Slots

`.Data:` Object of class `"character"` ~~

Extends

Class `"character"`, from data part. Class `"vector"`, by class "character", distance 2. Class `characterORMIAME`, by class "character", distance 2.

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
showClass("genesym")
genesym("CPNE1")
```

`gwSnpScreenResult-class`
Class "gwSnpScreenResult" – containers for GGtools gwSnpScreen method outputs and allied objects

Description

Class "gwSnpScreenResult" – container for GGtools gwSnpScreen method outputs and allied objects

Objects from the Class

Objects can be created by calls of the form `new("gwSnpScreenResult", ...)`. These will be primarily lists of inference tables (snps are rows, columns are statistics and p-values). Additional slots manage analysis metadata.

gwSnpScreenResult is intended for genome-wide analysis of expression for a single gene.

cwSnpScreenResult is intended for the restriction to a single chromosome.

multiGwSnpScreenResult is intended for analyses with multiple genes.

Because the vast majority of tests are uninformative, early filtering is important for managing object sizes. Instances of filteredGwSnpScreenResult and filteredMultiGwSnpScreenResult are created when a `snpdepth` parameter is used with gwSnpTests.

Slots

`.Data`: Object of class `"list"` containing inference tables (snps are rows, columns are statistics and p-values)

`gene`: Object of class `"character"` typically the HUGO symbol of the gene analyzed

`psid`: Object of class `"character"` the feature identifier of the associated microarray

`annotation`: Object of class `"character"` vector of relevant annotation package identifier names were used

`formula`: Object of class `"formula"` the formula used to fit the model relating expression to genotype

Extends

Class `"list"`, from data part. Class `"vector"`, by class "list", distance 2. Class `AssayData`, by class "list", distance 2.

Methods

plot and show

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```
showClass("gwSnpScreenResult")
showClass("cwSnpScreenResult")
```

plot_EvG-methods *formal method for visualizing expression distributions vs genotype*

Description

boxplot expression vs genotype

Methods

gsym = "genesym", rsid = "rsid", sms = "smlSet" generates an annotated boxplot

multisnp methods plot_EvG2 allows specification of a second SNP rsid and shows boxplots over the cross-tabulation of the allele combinations

Examples

```
library(GGtools)
data(hmceuB36.2021)
plot_EvG(genesym("CPNE1"), rsid("rs6060535"), hmceuB36.2021)
```

smlSet-class *Documentation on S4 class "smlSet" an eSet-derived container for snpMatrix lists, allowing efficient combination of SNP chip genotyping with microarray expression data, and allied classes*

Description

Documentation on S4 class "smlSet" an eSet-derived container for snpMatrix lists, allowing efficient combination of SNP chip genotyping with microarray expression data, and allied classes

Objects from the Class

Objects can be created by calls of the form `new("smlSet", assayData, phenoData, featureData, experimentData, annotation, ...)`. These objects respond to interrogation on samples, expression values, SNP values, and other metadata.

Slots

smlEnv: Object of class `"environment"` an environment with single key `smList` pointing to a list of package snpMatrix snp.matrix instances

chromInds: Object of class `"numeric"` numeric vector indicating what chromosomes are represented in the `smlEnv`

snpNames: Object of class `"character"` list of character strings naming SNPs genotyped in smList element of smlEnv

organism: Object of class `"character"` informal, "Hs" recommended for human

assayData: Object of class `"AssayData"` intended to hold expression data coordinated with the smlEnv data

phenoData: Object of class `"AnnotatedDataFrame"` standard sample-level data container from eSet design

featureData: Object of class "AnnotatedDataFrame" standard feature-level metadata container, implied usage is for documenting the expresion data elements

experimentData: Object of class "MIAME" standard metadata container from Biobase eSet design

annotation: Object of class "character" vector giving the Bioconductor annotation package (.db type) for decoding expression feature identifiers.

.__classVersion__: Object of class "Versions" class version tracking metadata

Extends

Class eSet, directly. Class VersionedBiobase, by class "eSet", distance 2. Class Versioned, by class "eSet", distance 3.

Methods

smList `signature(x = "smlSet")`: retrieves the actual list of snp.matrix entities

smlEnv `signature(x = "smlSet")`: retrieves the environment holding snp.matrix entities

exprs `signature(x = "smlSet")`: retrieves the matrix of expression values

combine: concatenates expression data and forms intersection of SNP sets

getAlleles(smlSet, rsid): returns A/B notations for SNP determined by rsid

Note

We have included a `[` method for snp.matrix instances that accepts an rsid instance as a column selector.

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

See Also

GGtools package makes extensive use of these classes and methods.

Examples

```
showClass("smlSet")
```

---

snp.rhs.tests154 *Score tests with SNP genotypes as independent variable – taken from snpMatrix 1.5.4 while 1.5.5 version of this function is diminished in speed*

Description

This function fits a generalized linear model with phenotype as dependent variable and, optionally, one or more potential confounders of a phenotype-genotype association as independent variable. A series of SNPs (or small groups of SNPs) are then tested for additional association with phenotype. In order to protect against misspecification of the variance function, "robust" tests may be selected.

Usage

```
snp.rhs.tests154(formula, family = "binomial", link, weights, subset, data = par
   snp.data, tests=NULL, robust = FALSE,
   control=glm.test.control(maxit=20, epsilon=1.e-4, R2Max=0.98), allow.missing=
```

Arguments

`formula`	The base model formula, with phenotype as dependent variable
`family`	A string defining the generalized linear model family. This currently should (partially) match one of `"binomial"`, `"Poisson"`, `"Gaussian"` or `"gamma"` (case-insensitive)
`link`	A string defining the link function for the GLM. This currently should (partially) match one of `"logit"`, `"log"`, `"identity"` or `"inverse"`. The default action is to use the "canonical" link for the family selected
`data`	The dataframe in which the base model is to be fitted
`snp.data`	An object of class `"snp.matrix"` or `"X.snp.matrix"` containing the SNP data
`tests`	Either a vector of column names or numbers for the SNPs to be tested, or a list of short vectors defining groups of SNPs to be tested (again by name or number). The default action is to carry out *all* single SNP tests, but `single.snp.tests` will often achieve the same result much faster
`weights`	"Prior" weights in the generalized linear model
`subset`	Array defining the subset of rows of `data` to use
`robust`	If `TRUE`, robust tests will be carried out
`control`	An object giving parameters for the IRLS algorithm fitting of the base model and for the acceptable aliasing amongst new terms to be tested. See codeglm.test.control
`allow.missing`	The maximum proportion of SNP genotype that can be missing before it becomes necessary to refit the base model

Details

The tests used are asymptotic chi-squared tests based on the vector of first and second derivatives of the log-likelihood with respect to the parameters of the additional model. The "robust" form is a generalized score test in the sense discussed by Boos(1992). The "base" model is first fitted, and a score test is performed for addition of one or more SNP genotypes to the model. Homozygous SNP genotypes are coded 0 or 2 and heterozygous genotypes are coded 1. For SNPs on the X chromosome, males are coded as homozygous females. For X SNPs, it will often be appropriate to include sex of subject in the base model (this is not done automatically).

If a `data` argument is supplied, the `snp.data` and `data` objects are aligned by rowname. Otherwise all variables in the model formulae are assumed to be stored in the same order as the columns of the `snp.data` object.

Value

A data frame containing, for each SNP,

`Chi.squared`	The value of the chi-squared test statistic
`Df`	The corresponding degrees of freedom

`Df.residual` The residual degrees of freedom for the base model; *i.e.* the number of observations minus the number of parameters fitted

For the binomial family model, the base model can, in some circumstances, lead to perfect prediction of some observations (*i.e.* fitted probabilities of 0 or 1). These observations are ignored in subsequent calculations; in particular they are not counted in the residual degrees of freedom. Similarly for Poisson means fitted exactly to zero.

Note

A factor (or several factors) may be included as arguments to the function `strata(...)` in the `formula`. This fits all interactions of the factors so included, but leads to faster computation than fitting these in the normal way. Additionally, a `cluster(...)` call may be included in the base model formula. This identifies clusters of potentially correlated observations (e.g. for members of the same family); in this case, an appropriate robust estimate of the variance of the score test is used.

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

Boos, Dennis D. (1992) On generalized score tests. *The American Statistician*, **46**:327-333.

See Also

`single.snp.tests`, `snp.lhs.tests`, `snp.matrix-class`, `X.snp.matrix-class`

Examples

```
library(snpMatrix)
data(testdata)
slt3 <- snp.rhs.tests154(cc~strata(region), family="binomial",
   data=subject.data, snp.data= Autosomes, tests=1:10)
print(slt3)
```

---

snpLocs.Hs *SNP location accessor*

Description

SNP location accessor

Usage

```
snpLocs.Hs(cnum, rsid)
```

Arguments

`cnum`	chrnum instance indicating chromosome
`rsid`	rsid instance giving dbSNP ids for snps of interest

Details

The SNPlocs.Hsapiens.dbSNP.* package is curated by Bioconductor and maintains location and allele information on SNP.

The hsSnpLocs environment is available through data(hsSnpLocs), and contains a unified representation of the information in the SNPlocs package (which is only available through various chromosome-specific calls).

Value

two-row matrix – top row is numeric suffix of dbSNP ids, bottom row is location, genome-wide if chrnum is missing

Author(s)

Vince Carey <stvjc@channing.harvard.edu>

Examples

```
library(GGtools)
data(hmceuB36.2021)
nn21 = colnames(smList(hmceuB36.2021)[["21"]])
length(nn21)
ss = snpLocs.Hs(chrnum(21), rsid(nn21))
dim(ss)
# notes that the locations available are not completely
# congruent with those asserted in HapMap data
```

---

snpsNear *obtain list of rs numbers for snps near a gene*

Description

obtain list of rs numbers for snps near a gene

Usage

```
snpsNear(sym, radius=1e+05, chrnum)
```

Arguments

sym	instance of genesym class [e.g., use genesym(string) for gene 'string'], or of rsid class, or of numeric class
radius	number of base-pairs in each direction to look
chrnum	chrnum instance .. optional

Details

simple arithmetic based on output of snpLocs.Hs

Value

character vector of rsxxxxxx, dbSNP id, according to locations from SNPlocs.Hsapiens.dbSNP.20071016 package, as transferred to snpLocs.Hs resource in GGBase

Note

first invocation can take longer than subsequent, if snpLocs.Hs has not been invoked previously

Author(s)

Vince Carey <stvjc@channing.harvard.edu>

Examples

```
nearc = snpsNear(genesym("CPNE1"), 10000, chrnum(20))
library(GGtools)
data(hmceuB36.2021)
ss = smList(hmceuB36.2021)[[1]]
# following calculation requires new "[" for j an instance of rsid
clo = ss[ , rsid(snpsNear(rsid("rs6060535"), rad=1500, chrnum(20))) ]
clo
```

Index