

# GGBase

February 9, 2012

---

GGbase-package      *GGbase Package Overview*

---

## Description

GGbase Package Overview

## Details

This package provides infrastructure for programming related to the genetics of gene expression. The GGtools package makes use of classes and methods defined in this package. GGdata and hmyriB36 packages use the class structures defined in this package for serialized data.

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

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

## Author(s)

V. Carey

---

MAFfilter      *restrict SNP in an smlSet to range of minor allele frequencies (MAF)  
or genotype frequencies (GTF)*

---

## Description

restrict SNP in an smlSet to range of minor allele frequencies (MAF) or genotype frequencies

## Usage

```
MAFfilter(x, lower = 0, upper = 1)
GTFfilter(x, lower = 0)
```

**Arguments**

x	smlSet instance
lower	numeric lower bound on minor allele frequency or genotype frequency for keeping a SNP
upper	numeric upper bound on minor allele frequency for keeping a SNP

**Details**

uses `SnpMatrix-class` summary method from `snpStats`

**Value**

revised instance of `smlSet-class`

**Author(s)**

VJ Carey <stvjc@channing.harvard.edu>

**Examples**

```
data(smlSet.example)
sapply(smList(MAFfilter(smlSet.example, lower=.1)), dim)
sapply(smList(GTFfilter(smlSet.example, lower=.1)), dim)
```

---

`SessionInfo-class` *Class "SessionInfo" - objects to help stamp an output with information on session state*

---

**Description**

Class "SessionInfo" – objects to help stamp an output with information on session state

**Objects from the Class**

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

**Slots**

`.S3Class:` Object of class "character" simple cast to allow checking

**Extends**

Class "`oldClass`", by class "sessionInfo", distance 2.

**Methods**

No methods defined with class "SessionInfo" in the signature.

**Examples**

```
showClass("SessionInfo")
```

---

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", ...)`, or by special constructor functions. As of GGBase version 3.7.1, you can use `genesym(...)`, `chrnum(...)`, `probeId(...)`, `rsid(...)`. These generally just extend character or numeric so that vector operations are straightforward, but attach type information so that methods such as [ 'know' what they are getting.

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")
```

---

`featureFilter`      *remove unannotated or undesired features from an smlSet instance*

---

### Description

remove unannotated or undesired features from an smlSet instance

### Usage

```
featureFilter(x, requires = c("loc", "autosomal"))
```

**Arguments**

`x` instance of `smlSet` class

`requires` character vector – if "loc" is present, require that a non-NA value is present in CHRLOC for each feature; if "autosomal" is present, require that CHR value is in 1:22 (presently assumes human genome)

**Value**

revised `smlSet` instance excluding features no

**Author(s)**

VJ Carey

**Examples**

```
data(smlSet.example)
dim(exprs(smlSet.example))
fff = featureFilter(smlSet.example)
dim(exprs(fff))
```

---

<code>getSS</code>	<i>construct a small-footprint <code>smlSet</code> instance from a specially structured package</i>
--------------------	---

---

**Description**

construct a small-footprint `smlSet` instance from a specially structured package

**Usage**

```
getSS(packname, chrs, renameChrs = NULL, probesToKeep = NULL, wrapperEndo = NULL)
```

**Arguments**

`packname` string naming a package with `eset.rda` in data folder, defining `ExpressionSet` instance `ex`, and folder `parts` (from `inst` in pre-installed image) containing a collection of `rda` files holding `snpStats` `SnpMatrix-class` instances

`chrs` vector of strings of names of `SnpMatrix` instances to be included, typically these are the basenames of files in the `parts` folder, and correspond to chromosomes

`renameChrs` vector of strings of same length of `chrs` that will supply names to the elements of the `smList` component of the returned `smlSet`

`probesToKeep` vector of strings of probe names to be retained in the returned `smlSet`

`wrapperEndo` function receiving and returning an `smlSet` instance to be invoked prior to returning the `smlSet`

**Value**

an instance of `smlSet-class`

**Examples**

```
## Not run:
hm20 = getSS("GGdata", "20", renameChrs="chr20")

## End(Not run)

### The function is currently defined as
#function (packname, chrs, renameChrs = NULL, probesToKeep = NULL,
#         wrapperEndo = NULL)
#{
#   if (!is.null(renameChrs) && (length(chrs) != length(renameChrs)))
#     stop("renameChrs must have same length as chrs in call to getSS")
#   require(packname, character.only = TRUE)
#   ex = get(load(system.file(package = packname, "data/eset.rda")))
#   if (!is.null(probesToKeep))
#     ex = ex[probesToKeep, ]
#   partsfol = system.file("parts", package = packname)
#   chk = sapply(chrs, function(x) file.exists(paste(partsfol,
#     "/", x, ".rda", sep = "")))
#   if (!all(chk)) {
#     cat("requesting ", paste(chrs, ".rda", sep = ""))
#     cat(" but finding\n")
#     print(dir(partsfol))
#     stop("cannot retrieve requested SNP file.")
#   }
#   sml = lapply(chrs, function(x) get(load(paste(partsfol, "/",
#     x, ".rda", sep = ""))))
#   if (is.null(renameChrs))
#     names(sml) = chrs
#   else names(sml) = renameChrs
#   ans = make_smlSet(ex, sml, harmonizeSamples = TRUE)
#   if (is.null(wrapperEndo))
#     return(ans)
#   else {
#     ans = wrapperEndo(ans)
#     if (isTRUE(tst <- validObject(ans)))
#       return(ans)
#     stop(tst)
#   }
# }
```

---

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 `snppdepth` 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

`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")
```

---

make_smlSet	<i>create an smlSet instance from components</i>
-------------	--

---

### Description

create an smlSet instance from components

### Usage

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

### Arguments

es	ExpressionSet instance
sml	list of SnpMatrix instances
organism	string naming organism
harmonizeSamples	logical telling whether to intersect samples from expression and SNP data when sample sets do not coincide

### Details

combines SnpMatrix instances with expression data

### Value

instance of smlSet class

### Author(s)

VJ Carey <stvjc@channing.harvard.edu>

### Examples

```
data(smlSet.example) # here we just show the mechanics from a working smlSet  
es = as(smlSet.example, "ExpressionSet")  
sl = smList(smlSet.example)  
mm = make_smlSet(es, sl)  
validObject(mm)  
mm
```

---

```
multiCisTestResult-class
      Class "multiCisTestResult"
```

---

**Description**

object to contain results of restricted gene-centric searches for eQTL

**Objects from the Class**

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

**Slots**

`.Data`: Object of class "list" – list of results of `snprhs.tests`  
`conditions`: Object of class "list" – list of runtime conditions encountered  
`call`: Object of class "call" – for auditing, the call used is saved

**Extends**

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

**Methods**

**show** signature(object = "multiCisTestResult"):...

**Author(s)**

VJ Carey <stvjc@channing.harvard.edu>

**Examples**

```
showClass("multiCisTestResult")
```

---

```
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**

```
data(smlSet.example)
plot_EvG(genesym("WBP5"), rsid("rs10483083"), smlSet.example)
```

---

smlSet-class	<i>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</i>
--------------	--

---

## 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 `smlList` pointing to a list of package `snpStats` SnpMatrix instances

**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 expression 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

**smlList** signature(`x = "smlSet"`): retrieves the actual list of SnpMatrix entities

**smlEnv** signature(`x = "smlSet"`): retrieves the environment holding SnpMatrix entities

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

**snp** signature(`x = "smlSet"`, `chr = "chrnum"`): retrieves the raw matrix of genotype values (SnpMatrix instance from `snpStats` package)

**combine:** concatenates expression data and forms intersection of SNP sets

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

`coerce`: extracts `exprs`, `phenoData` and annotation and constructs `ExpressionSet`

`nsFilter`: apply `nsFilter` to expression component and rebind the genotype data after filtering

[ `signature(x = "smlSet", i = "ANY", j = "ANY", drop = "ANY")`: Quick methods for subsetting elements of `smlSets` have been provided.

If `X` is an `smlSet` instance and `G` is a vector of class `probeId-class`, then `X[G, ]` will reduce the expression data to the probes specified in `G`.

If `X` is an `smlSet` instance and `G` is a vector of class `chrnum-class`, then `X[G, ]` will reduce the SNP genotype data to the SNPs resident on chromosomes enumerated in `G`.

If `X` is an `smlSet` instance and `G` is a vector of class `rsid-class`, then `X[G, ]` will reduce the SNP genotype data to the SNPs enumerated in the dbSNP id in `G`.

### Note

We have included a [ method for `SnpMatrix` 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")
data(smlSet.example)
smlSet.example
validObject(smlSet.example)
# workout on expression components
dim(exprs(smlSet.example))
fn = featureNames(smlSet.example)[1:10]
fn
ss2 = smlSet.example[ probeId(fn), ] # restrict exprs to set of probes
dim(exprs(ss2))
# workout on SNP components
smList(smlSet.example)
dim(smList(ss2)[[1]])
ss2[ chrnum(21), ] # trivial restriction of SNP to a chromosome
sn = colnames(smList(ss2)[[1]])[1:20] # get some dbSNP ids
ss3 = ss2[ rsid(sn), ] # subset the snps
dim(smList(ss3)[[1]])
dim(smList(ss3)[["21"]]) # check names
ss3
as(snps(ss3, chrnum(21)), "character")[1:5,1:5] # generic codes
as(snps(ss3, chrnum(21)), "numeric")[1:5,1:5] # number copies of B
as(snps(ss3, chrnum(21)), "matrix")[1:5,1:5] # raw
```

---

smlSummary	<i>class and function to summarize frequency information on genotypes in an smlSet</i>
------------	--

---

**Description**

generates information on sample size, minor allele frequency, specific call frequencies, and HWE test results on all SNP in an smlSet

**Usage**

```
smlSummary(x)
```

**Arguments**

x                   instance of `smlSet-class`

**Details**

to control volume of printout a simple list extending class is defined for show method

**Value**

Instance of smlSummary class, which simply extends list. Each list element is a matrix of results provided by `summary, SnpMatrix-method`.

**Author(s)**

VJCarey <stvjc@channing.harvard.edu>

**Examples**

```
data(smlSet.example)
smlSummary(smlSet.example)
```

# Index

## \*Topic classes

genesym-class, 3  
gwSnpScreenResult-class, 5  
multiCisTestResult-class, 8  
SessionInfo-class, 2  
smlSet-class, 9

## \*Topic methods

plot\_EvG-methods, 8

## \*Topic models

featureFilter, 3  
getSS, 4  
MAFfilter, 1  
make\_smlSet, 7  
smlSummary, 11

## \*Topic package

GGbase-package, 1  
[, SnpMatrix, ANY, rsid, ANY-method  
(smlSet-class), 9  
[, cwSnpScreenResult, ANY, ANY, ANY-method  
(gwSnpScreenResult-class),  
5  
[, gwSnpScreenResult, ANY, ANY, ANY-method  
(gwSnpScreenResult-class),  
5  
[, smlSet, ANY, ANY, ANY-method  
(smlSet-class), 9  
[, smlSet-method (smlSet-class), 9

AssayData, 6, 8

character, 3  
characterORMIAME, 3  
chrnum (genesym-class), 3  
chrnum, character-method  
(genesym-class), 3  
chrnum, numeric-method  
(genesym-class), 3  
chrnum-class, 10  
chrnum-class (genesym-class), 3  
cnumOrMissing (genesym-class), 3  
cnumOrMissing-class  
(genesym-class), 3  
coerce, smlSet, ExpressionSet-method  
(smlSet-class), 9

combine, filteredMultiGwSnpScreenResult, filteredMultiGwSnpScreenResult  
(gwSnpScreenResult-class),  
5

combine, multiGwSnpScreenResult, multiGwSnpScreenResult  
(gwSnpScreenResult-class),  
5

combine, smlSet, smlSet-method  
(smlSet-class), 9

cwSnpScreenResult-class  
(gwSnpScreenResult-class),  
5

eSet, 9

exprs, smlSet-method  
(smlSet-class), 9

featureFilter, 3  
filteredGwSnpScreenResult-class  
(gwSnpScreenResult-class),  
5

filteredMultiGwSnpScreenResult-class  
(gwSnpScreenResult-class),  
5

genesym (genesym-class), 3  
genesym, character-method  
(genesym-class), 3

genesym-class, 3  
getAlleles (smlSet-class), 9  
getAlleles, smlSet, rsid-method  
(smlSet-class), 9

getSS, 4  
GGbase (GGbase-package), 1

GGbase-package, 1  
GTFfilter (MAFfilter), 1  
gwSnpScreenResult  
(gwSnpScreenResult-class),  
5

gwSnpScreenResult-class, 5

list, 6, 8

MAFfilter, 1

make\_smlSet, 7

multiCisTestResult-class, 8

- multiGwSnpScreenResult-class
  - (*gwSnpScreenResult-class*), 5
- nsFilter, 10
- nsFilter, smlSet-method
  - (*smlSet-class*), 9
- oldClass, 2
- phenoVar (*genesym-class*), 3
- phenoVar, character-method
  - (*genesym-class*), 3
- phenoVar-class (*genesym-class*), 3
- plot, cwSnpScreenResult, ANY-method
  - (*gwSnpScreenResult-class*), 5
- plot, cwSnpScreenResult, character-method
  - (*gwSnpScreenResult-class*), 5
- plot, cwSnpScreenResult, missing-method
  - (*gwSnpScreenResult-class*), 5
- plot, gwSnpScreenResult, ANY-method
  - (*gwSnpScreenResult-class*), 5
- plot, multiGwSnpScreenResult, ANY-method
  - (*gwSnpScreenResult-class*), 5
- plot\_EvG (*plot\_EvG-methods*), 8
- plot\_EvG, genesym, rsid, smlSet-method
  - (*plot\_EvG-methods*), 8
- plot\_EvG, probeId, rsid, smlSet-method
  - (*plot\_EvG-methods*), 8
- plot\_EvG-methods, 8
- plot\_EvG2 (*plot\_EvG-methods*), 8
- plot\_EvG2, genesym, rsid, rsid, smlSet-method
  - (*plot\_EvG-methods*), 8
- plot\_EvG2, probeId, rsid, rsid, smlSet-method
  - (*plot\_EvG-methods*), 8
- probeId (*genesym-class*), 3
- probeId, character-method
  - (*genesym-class*), 3
- probeId-class, 10
- probeId-class (*genesym-class*), 3
- rsid (*genesym-class*), 3
- rsid, character-method
  - (*genesym-class*), 3
- rsid, numeric-method
  - (*genesym-class*), 3
- rsid-class, 10
- rsid-class (*genesym-class*), 3
- SessionInfo-class, 2
- show, chrnum-method
  - (*genesym-class*), 3
- show, cwSnpScreenResult-method
  - (*gwSnpScreenResult-class*), 5
- show, filteredGwSnpScreenResult-method
  - (*gwSnpScreenResult-class*), 5
- show, filteredMultiGwSnpScreenResult-method
  - (*gwSnpScreenResult-class*), 5
- show, gwSnpScreenResult-method
  - (*gwSnpScreenResult-class*), 5
- show, multiCisTestResult-method
  - (*multiCisTestResult-class*), 8
- show, multiGwSnpScreenResult-method
  - (*gwSnpScreenResult-class*), 5
- show, rsid-method (*genesym-class*), 3
- show, smlSet-method
  - (*smlSet-class*), 9
- show, smlSummary-method
  - (*smlSummary*), 11
- smlEnv (*smlSet-class*), 9
- smlEnv, smlSet-method
  - (*smlSet-class*), 9
- smList (*smlSet-class*), 9
- smList, smlSet-method
  - (*smlSet-class*), 9
- smlSet (*smlSet-class*), 9
- smlSet-class, 2, 4, 11
- smlSet-class, 9
- smlSet.example (*smlSet-class*), 9
- smlSummary, 11
- smlSummary-class (*smlSummary*), 11
- snp.rhs.tests, 8
- snpdepth (*genesym-class*), 3
- snpdepth-class (*genesym-class*), 3
- SnpMatrix-class, 2, 4
- snpNames (*smlSet-class*), 9
- snpNames, smlSet, chrnum-method
  - (*smlSet-class*), 9
- snpNames, smlSet, missing-method
  - (*smlSet-class*), 9
- snps (*smlSet-class*), 9
- snps, smlSet, chrnum-method
  - (*smlSet-class*), 9
- summary, SnpMatrix-method, 11

updateObject, smlSet-method  
(*smlSet-class*), 9

vector, 3, 6, 8

Versioned, 9

VersionedBiobase, 9