

# PGSEA

April 20, 2011

---

`aggregateExprs`      *Aggregate expression data*

---

## Description

This function removes duplicates row names from an expression set, summarizing them with a function of the users choice. The "absMax" function located in package "reb" we have found to be useful.

## Usage

```
aggregateExprs(x, package = "hgu133plus2", using = "ENTREZID", FUN, ...)
```

## Arguments

<code>x</code>	expression data - matrix, eSet, or ExpressionSet
<code>package</code>	annotation package of expression data
<code>using</code>	format type that gene IDs are converted to
<code>FUN</code>	function by which to summarize duplicated values
<code>...</code>	extra parameters passed on to FUN

## Value

A matrix of expression data with the rows aggregated to a unique format chosen by the user. The new identifiers of the returned matrix are those specified with the "using" argument. To see possible values, use the `ls()` command illustrated below in the examples.

## Author(s)

Kyle Furge <kyle.furge@vai.org> and Karl Dykema <karl.dykema@vai.org>

**Examples**

```

if (require(hgu95av2.db) & require(annaffy)) {
  library(annaffy)
  data(aafExpr)
  class(exprs(aafExpr))
  exprs(aafExpr)[1:4, 1:4]

  #list possible values for the "using" argument
  ls(pos=which(search()=="package:hgu95av2.db"))

  convert <- aggregateExprs(exprs(aafExpr), "hgu95av2.db", FUN=mean, na.rm=TRUE)
  convert[1:4, 1:4]
}

```

---

 convertSmc

---

*Convert Entrez ID based "smc" object*


---

**Description**

This function will convert the Entrez IDs of an smc object to the corresponding Entrez IDs from a different species. Data from the homogene project is downloaded and used within this function.

**Usage**

```
convertSmc(mcs, fromSpecies = "h", toSpecies = "r", hgX="./homologene.data")
```

**Arguments**

mcs	a list of "smc" objects
fromSpecies	character - a single letter describing the species to convert from ie, h=human, r=rat, etc..
toSpecies	character - a single letter describing the species to convert to ie, h=human, r=rat, etc..
hgX	character - file name of homologene data file

**Details**

This function will not work if you have not downloaded the homologene data file. Please use this command to do so: `download.file("ftp://ftp.ncbi.nih.gov/pub/HomoloGene/current/homologene.data", destfile="homologene.data")`

**Value**

a list of converted "smc" objects

**Author(s)**

Karl Dykema <karl.dykema@vai.org>

## Examples

```
## Not run:
download.file("ftp://ftp.ncbi.nih.gov/pub/HomoloGene/current/homologene.data", destfile="h

datadir <- system.file("data", package = "PGSEA")
sample <- readGmt(file.path(datadir, "sample.gmt"))

converted <- convertSmc(sample[1:2], "h", "r")
str(converted)

## End(Not run)
```

---

editSmc

*Edit "smc" objects*

---

## Description

This function will edit a single or list of "smc" objects.

## Usage

```
editSmc(smcList, attName = "creator", newAtt = "changed!!")
```

## Arguments

smcList	a list of "smc" objects
attName	character - which slot to change
newAtt	character - what to change the slot to

## Value

a list of edited "smc" objects

## Author(s)

Karl Dykema <karl.dykema@vai.org>

## Examples

```
datadir <- system.file("data", package = "PGSEA")
sample <- readGmt(file.path(datadir, "sample.gmt"))
str(sample[1:2])

temp <- editSmc(sample[1:2], "creator", "Joe Smith")

str(temp)
```

go2smc

*Gene Ontology 2 "smc"*

---

**Description**

This function creates "smc" objects from the "GO" Bioconductor library.

**Usage**

```
go2smc(min = 50, max = 200, organism="human")
```

**Arguments**

min	numeric - minimum length of ids to be included
max	numeric - maximum length of ids to be included
organism	character - organism

**Value**

a list of "smc" objects

**Author(s)**

Karl Dykema <karl.dykema@vai.org>

**Examples**

```
if(require(GO)){  
  mcs <- go2smc()[1:2]  
  str(mcs)  
}
```

---

GOLUBmcs*Molecular Concepts prepared at VAI from data created by Golub et al.*

---

**Description**

386 molecular concepts generated at VAI. The data these concepts were generate from is available from <http://www.broad.mit.edu/cmap/>.

**Usage**

```
data(GOLUBmcs)
```

**Format**

a list of "smc" objects

## Details

These concepts were generated using the limma BioConductor package. The code used for generation of these concepts is available upon request.

## Source

<http://www.broad.mit.edu/cmap/>

## Examples

```
data(GOLUBmcs)
str(GOLUBmcs[1:4])
```

---

kegg2smc	<i>KEGG pathway to "smc"</i>
----------	------------------------------

---

## Description

This function creates "smc" objects from the "KEGG" Bioconductor library.

## Usage

```
smc <- kegg2smc(min = 1, max = 284, organism="human")
```

## Arguments

min	numeric - minimum length of ids to be included
max	numeric - maximum length of ids to be included
organism	character - organism

## Value

a list of "smc" objects

## Author(s)

Karl Dykema <karl.dykema@vai.org> and Richard Birnie <richard.birnie@pro-curetherapeutics.com>

## Examples

```
if(require(KEGG)){
mcs <- kegg2smc(min=20,max=284)
length(mcs)
str(mcs[[1]])
}
```

---

nbEset

*Reduced Neuroblastoma data set*

---

### Description

Neuroblastoma Data set - reduced in size to comply with BioC package guidelines

### Usage

```
data(nbEset)
```

### Details

This dataset was retrieved from GEO <http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE3960> and consists of five reference samples, and ten primary neuroblastoma tumors. Four of the five reference samples GSM2827, GSM2842, GSM2883, and GSM2895 came from a separate dataset, [http://www.ncbi.nlm.nih.gov/geo/gds/gds\\_browse.cgi?gds=181](http://www.ncbi.nlm.nih.gov/geo/gds/gds_browse.cgi?gds=181)

### Source

<http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE3960>

### Examples

```
data(nbEset)
nbEset
```

---

PGSEA

*Parametric Gene Set Enrichment Analysis*

---

### Description

This package contains functions for parametric analysis of gene expression data. This type of analysis can assist in determining of lists of genes, such as those deregulated in defined experimental systems, are similarly deregulated in other data sets.

This function subsets the data based on lists of genes, computes a summary statistic for each gene list, and returns the results in a convenient form.

### Usage

```
PGSEA(exprs, cl, range = c(25, 500), ref = NULL, center = TRUE, p.value = 0.005,
```

**Arguments**

<code>exprs</code>	matrix expression data, a numeric matrix, <code>eSet</code> , or <code>ExpressionSet</code>
<code>cl</code>	gene set list - "GeneSetCollection" or list of "SMC" objects
<code>range</code>	a 2 element vector describing the min and max length of concepts to analyze
<code>enforceRange</code>	boolean - if TRUE, the expression matrix must contain data for the proper number of genes as set by the range argument to return a significant result. (this argument is used for data that contains NA's...)
<code>ref</code>	a vector containing the index of reference samples from which to make comparisons. Defaults to NULL (internally referenced samples)
<code>center</code>	boolean - median center gene expression matrix columns prior to analysis. Can be helpful if 'ref' is used
<code>p.value</code>	numeric p.value threshold or NA to return all data or TRUE to return a matrix of p.values
<code>weighted</code>	boolean - weight results by the size of each gene list
<code>...</code>	extra arguments passed along to FUN

**Details**

Gene expression values are separated into subsets based on the lists of genes contained in the `cl` argument. This can be a "GeneSetCollection" or a list of "SMC" (Simple Molecular Concept) objects. For example, `readGmt` can be used to produce a 'smc' object list from a simple tab-delimited text file. The gene expression values from each of these gene lists is extracted and a summary statistic is computed for each subset (or region in the case of chromosomal bands/arms).

The expression data must have the same identifiers as the list of genes being tested. If they are not, the expression data can be converted using the `aggregateExprs` function, that can use a current annotation environment to convert and condense the gene expression data.

By default the method set out by Kim and Volsky <http://www.biomedcentral.com/1471-2105/6/144> is applied to the gene set. If `weighted==FALSE` than the default `t.test` function is used.

The function is set up to perform the analysis on individual samples. For convenient method to analyze groups of samples, see the "Limma User's Guide" for more information on how to see up a contrast matrix and perform a linear model fit. The coefficients of the fit can then be used a input into the `PGSEA` function.

**Value**

If `p.value` is set to a number, a matrix of results that pass at that significance is returned, of size `<number of samples> x <number of molecular concepts>`.

If `p.value` is set to NA, all results are returned.

If `p.value` is set to TRUE, then a list is returned that consists of the `PGSEA` results as well as their `p.values`.

**Note**

<http://www.biomedcentral.com/1471-2105/6/144>

**Author(s)**

Kim SY, Volsky DJ., [kyle.furge@vai.org](mailto:kyle.furge@vai.org) and [karl.dykema@vai.org](mailto:karl.dykema@vai.org)

## References

PGSEA: Parametric Analysis of Gene Set Enrichment

## Examples

```
datadir <- system.file("data", package = "PGSEA")
sample <- readGmt(file.path(datadir, "sample.gmt"))
data(nbEset)
pg <- PGSEA(nbEset, cl=sample, ref=1:5)

print(pg[, -c(1:5)])
```

---

readGmt

*readGmt*

---

## Description

This function will read a "gmt" file into R, returning results as a list of SMC objects.

## Usage

```
readGmt (fname)
```

## Arguments

fname                      File name of concepts in .gmt format

## Details

The .gmt file format is a tab delimited file format used to store gene lists. These gene lists are stored row by row. The first column is the gene set name. The second column is a brief description, and every entry after that is a gene within that gene set.

## Value

A list of SMC objects

## Author(s)

Karl Dykema <karl.dykema@vai.org>

## References

[http://www.broad.mit.edu/gsea/doc/data\\_formats.html#gmt](http://www.broad.mit.edu/gsea/doc/data_formats.html#gmt)

## See Also

[writeGmt](#)



## Examples

```
datadir <- system.file("data", package = "PGSEA")
sample <- readGmt(file.path(datadir, "sample.gmt"))
str(sample)
```

---

readSmc	<i>Read SMC files</i>
---------	-----------------------

---

## Description

This function reads in SMCs (simple molecular concepts) from individual text files.

## Usage

```
readSmc(files)
```

## Arguments

`files` a character vector of file names

## Value

A list of SMC objects

## Author(s)

Kyle Furge <kyle.furge@vai.org> and Karl Dykema <karl.dykema@vai.org>

## References

??

## See Also

[writeSmc](#)

## Examples

```
datadir <- system.file("data", package = "PGSEA")
sample <- readGmt(file.path(datadir, "sample.gmt"))
str(sample)
sample[[1]]@reference <- "fileName"

## Not run:
writeSmc(sample[[1]])
smc <- readSmc("fileName-0.txt")

## End(Not run)
```

---

`scanSmc`*Scan through smc objects*

---

**Description**

This function scans through smc objects and returns those with specified attributes.

**Usage**

```
scanSmc(smcList, scanSlot = "private", scanFor = "no")
```

**Arguments**

<code>smcList</code>	list of "smc" objects
<code>scanSlot</code>	character - which smc slot to investigate
<code>scanFor</code>	character - what character string to look for

**Value**

a list of "smc" objects with the desired attribute

**Author(s)**

Karl Dykema <karl.dykema@vai.org>

**Examples**

```
datadir <- system.file("data", package = "PGSEA")
sample <- readGmt(file.path(datadir, "sample.gmt"))
sample[1:2] <- editSmc(sample[1:2], "creator", "Joe Smith")

scanned <- scanSmc(sample, "creator", "Joe Smith")

str(scanned)
```

---

`smcPlot`*Plot PGSEA results*

---

**Description**

This basic function will plot results from PGSEA with easy altering of margins, colors, and text.

**Usage**

```
smcPlot(m, ff = NULL, skip = "NO", scale = c(-3, 3), na.color = par("bg"), margin
```

**Arguments**

<code>m</code>	matrix - your results from PGSEA (or any other numeric matrix of data)
<code>ff</code>	factor - this factor corresponds to the subtypes of your samples and will control the column names
<code>skip</code>	character - which subtype(s) to skip from "ff"
<code>scale</code>	vector, length 2 - this vector sets the minimum and maximum values for the graph scale (at bottom of plot)
<code>na.color</code>	character - color to display in the result of an NA
<code>margins</code>	vector, length 4 - this vector gives the expansion values for the margins
<code>r.cex</code>	numeric - number giving the amount by which row names should be scaled relative to the default
<code>c.cex</code>	numeric - number giving the amount by which column names should be scaled relative to the default
<code>show.grid</code>	boolean - show grid outlines within plot?
<code>cnames</code>	boolean or character - vector of alternative column names
<code>rnames</code>	boolean or character - vector of alternative row names
<code>grid.lty</code>	numeric - line type of the grid lines
<code>clust</code>	boolean - want to cluster?
<code>...</code>	additional graphical parameters passed along to the plotting function

**Author(s)**

Karl Dykema <karl.dykema@vai.org>

**Examples**

```
library(PGSEA)
datadir <- system.file("data", package = "PGSEA")
sample <- readGMT(file.path(datadir, "sample.gmt"))
data(nbEset)

pg <- PGSEA(nbEset, cl=sample, ref=1:5)
sub <- factor(c(rep(NA, 5), rep("NeuroB", 5), rep("NeuroB_MYC+", 5)))

smcPlot(pg, sub, scale=c(-10, 10), col=.rwb, margins=c(1, 1, 8, 13))
```

**Description**

A few gene sets compiled at VAI. We have found useful in our analysis.

**Usage**

```
data(VAIgsc)
```

**Format**

The format is: chr "VAIgsc"

**Source**

Various sources... See individual objects for PMID, GEO accession, etc..

**Examples**

```
data(VAIgsc)
summary(VAIgsc)
details(VAIgsc[[1]])
```

---

VAImcs

*Molecular Concepts prepared at VAI*

---

**Description**

A few gene sets compiled at VAI. We have found useful in our analysis.

**Usage**

```
data(VAImcs)
```

**Format**

a list of "smc" objects

**Source**

See individual concepts for PMID or other source information.

**Examples**

```
data(VAImcs)
str(VAImcs)
```

---

`writeGmt`*writeGmt*

---

## Description

This function writes out SMC objects into .gmt file format

## Usage

```
writeGmt(fname, cl)
```

## Arguments

<code>fname</code>	name of the file to be written out
<code>cl</code>	list of SMC objects

## Details

The .gmt file format is a tab delimited file format used to store gene lists. These gene lists are stored row by row. The first column is the gene set name. The second column is a brief description, and every entry after that is a gene within that gene set.

## Author(s)

Kyle Furge <kyle.furge@vai.org> and Karl Dykema <karl.dykema@vai.org>

## References

[http://www.broad.mit.edu/gsea/doc/data\\_formats.html#gmt](http://www.broad.mit.edu/gsea/doc/data_formats.html#gmt)

## See Also

[readGmt](#)

## Examples

```
datadir <- system.file("data", package = "PGSEA")
sample <- readGmt(file.path(datadir, "sample.gmt"))
str(sample)

## Not run:
writeGmt(paste(datadir, "/output.gmt", sep=""), sample)

## End(Not run)
```

writeSmc

*writeSmc*

---

**Description**

This function will write out SMC objects to individual text files

**Usage**

```
writeSmc(x)
```

**Arguments**

x                    an object of class SMC

**Details**

The file name is determined by the reference slot of the SMC object.

**Author(s)**

Kyle Furge <kyle.furge@vai.org> and Karl Dykema <karl.dykema@vai.org>

**See Also**

[writeSmc](#)

**Examples**

```
datadir <- system.file("data", package = "PGSEA")
sample <- readGmt(file.path(datadir, "sample.gmt"))
str(sample)
sample[[1]]@reference <- "fileName"

## Not run:
writeSmc(sample[[1]])

## End(Not run)
```

# Index

## \*Topic **datasets**

aggregateExprs, 1  
GOLUBmcs, 4  
nbEset, 6  
VAIgsc, 11  
VAImcs, 12

## \*Topic **data**

readGmt, 8  
readSmc, 9  
writeGmt, 13  
writeSmc, 14

## \*Topic **hplot**

smcPlot, 10

## \*Topic **htest**

PGSEA, 6

## \*Topic **manip**

convertSmc, 2  
editSmc, 3  
go2smc, 4  
kegg2smc, 5  
scanSmc, 10

aggregateExprs, 1

convertSmc, 2

editSmc, 3

go2smc, 4

GOLUBmcs, 4

kegg2smc, 5

nbEset, 6

PGSEA, 6

readGmt, 8, 13

readSmc, 9

scanSmc, 10

smc-class (*readSmc*), 9

smcPlot, 10

VAIgsc, 11

VAImcs, 12

writeGmt, 8, 13

writeSmc, 9, 14, 14