# Package: goProfiles (via r-universe)

October 9, 2026

**Version** 1.75.0

**Title** goProfiles: an R package for the statistical analysis of functional profiles

**Author** Alex Sanchez, Jordi Ocana and Miquel Salicru

**Type** Package

**Depends** Biobase, AnnotationDbi, GO.db, CompQuadForm, stringr

**Suggests** org.Hs.eg.db

**Date** 2018-04-20

**Maintainer** Alex Sanchez <asanchez@ub.edu>

**Description** The package implements methods to compare lists of genes based on comparing the corresponding 'functional profiles'.

**License** GPL-2

**biocViews** Annotation, GO, GeneExpression, GeneSetEnrichment, GraphAndNetwork, Microarray, MultipleComparison, Pathways, Software

**url** https://github.com/alexsanchezpla/goProfiles/wiki

**NeedsCompilation** no

**RoxygenNote** 6.0.1

**Config/pak/sysreqs** libicu-dev libpng-dev libssl-dev zlib1g-dev

**Repository** https://bioc.r-universe.dev

**Date/Publication** 2026-04-28 12:32:52 UTC

**RemoteUrl** https://github.com/bioc/goProfiles

**RemoteRef** HEAD

**RemoteSha** 8569caa971224ac79c322f7c854486847c011645

## Contents

goProfiles-package . . . . . 2
basicProfile . . . . . 3

CD4Ids . . . 4
clustKidneyMF2 . . . 5
compareGeneLists . . . 5
compareGOProfiles . . . 7
compareProfilesLists . . . 9
compSummary . . . 11
contractedProfile . . . 12
conversionFunctions . . . 13
drosophila . . . 14
equivalentGOProfiles . . . 15
equivClust . . . 17
equivClust2pdf . . . 19
equivSummary . . . 20
expandedLevel . . . 21
expandedProfile . . . 22
fisherGOProfiles . . . 23
fitGOProfile . . . 26
GOTermsList . . . 28
hugoIds . . . 29
iterEquivClust . . . 30
kidneyGeneLists . . . 31
mergeProfilesLists . . . 32
ngenes . . . 33
omimIds . . . 34
plotProfiles . . . 35
printProfiles . . . 37
prostateIds . . . 38

**Index** **39**

---

`goProfiles-package` *Performs Gene Ontology based analysis using Functional Profiles.*

---

## Description

Performs Gene Ontology based analysis for gene sets or other type of biological identifiers which can be annotated in the Gene Ontology.

## Details

| | |
|---|---|
| Package: | goProfiles |
| Type: | Package |
| Version: | 1.33.4 |
| Date: | 2016-03-29 |
| License: | GPL |

### Author(s)

Alex Sanchez and Jordi Ocana

### References

Sanchez-Pla, A., Salicru M. and J.Ocana. Statistical methods for the analysis of high-throughput data based on functional profiles derived from the gene ontology. Journal of Statistical Planning and Inference, 2007.

Salicru, M., Ocana, J. Sanchez-Pla, A. Comparison of Gene Lists based on Functional Profiles. BMC Bioinformatics, DOI: 10.1186/1471-2105-12-401, 2011.

### See Also

goTools, GOstats, topGO, and other Bioconductor packages for GO based analysis

---

## basicProfile *Builds basic functional profile*

---

### Description

Compute basic functional profile for a given list of genes/GO identifiers, a given ontology at a given level of the GO

### Usage

```
basicProfile(genelist, idType = "Entrez", onto = "ANY", level = 2, orgPackage=NULL,
anotPackage=NULL,  ord = TRUE, multilevels = NULL,
empty.cats = TRUE, cat.names = TRUE, na.rm = TRUE)
```

### Arguments

| | |
|---|---|
| genelist | List of genes on which the Profile has to be based |
| idType | Type of identifiers for the genes. May be 'Entrez' (default), BiocProbes or GoTermsFrame (see details below). |
| onto | Ontology on which the profile has to be built |
| level | Level of the ontology at which the profile has to be built |
| orgPackage | Name of a Bioconductor's organism annotations package ('org.Xx-eg-db'). This field must be provided if the gene list passed to the function is either a character vector of 'Entrez' (NCBI) identifiers or a character vector of probe names |
| anotPackage | Name of Bioconductor's microarray annotations package. This field must be provided if the gene list passed to the function is a character vector of probe names |
| ord | Set to 'TRUE' if the profile has to appear ordered by the category names |
| multilevels | If it is not NULL it must be a vector of GO categories that defines the level at where the profile is built |

| | |
|---|---|
| empty.cats | Set to 'TRUE' if empty categories should appear in the profile |
| cat.names | Set to 'TRUE' if the profile has to contain the names of categories |
| na.rm | Set to 'TRUE' if NAs should be removed |

## Details

The function admits three types of entries: Entrez ('Entrez'), Bioconductor probe set names ('BioCprobes') or a special type of data frames ('GOTermsFrames'). If the identifier type are 'BioCprobes' then an annotation package name must be provided too.

## Value

An object of class GOProfile (one or more data frames in a list named by the ontologies)

## Author(s)

Alex Sanchez

## References

Sanchez-Pla, A., Salicru, M. and Ocana, J. Statistical methods for the analysis of high-throughput data based on functional profiles derived from the gene ontology. Journal of Statistical Planning and Inference, Volume 137, Issue 12, Pages 3975-3989, 2007

## See Also

expandedProfile

## Examples

```
data(CD4Ids)
CD4.MF.Profiles <-basicProfile(genelist=CD4LLids, onto='MF', level=2, orgPackage="org.Hs.eg.db")
print(CD4.MF.Profiles)
```

---

# CD4Ids *Entrez identifiers for CD4-TCells example*

---

## Description

This dataset contains the entrez identifiers CD4EntrezIds and their associated GO Terms CD4GOTermsFrame and CD4GOTermsList corresponding to the list of differentially expressed genes in a study by Henkel et al.

## Usage

```
data(CD4Ids)
```

## Source

Hengel, R.L. and Thaker, V. and Pavlick, M.V. and Metcalf, J.A. and Dennis, G. Jr. and Yang, J. and Lempicki, R.A. and Sereti, I. and Lane, H.C. (2003). L-selectin (CD62L) expression distinguishes small resting memory CD4+ T cells that preferentially respond to recall antigen. J. Immunol., 170, 28-32. (2003)

## Examples

```
data(CD4Ids)
```

---

# clustKidneyMF2 *Ready 2 cluster equivalence distance matrix obtained from the analysis of the "Kidney Dataset" at level 2 of the MF ontology*

---

## Description

An object of class "list" with the information outputted from of the analysis performed by function "profileEquiv_topDown2"

## Usage

```
clustKidneyMF2
```

## Format

An object of class `equivClust` (inherits from `hclust`) of length 7.

---

# compareGeneLists *Compares two lists of genes by building (expanded) profiles and comparing them*

---

## Description

This function wraps all the needed steps to compare two lists of genes following the methodology developed by Sanchez, Salicru and Ocan\~a (2007)

## Usage

```
compareGeneLists(genelist1, genelist2, idType = "Entrez", onto = "ANY",
level = 2, orgPackage, method = "lcombChisq", ab.approx = "asymptotic",
confidence = 0.95, compareFunction="compareGOProfiles", ...)
```

## Arguments

| | |
|---|---|
| genelist1 | First gene set to be compared |
| genelist2 | Second gene set to be compared |
| idType | Type of identifiers for the genes. May be 'Entrez' (default), BiocProbes or GoTermsFrame. See the 'Details' section below |
| onto | Ontology on which the profile has to be built |
| level | Level of the ontology at which the profile has to be built |
| orgPackage | Name of a Bioconductor's organism annotations package ('org.Xx-eg-db') |
| method | The approximation method to the sampling distribution under the null hypothesis specifying that the samples pn and qm come from the same population. See the 'Details' section below |
| confidence | The confidence level of the confidence interval in the result |
| ab.approx | The approximation used for computing 'a' and 'b' coefficients (see details) |
| compareFunction | Allows to use 'fitGOProfile' (sample vs population) or 'compareGOProfiles' (sample1 vs sample2) |
| ... | Other arguments for the methods 'basicProfile' or 'compareGoProfiles' |

## Value

The result of the comparison is a list with a variable number of arguments, depending for which ontologies has been performed the comparison. Each list member is an object of class 'htest' corresponding to the output of the function `compareGOProfiles`

## Author(s)

Alex Sanchez

## References

Sanchez-Pla, A., Salicru, M. and Ocana, J. Statistical methods for the analysis of high-throughput data based on functional profiles derived from the gene ontology. Journal of Statistical Planning and Inference, 2007

## See Also

compareGOProfiles, basicProfile

## Examples

```
data(prostateIds)
prostateCompared<- compareGeneLists (welsh01EntrezIDs[1:500],
singh01EntrezIDs[1:500], level=2, onto='MF', orgPackage="org.Hs.eg.db")
print(prostateCompared)
# print(compSummary(prostateCompared))
```

compareGOProfiles *Comparison of lists of genes through their functional profiles*

## Description

Compare two samples of genes in terms of their GO profiles pn and qm. Both samples may share a common subsample of genes, with GO profile pqn0. 'compareGOProfiles' implements some inferential procedures based on asymptotic properties of the squared euclidean distance between the contracted versions of pn and qm

## Usage

```
compareGOProfiles(pn, qm = NULL, pqn0 = NULL, n = ngenes(pn), m = ngenes(qm),
n0 = ngenes(pqn0), method = "lcombChisq", ab.approx = "asymptotic",
confidence = 0.95, simplify = T, ...)
```

## Arguments

| | |
|---|---|
| pn | an object of class ExpandedGOProfile representing one or more "sample" expanded GO profiles for a fixed ontology (see the 'Details' section) |
| qm | an object of class ExpandedGOProfile representing one or more "sample" expanded GO profiles for a fixed ontology (see the 'Details' section) |
| pqn0 | an object of class ExpandedGOProfile representing one or more "sample" expanded GO profiles for a fixed ontology (see the 'Details' section) |
| n | a numeric vector with the number of genes profiled in each column of pn. This parameter is included to allow the possibility of exploring the consequences of varying sample sizes, other than the true sample size in pn. |
| m | a numeric vector with the number of genes profiled in each column of qm. |
| n0 | a numeric vector with the number of genes profiled in each column of pqn0. |
| method | the approximation method to the sampling distribution under the null hypothesis specifying that the samples pn and qm come from the same population. See the 'Details' section below |
| confidence | the confidence level of the confidence interval in the result |
| ab.approx | the approximation used for computing 'a' and 'b' coefficients (see details) |
| simplify | should the result be simplified, if possible? See the 'Details' section |
| ... | Other arguments needed |

## Details

An object of S3 class 'ExpandedGOProfile' is, essentially, a 'data.frame' object with each column representing the relative frequencies in all observed node combinations, resulting from profiling a set of genes, for a given and fixed ontology. The row.names attribute codifies the node combinations and each data.frame column (say, each profile) has an attribute, 'ngenes', indicating the number of

profiled genes. The arguments 'pn', 'qm' and 'pqn0' are compared in a column by column wise, recycling columns, if necessary, in order to perform max(ncol(pn),ncol(qm),ncol(pqn0)) comparisons (each comparison resulting in an object of class 'GOProfileHtest', an specialization of 'htest'). In order to be properly compared, these arguments are expanded by row, according to their row names. That is, the data arguments can have unequal row numbers. Then, they are expanded adding rows with zero frequencies, in order to make them comparable.

In the i-th comparison (i from 1 to max(ncol(pn),ncol(qm),ncol(pqn0))), the parameters n, m and n0 are included to allow the possibility of exploring the consequences of varying sample sizes, other than the true sample sizes included as an attribute in pn, qm and pqn0.

When qm = NULL, the genes profiled in pn are compared with a subsample of them, those profiled in pqn0 (compare a set of genes with a restricted subset, e.g. those overexpressed under a disease). In this case we take qm=pqn0. When pqn0 = NULL, two profiles with no genes in common are compared.

Let Pn and Qm correspond to the contracted functional profiles (the total counts or relative frequencies of hits in each one of the s GO categories being compared) obtained from pn and qm. If P stands for the "population" profile originating the sample profile Pn[,j], Q for the profile originating Qm[,j] and d(,) for the squared euclidean distance, if P != Q, the distribution of sqrt(nm/(n+m))(d(Pn[,j],Qm[,j]) - d(P,Q))/se(d) is approximately standard normal, N(0,1). This provides the basis for the confidence interval in the result field icDistance. When P=Q, the asymptotic distribution of (nm/(n+m)) d(Pn[,j],Qm[,j]) corresponds to the distribution of a mixture of independent chi-square random variables, each one with one degree of freedom. The sampling distribution under H0 P=Q may be directly computed from this distribution (approximating it by simulation) (method="lcombChisq") or by a chi-square approximation to it, based on two correcting constants a and b (method="chi-square"). These constants are chosen to equate the first two moments of both distributions (the linear combination of chi-square random variables distribution and the approximating chi-square distribution). When method="chi-square", the returned test statistic value is the chi-square approximation (n d(pn[,j],qm[,j]) - b) / a. Then, the result field 'parameter' is a vector containing the 'a' and 'b' values and the number of degrees of freedom, 'df'. Otherwise, the returned test statistic value is (nm/(n+m)) d(Pn[,j],Qm[,j]) and 'parameter' contains the coefficients of the linear combination of chi-squares.

## Value

A list containing max(ncol(pn),ncol(qm),ncol(pqn0)) objects of class 'GOProfileHtest', directly inheriting from 'htest' or a single 'GOProfileHtest' object if max(ncol(pn),ncol(qm),ncol(pqn0))==1 and simplify == T. Each object of class 'GOProfileHtest' has the following fields:

| | |
|---|---|
| `profilePn` | the first contracted profile to compute the squared Euclidean distance |
| `profileQm` | the second contracted profile to compute the squared Euclidean distance |
| `statistic` | test statistic; its meaning depends on the value of "method", see the 'Details' section. |
| `parameter` | parameters of the sample distribution of the test statistic, see the 'Details' section. |
| `p.value` | associated p-value to test the null hypothesis of profiles equality. |
| `conf.int` | asymptotic confidence interval for the squared euclidean distance. Its attribute "conf.level" contains its nominal confidence level. |

| | |
|---|---|
| estimate | squared euclidean distance between the contracted profiles. Its attribute "se" contains its standard error estimate. |
| method | a character string indicating the method used to perform the test. |
| data.name | a character string giving the names of the data. |
| alternative | a character string describing the alternative hypothesis (always 'true squared Euclidean distance between the contracted profiles is greater than zero' |

## Author(s)

Jordi Ocana

## References

Sanchez-Pla, A., Salicru M. and Ocana, J. Statistical methods for the analysis of highthroughput data based on functional profiles derived from the gene ontology. Journal of Statistical Planning and Inference, 2007.

## See Also

fitGOProfile, equivalentGOProfiles

## Examples

```
# [NOT RUN COMPLETELY]
data(prostateIds)
expandedWelsh <- expandedProfile(welsh01EntrezIDs[1:100], onto="MF",
                        level=2, orgPackage="org.Hs.eg.db")
expandedSingh <- expandedProfile(singh01EntrezIDs[1:100], onto="MF",
                        level=2, orgPackage="org.Hs.eg.db")
commonGenes <- intersect(welsh01EntrezIDs[1:100], singh01EntrezIDs[1:100])
commonExpanded <- expandedProfile(commonGenes, onto="MF", level=2, orgPackage="org.Hs.eg.db")
# comparedMF <-compareGOProfiles (pn=expandedWelsh,
#                           qm  = expandedSingh,
#                           pqn0= commonExpanded)
# print(comparedMF)
# print(compSummary(comparedMF))
#
```

---

# compareProfilesLists *Compares two of expanded profiles*

---

## Description

This function compares two lists ("sensu R lists") of expanded profiles by successive calls to function compareGOProfiles following the methodology developed by Sanchez, Salicru and Ocan\~a (2007)

## Usage

```
compareProfilesLists(expanded1, expanded2, common.expanded=NULL, relationType,
method = "lcombChisq", ab.approx = "asymptotic", confidence = 0.95, ...)
```

## Arguments

| | |
|---|---|
| expanded1 | First expanded profile to be compared |
| expanded2 | Second expanded profile to to be compared |
| common.expanded | Expanded profile made from the genes appearing in both lists of genes |
| relationType | Type of relation between gene lists compared through the expanded profiles. It can be INCLUSION, INTERSECTION or DISJOINT |
| method | The approximation method to the sampling distribution under the null hypothesis specifying that the samples pn and qm come from the same population. See the 'Details' section below |
| confidence | The confidence level of the confidence interval in the result |
| ab.approx | The approximation used for computing 'a' and 'b' coefficients (see details) |
| ... | Other arguments for the methods 'basicProfile' or 'compareGoProfiles' |

## Value

The result of the comparison is a list with a variable number of arguments, depending for which ontologies has been performed the comparison. Each list member is an object of class 'htest' corresponding to the output of the function compareGOProfiles

## Author(s)

Alex Sanchez

## References

Sanchez-Pla, A., Salicru, M. and Ocana, J. Statistical methods for the analysis of high-throughput data based on functional profiles derived from the gene ontology. Journal of Statistical Planning and Inference, 2007

## See Also

compareGeneLists, expandedProfile

## Examples

```
#[NOT RUN]
#data(ProstateIds)
#expandedWelsh <- expandedProfile(welsh01EntrezIDs[1:100], onto="MF",
#                        level=2, orgPackage="org.Hs.eg.db")
#expandedSingh <- expandedProfile(singh01EntrezIDs[1:100], onto="MF",
#                        level=2, orgPackage="org.Hs.eg.db")
#commonGenes <- intersect(welsh01EntrezIDs[1:100], singh01EntrezIDs[1:100])
```

```
#commonExpanded <- expandedProfile(commonGenes, onto="MF", level=2, orgPackage="org.Hs.eg.db")
#comparedMF<- compareProfilesLists (expandedWelsh, expandedSingh, commonExpanded,
#                                    relationType="COMMON")
#print(comparedMF)
#print(compSummary(comparedMF))
```

---

`compSummary` *This function returns a brief summary of the comparison between two (expanded) profiles.*

---

## Description

Function to return a brief summary of the comparison between two (expanded) profiles.

## Usage

```
compSummary(l, decs = 6)
```

## Arguments

| | |
|---|---|
| `l` | A list of comparison results as returned by a call to `compareGenelists` |
| `decs` | Number of decimal places to use in the output |

## Value

A data frame with the summarized results of each comparison. The values contained are: `Sqr.Eucl.Dist`: The squared euclidean distance, `Standard Err`: The standard error estimate, `pValue` p value of the test, `low conf.int`Lower value for the desired confidence interval, `up conf.int`Upper value for the desired condfidence interval.

## Author(s)

Alex Sanchez

## Examples

```
# (NOT RUN)
# data(prostateIds)
# expandedWelsh <- expandedProfile(welsh01EntrezIDs[1:100], onto="MF",
#                         level=2, orgPackage="org.Hs.eg.db")
# expandedSingh <- expandedProfile(singh01EntrezIDs[1:100], onto="MF",
#                         level=2, orgPackage="org.Hs.eg.db")
# commonGenes <- intersect(welsh01EntrezIDs[1:100], singh01EntrezIDs[1:100])
# commonExpanded <- expandedProfile(commonGenes, onto="MF", level=2, orgPackage="org.Hs.eg.db")
# comparedMF <-compareGOProfiles (pn=expandedWelsh,
#                            qm  = expandedSingh,
#                            pqn0= commonExpanded)
# print(comparedMF)
# print(compSummary(comparedMF))
#
```

`contractedProfile` *Converts an expanded GO profile into a basic (contracted) GO profile*

## Description

Converts an object of class 'ExpandedGOProfile', or assimilable to it, in an object of class 'BasicGOProfile'

## Usage

```
contractedProfile(prof, nams = NULL)
## S3 method for class 'ExpandedGOProfile'
contractedProfile(prof, nams = NULL)
## Default S3 method:
contractedProfile(prof, nams = NULL)
```

## Arguments

| | |
|---|---|
| `prof` | an expanded GO profile, i.e. and object of class 'ExpandedGOProfile', or a numeric vector assimilable to an expanded profile, see the "details" section |
| `nams` | optionally, the names of the annotated combinations of GO nodes whose frequency is represented in the expanded profile, see the "details" section |

## Details

Given a list of n genes, and a set of s GO nodes X, Y, Z, ... in a given ontology (BP, MF or CC), its associated (contracted) "profile" is the frequencies vector (either absolute or relative frequencies) of annotations or hits of the n genes in each node. For a given node, say X, this frequency includes all annotations for X alone, for X and Y, for X and Z and so on. Thus, as relative frequencies, its sum is not necessarily one, or as absolute frequencies their sum is not necessarily n. Basic contracted profiles are represented by objects of S3 class 'BasicGOProfile'. On the other hand, an "expanded profile" corresponds to the frequencies in ALL OBSERVED NODE COMBINATIONS. That is, if n genes have been profiled, the expanded profile stands for the frequency of all hits EXCLUSIVELY in nodes X, Y, Z, ..., jointly with all hits simultaneously in nodes X and Y (and only in X and Y), simultaneously in X and Z, in Y and Z, ... , in X and Y and Z (and only in X,Y,Z), and so on. Thus, their sum is one. Expanded profiles are represented by objects of S3 class 'ExpandedGOProfile'. The generic function 'contractedProfile' "contracts" an expanded profile, either represented by a 'ExpandedGOProfile' object or a numeric vector interpretable as an expanded profile, in order to obtain its contracted profile representation.

The `rownames` attribute of an 'ExpandedGOProfile' or, equivalently, the `names` attribute of a vector representing an expanded profile, or the `nams` argument, must represent the GO nodes combinations separating the node names with dots, ".", for example: "X", "Y", "Z", "X.Y", "X.Z", "Y.Z", "X.Y.Z" and so on.

## Value

An object of class 'BasicGOProfile' the contracted profile representation of the expanded profile

### Author(s)

Jordi Ocana

### Examples

```
data(prostateIds)
expandedWelsh <- expandedProfile(welsh01EntrezIDs[1:100], onto="MF",
                        level=2, orgPackage="org.Hs.eg.db")
reContractedWelsh <- contractedProfile(expandedWelsh[["MF"]])
print(expandedWelsh)
print(reContractedWelsh)
class(reContractedWelsh)
ngenes(reContractedWelsh)
```

---

## conversionFunctions *Functions to transformconvert objects between different types*

---

### Description

These functions transform data from one classtype into another, or pack simple processes such as compute the profiles needed for one annotations package.

### Usage

```
as.GOTerms.frame(myGOTermsList, na.rm=TRUE)
as.GOTerms.list(genelist, probeType, orgPackage=NULL, anotPkg=NULL,
onto="any", na.rm=TRUE)
BioCpack2EntrezIDS(anotPkg, na.rm=TRUE)
BioCpack2Profiles(anotPkg, orgPackage, level=2, na.rm=TRUE, expanded=FALSE)
BioCprobes2Entrez(probeslist , anotPkg, na.rm=TRUE)
GOTermsFrame2GOTermsList(myGOTermsFrame, evid=FALSE)
```

### Arguments

| | |
|---|---|
| myGOTermsList | GOTermsList object to transform |
| myGOTermsFrame | GOTermsFrame object to transform |
| genelist | List of genes (Entrez Ids) to transform |
| evid | Type of evidence supporting the selected GO Terms |
| na.rm | Flag indicating if those ids returning NA must be removed from the output |
| probeType | Type of probes to transform into Entrez Ids |
| probeslist | List of probes to transform into Entrez Ids |
| orgPackage | Name of the organism ('org.Xx.eg.db') annotation package |
| anotPkg | Name of the chip annotation package |
| level | GO level at which the profile is built |
| onto | ontology |
| expanded | Flag to decide if an expanded profile has to be computed |

## Details

Not yet available

## Value

Every function returns a transformed object or a list of computed profiles

## Author(s)

Alex Sanchez

## Examples

```
data(CD4Ids)
myGOTermsList <- GOTermsList(CD4LLids[1:5], orgPkg="org.Hs.eg.db")
myGOTermsFrame<- as.GOTerms.frame(myGOTermsList, na.rm=TRUE)
GOTermsFrame2GOTermsList(myGOTermsFrame, evid=FALSE)
```

---

# drosophila *Entrez identifiers for genes related with an eye mutation in drosophila*

---

## Description

Entrez identifiers for genes related with an eye mutation in drosophila.

`ostrinIds` List of genes in Entrez, generated by Ostrin et al.

`michaudIds` List of genes in Entrez, generated by Michaud et al.

`drosophilaIds` List of Drosophila genes in Entrez.

## Usage

```
data(drosophila)
```

## Format

Each dataset is a character vector with a different number of elements which (should) correspond to valid Entrez identifiers

## Examples

```
data(drosophila)
```

`equivalentGOProfiles` *Are two lists of genes equivalent in terms of their Gene Ontology profiles?*

## Description

Performs an equivalence test based on the squared Euclidean distance between the Gene Ontology profiles of two lists of genes. Equivalence is declared if the upper limit d.sup of a one-sided confidence interval [0, d.sup] for the distance is lesser than the equivalence limit d0.

## Usage

```
equivalentGOProfiles(goObject, ...)
## S3 method for class 'GOProfileHtest'
equivalentGOProfiles(goObject, equivEpsilon = 0.05, d0 = NULL, confidence = NULL, ...)
## S3 method for class 'ExpandedGOProfile'
equivalentGOProfiles(goObject, qm=NULL, pqn0=NULL,
    n = ngenes(goObject), m = ngenes(qm), n0 = ngenes(pqn0),
    confidence = 0.95,
    equivEpsilon = 0.05, d0 = NULL,
    simplify = FALSE, ...)
## Default S3 method:
equivalentGOProfiles(goObject, ...)
```

## Arguments

| | |
|---|---|
| `goObject` | an object related to GO profiles or comparisons between them |
| `qm` | an expanded GO profile, i.e. and object of class 'ExpandedGOProfile' |
| `pqn0` | an expanded GO profile, i.e. and object of class 'ExpandedGOProfile' |
| `n` | a numeric vector with the number of genes profiled in each column of goObject. This parameter is included to allow the possibility of exploring the consequences of varying sample sizes, other than the true sample size in goObject. |
| `m` | a numeric vector with the number of genes profiled in each column of qm. |
| `n0` | a numeric vector with the number of genes profiled in each column of pqn0. |
| `confidence` | the nominal confidence level of the one-sided confidence interval on the distance |
| `d0` | a positive value specifying the equivalence limit |
| `equivEpsilon` | a positive value used to compute 'd0' if it is not directly available |
| `simplify` | should the result be simplified, if possible? See the 'Details' section |
| `...` | further arguments, tipically the same than to 'compareGOProfiles' |

## Details

An object of S3 class "ExpandedGOProfile" is, essentially, a "data.frame" object with each column representing the relative frequencies in all observed node combinations, resulting from profiling a set of genes, for a given and fixed ontology. The 'row.names' attribute codifies the node combinations and each "data.frame" column (say, each profile) has an attribute, 'ngenes', indicating the number of profiled genes.

In the 'ExpandedGOProfile' interface, the arguments 'goObject', 'qm' and 'pqn0' are compared in a column by column wise, recycling columns, if necessary, in order to perform max(ncol(goObject),ncol(qm),ncol(pqn0)) equivalence tests (each test resulting in an object of class 'htest'). In order to be properly tested, these arguments are expanded by row, according to their row names. That is, the data arguments can have unequal row numbers. Then, they are expanded adding rows with zero frequencies, in order to make them comparable. In the i-th comparison (i from 1 to max(ncol(goObject),ncol(qm),ncol(pqn0))), the parameters n, m and n0 are included to allow the possibility of exploring the consequences of varying sample sizes, other than the true sample sizes included as an attribute in goObject, qm and pqn0. When qm = NULL, the genes profiled in goObject are compared with a subsample of them, those profiled in pqn0 (is there equivalence between a set of genes and a restricted subset, e.g. those overexpressed under a disease, in terms of their profiles?). When pqn0 = NULL, an equivalence test between two profiles with no genes in common is performed.

In the 'GOProfileHtest' interface, the one-sided confidence interval for the squared Euclidean distance is computed from the distance and its standard error stored in the corresponding fields of the argument goObject, itself typically an object of class 'GOProfileHtest' resulting from a call to 'compareGOProfiles' with simplify=T.

In the default interface, the 'goObject' argument is previously converted into an object of class 'ExpandedGOProfile' and then this interface is used.

If the argument 'd0' is not provided it is computed as $d0 < -s * equivEpsilon^2$, where 's' stands for the number of non empty GO nodes in any of the GO profiles being compared.

## Value

In the 'ExpandedGOProfile' interface, the result is an object of class "list" containg one or more "htest" objects, each of which may come from previous profiles comparisons. In the other interfaces, the result is a single "htest" object. Each one of these "htest" objects has the following fields:

| | |
|---|---|
| `statistic` | test statistic, (distance - d0) / se |
| `parameter` | d0 and the sample sizes (number of genes) n and m |
| `p.value` | associated p-value to test the null hypothesis of profiles inequivalence |
| `conf.int` | asymptotic one-sided confidence interval for the squared euclidean distance. Its attribute "conf.level" contains its nominal confidence level. |
| `estimate` | squared euclidean distance between the contracted profiles. Its attribute "se" contains its standard error estimate |
| `data.name` | a character string giving the names of the data |
| `alternative` | a character string describing the alternative hypothesis (always 'Equivalence or similarity, true squared Euclidean distance between the contracted profiles is less than d0' |

### Author(s)

Jordi Ocana

### See Also

'compareGOProfiles'

### Examples

```
data(prostateIds)

expandedWelsh <- expandedProfile(welsh01EntrezIDs[1:100], onto="ANY",
                                 level=2, orgPackage="org.Hs.eg.db")
expandedSingh <- expandedProfile(singh01EntrezIDs[1:100], onto="ANY",
                                 level=2, orgPackage="org.Hs.eg.db")
commonGenes <- intersect(welsh01EntrezIDs[1:100], singh01EntrezIDs[1:100])
commonExpanded <- expandedProfile(commonGenes, onto="ANY", level=2,
                                  orgPackage="org.Hs.eg.db")

### FUnciona si fem:

equivMF <-equivalentGOProfiles (expandedWelsh[["MF"]],
                            qm  = expandedSingh[["MF"]],
                            pqn0= commonExpanded[["MF"]])

equivsList <- lapply(1:length(expandedWelsh),
                     function (onto){
                          equivalentGOProfiles (expandedWelsh[[onto]],
                                qm  = expandedSingh[[onto]],
                                pqn0= commonExpanded[[onto]])
                     }
)
```

---

## equivClust

*For a given level (2, 3, ...) in a GO ontology (BP, MF or CC), compute the equivalence threshold distance matrix and generate a dendrogram from it.*

---

### Description

For a given level (2, 3, ...) in a GO ontology (BP, MF or CC), compute the equivalence threshold distance matrix and generate a dendrogram from it.

### Usage

```
equivClust(ontoLevel, onto, geneLists, trace = TRUE, onTheFlyDev = NULL,
  method = "complete", jobName = paste("Equivalence cluster", onto,
  ontoLevel, method, sep = "_"), ylab = "Equivalence threshold distance",
  alpha = 0.05, precis = 0.001, ...)
```

## Arguments

| | |
|---|---|
| ontoLevel | integer (2, 3, ...) level of a GO ontology where the GO profiles are built |
| onto | character, GO ontology ("BP", "MF" or "CC") under consideration |
| geneLists | list of character vectors, each vector stands for the gene names in a given gene set |
| trace | boolean, the full process must be traced? Defaults to TRUE |
| onTheFlyDev | character, name of the graphical device where to immediately display the resulting diagram. The appropriate names depend on the operating system. Defaults to NULL and then nothing is displayed |
| method | character, one of the admissible methods in function hclust. Defaults to "complete" |
| jobName | character, main plot name, defaults to paste("Equivalence cluster", onto, ontoLevel, method, sep = "_") |
| ylab | character, label of the vertical axis of the plot, defaults to "Equivalence threshold distance" |
| alpha | simultaneous nominal significance level for the equivalence tests to be repeteadly performed, defaults to 0.05 |
| precis | numerical precission in the iterative search of the equivalence threshold distances, defaults to 0.001 |
| ... | additional arguments to hclust |

## Details

Do not confuse the threshold distance matrix with the squared distances computed in each equivalence test.

## Value

An object of class equivClust, descending from class hclust with some additional attributes:

**jobName** The main job name

**sub** The graphic subtittle

**ylab** The vertical axis label

**distMat** The equivalence threshold distance matrix

**allComps** A list with some information on all the pairwise equivalence tests: the Euclidean squared distance, its standard error and the corresponding GO profiles

## Examples

```
## Not run:
data(kidneyGeneLists)
clustMF2 <- equivClust(2, "MF", kidneyGeneLists, orgPackage="org.Hs.eg.db")
plot(clustMF2)
plot(clustMF2,
     main = "Dendrogram (method = complete)", sub = attr(clustMF2, "sub"),
```

```
     ylab = "Equivalence threshold distance")
# With the same data, an UPGMA dendrogram:
equivClust(2, "MF", kidneyGeneLists, method = "average",
orgPackage="org.Hs.eg.db")

## End(Not run)
```

---

`equivClust2pdf` *Save the graphical representation of objects of class* `equivClust` *or* `iterEquivClust` *as pdf files.*

---

## Description

Save the graphical representation of objects of class `equivClust` or `iterEquivClust` as pdf files.

## Usage

```
equivClust2pdf(x, ...)

## S3 method for class 'equivClust'
equivClust2pdf(x, jobName, ylab, ...)

## S3 method for class 'iterEquivClust'
equivClust2pdf(x, jobName, ylab, ...)
```

## Arguments

| | |
|---|---|
| `x` | an object of class `equivClust` or `iterEquivClust` |
| `...` | additional arguments to function `pdf` |
| `jobName` | character, main plot title and file name (it should be correct as a file name!) |
| `ylab` | character, label of the plot vertical axis |

## Methods (by class)

- `equivClust`: `equivClust2pdf` method for class `equivClust`
- `iterEquivClust`: `equivClust2pdf` method for class `iterEquivClust`

## Examples

```
data(clustKidneyMF2)
equivClust2pdf(clustKidneyMF2)
# And then open file "Equivalence cluster_MF_2_complete.pdf"...
equivClust2pdf(clustKidneyMF2,
               jobName = "Method 'complete' dendrogram for level 2 of GO ontology MF")
# And then open file "Method 'complete' dendrogram for level 2 of GO ontology MF.pdf"...
```

`equivSummary` *This function returns a brief summary of the equivalence test between two profiles.*

## Description

Function to return a brief summary of the equivalence test between two profiles. If In its current version it is better that `equivalentGOProfiles` is called with option `simplify` set to `FALSE` before `equivSummary` can be used

## Usage

```
equivSummary(l, decs = 6)
```

## Arguments

| | |
|---|---|
| `l` | A list of comparison results as returned by a call to `compareGenelists` |
| `decs` | Number of decimal places to use in the output |

## Value

A data frame with the summarized results of each comparison. The values contained are: `Sqr.Eucl.Dist`: The squared euclidean distance, `Standard Err`: The standard error estimate, `pValue` p value of the equivalence test, up `conf.int`Upper value for the desired condfidence interval. `d0`Threshold value for equivalence test. `Equivalent`?Numerical value set to 1 if profiles can be considered equivalent and to zero if they cannot.

## Author(s)

Alex Sanchez

## See Also

'equivalentGOProfiles'

## Examples

```
# data(prostateIds)
# expandedWelsh <- expandedProfile(welsh01EntrezIDs[1:100], onto="MF",
#                         level=2, orgPackage="org.Hs.eg.db")
# expandedSingh <- expandedProfile(singh01EntrezIDs[1:100], onto="MF",
#                         level=2, orgPackage="org.Hs.eg.db")
#commonGenes <- intersect(welsh01EntrezIDs[1:100], singh01EntrezIDs[1:100])
#commonExpanded <- expandedProfile(commonGenes, onto="MF", level=2, orgPackage="org.Hs.eg.db")

# equivMF <-equivalentGOProfiles (pn=expandedWelsh,
#                           qm  = expandedSingh,
#                           pqn0= commonExpanded)
#print(equivSummary(equivMF, decs=5))
```

expandedLevel *Function to create expanded levels which can contain GO Terms at different GO levels*

## Description

This function, combined with function expandTerm, allows to create mixed levels which can contain terms belonging to different GO levels. Specifically one can take one (or several, but one by one) term at a given GO level and expand it into its children terms using function expandTerm and then combine them into a new level using this function.

## Usage

```
expandedLevel(LevelTerms, Term2Expand, onto)
expandTerm(GOTerm, onto)
```

## Arguments

| | |
|---|---|
| LevelTerms | Other terms which have not been expanded, and will be combined with the expanded ones |
| Term2Expand | The GO term which will be substituted by its children terms |
| GOTerm | The GO term which will be substituted by its children terms |
| onto | The ontology ('MF','BP','CC' |

## Value

The value returned is the vector combining the original terms with the children of the term that had to be expanded.

## Author(s)

Alex Sanchez

## Examples

```
got<-toTable(GOTERM)[,2:3]
desc<-function(s) got[got[,1]==s,2]
MFLevel2<-getGOLevel("MF",2)
bindingLevel2<-MFLevel2 [2]
bindingLevel3 <- expandTerm(bindingLevel2,"MF")
print(descbindingLevel3<-as.matrix(sapply(bindingLevel3,desc )))
mixedLevel<-c(MFLevel2[-2],bindingLevel3)
print(mixedLevel<-as.matrix(sapply(mixedLevel,desc )))
```

`expandedProfile` *Builds expanded profiles*

## Description

Expanded profiles are used mainly for comparison of profiles based on the theory developed by Sanchez et al (2007) (see references)

## Usage

```
expandedProfile(genelist, idType = "Entrez", onto = "ANY", level = 2,
orgPackage=NULL, anotPackage=NULL,  multilevels = NULL, ord = TRUE,
na.rm = TRUE, percentage = TRUE)
```

## Arguments

| | |
|---|---|
| `genelist` | List of genes on which the Profile has to be based |
| `idType` | Type of identifiers for the genes. Use 'Entrez' preferably |
| `onto` | Ontology on which the profile has to be built |
| `level` | Level of the ontology at which the profile has to be built |
| `orgPackage` | Name of a Bioconductor's organism annotations package ('org.Xx-eg-db'). This field must be provided if the gene list passed to the function is either a character vector of 'Entrez' (NCBI) identifiers or a character vector of probe names |
| `anotPackage` | Name of Bioconductor annotations package. This field must be provided if the gene list passed to the function is a character vector of probe names |
| `ord` | Set to 'TRUE' if the profile has to appear ordered by the category names |
| `multilevels` | If it is not NULL it must be a vector of GO categories that defines the level at where the profile is built |
| `na.rm` | Set to 'TRUE' if NAs should be removed |
| `percentage` | Set to 'TRUE' if the profile must be built using percentages |

## Details

The function admits three types of entries: Entrez ('Entrez'), Bioconductor probe set names ('BioCprobes') or a special type of data frames ('GOTermsFrames'). If the identifier type are 'BioCprobes' then an annotation package name must be provided too.

## Value

An object of class GOProfile containing an expanded profile

## Author(s)

Alex Sanchez

## References

Sanchez-Pla, A., Salicru, M. and Ocana, J. Statistical methods for the analysis of high-throughput data based on functional profiles derived from the gene ontology. Journal of Statistical Planning and Inference, Volume 137, Issue 12, Pages 3975-3989, 2007.

## See Also

basicProfile

## Examples

```
data(CD4Ids)
CD4.Expanded <-expandedProfile(genelist=CD4LLids[1:50], onto='MF',
level=2, orgPackage="org.Hs.eg.db")
```

---

`fisherGOProfiles` *GO Class-by-class Fisher tests in lists of genes characterized by their functional profiles*

---

## Description

Given two lists of genes, both characterized by their frequencies of annotations (or "hits") in the same set of GO nodes (also designated as GO terms or GO classes), for each node determine if the annotation frequencies depart from what is expected by chance. The annotation frequencies are specified in the "GO profiles" arguments `pn`, `qm` and `pn`. Both samples may share a common subsample of genes, with GO profile `pqn0`. The analysis is based on the Fisher's exact test, as is implemented by `fisher.test` R function, followed by p-value adjustment for multitesting based on function `p.adjust`. Usually, this function will be called after a significant result on `compareGOProfiles` which performs global (all GO nodes simultaneously) profile comparisons (with better type I and type II error control), to identify the more rellevant nodes.

## Usage

```
fisherGOProfiles(pn, ...)
## S3 method for class 'numeric'
fisherGOProfiles(pn, qm=NULL, pqn0=NULL,
    n = ngenes(pn), m = ngenes(qm), n0 = ngenes(pqn0),
    method = "BH", simplify=T, expanded=F, ...)
## S3 method for class 'matrix'
fisherGOProfiles(pn, n, m, method = "BH", ...)
## S3 method for class 'BasicGOProfile'
fisherGOProfiles(pn, qm=NULL, pqn0=NULL,
    method = "BH", goIds=T, ...)
## S3 method for class 'ExpandedGOProfile'
fisherGOProfiles(pn, qm=NULL, pqn0=NULL,
    method = "BH", simplify=T, ...)
```

## Arguments

| | |
|---|---|
| pn | an object of class BasicGOProfile or ExpandedGOProfile representing a "sample" GO profile for a fixed ontology, or a numeric vector interpretable as a GO profile (expanded or not), or a two-dimensional frequency matrix (see the 'Details' section). This is a required argument |
| qm | similarly, an object representing a "sample" GO profiles for a fixed ontology |
| pqn0 | an object representing a "sample" GO profile for a fixed ontology |
| n | the number of genes profiled in pn |
| m | the number of genes profiled in qm |
| n0 | the number of genes profiled in pqn0 |
| method | the p-values adjusting method for multiple comparisons; the same possibilities as in standard R function p.adjust |
| expanded | boolean; are these numeric vectors representing expanded profiles? |
| simplify | should the result be simplified, if possible? See the 'Details' section |
| goIds | if TRUE, each node is represented by its GO identifier |
| ... | other arguments (to be passed to p.adjust or fisher.test functions) |

## Details

Given a list of n genes, and a set of s GO classes or nodes X, Y, Z, ... in a given ontology (BP, MF or CC), its associated ("contracted" or "basic") "profile" is the absolute frequencies vector of annotations or hits of the n genes in each one of the s GO nodes. For a given node, say X, this frequency includes all annotations for X alone, for X and Y, for X and Z and so on. Thus, as relative frequencies, its sum is not necessarily one, or as absolute frequencies their sum is not necessarily n. On the other hand, an "expanded profile" corresponds to the relative frequencies in ALL NODE COMBINATIONS. That is, if n genes have been profiled, the expanded profile stands for the frequency of all hits EXCLUSIVELY in node X, exclusively in node Y, exclusively in Z, ..., jointly with all hits simultaneously in nodes X and Y (and only in X and Y), simultaneously in X and Z, in Y and Z, ... , in X and Y and Z (and only in X,Y,Z), and so on. Thus, their sum is one.

Let n, m and n0 designate the total number of genes profiled in pn, qm and pqn0 respectively. According to these profiles, n[i], m[i] and n0[i] genes are annotated for node 'i', i = 1, . . . , s. Note that the sum of all the n[i] not necessarily equals n and so on. If not NULL, pqn0 stands for the profile of the n0 genes common to the gene lists that gave rise to pn and qm. fisherGOProfiles builds a sx2 absolute frequencies matrix

| | | |
|---|---|---|
| GO node 1 | N[1,1] | N[1,2] |
| GO node 2 | N[2,1] | N[2,2] |
| ... | ... | ... |
| GO node s | N[2,1] | N[s,2] |

with column totals N1 and N2 (not necessarily equal to the column sums) and performs a Fisher's exact test over each one of the 2x2 tables

| | | |
|---|---|---|
| GO node i | N[i,1] | N[i,2] |
| All nodes except i | N1 - N[i,1] | N2 - N[i,2] |

followed by a p-value correction for multiplicity in testing. If pqn0 is NULL, then both gene lists do not have any genes in common, N[i,1] = n[i] and N[i,2] = m[i], and N1 = n, N2 = m, n0 = 0. Otherwhise (if pqn0 is not NULL) N[i,1] = n[i] - n0[i], N1 = n - n0 and N[i,2] = n0[i], N2 = n0 if qm is NULL, or N[i,2] = m[i], N2 = m if qm is not NULL.

In other words, this function provides a general setting for diverse, common in practice, situations where a node-by-node analysis is required. When pqn0 = NULL, two lists with no genes in common are compared. Otherwise, when qm = NULL, the genes profiled in pn are compared with a subsample of them, those profiled in pqn0 (a set of genes vs a restricted subset, e.g. those over-expressed under a disease). Finally, if both arguments qm and pqn0 are not NULL (pn is always required) two gene lists with some genes in common are analised.

If both qm and pqn0 are NULL, pn should correspond to an absolute frequencies matrix with s rows and 2 columns.

The arguments n, m or n0 are only required in case of numeric vectors or matrices specifying profiles but lacking the 'ngenes' attribute.

## Value

A list containing max(ncol(pn),ncol(qm),ncol(pqn0)) p-values numeric vectors, or a single p-values vector if max(ncol(pn),ncol(qm),ncol(pqn0))==1 and simplify == T.

## Author(s)

Jordi Ocana

## References

Sanchez-Pla, A., Salicru M. and Ocana, J. Statistical methods for the analysis of highthroughput data based on functional profiles derived from the gene ontology. Journal of Statistical Planning and Inference, 2007.

## See Also

fitGOProfile, compareGOProfiles, equivalentGOProfiles

## Examples

```
require("org.Hs.eg.db")
data(prostateIds)
# To improve speed, use only the first 100 genes:
list1 <- welsh01EntrezIDs[1:100]
list2 <- singh01EntrezIDs[1:100]
prof1 <- basicProfile(list1, onto="MF", level=2, orgPackage="org.Hs.eg.db")$MF
prof2 <- basicProfile(list2, onto="MF", level=2, orgPackage="org.Hs.eg.db")$MF
commProf<-basicProfile(intersect(list1, list2), onto="MF",level=2, orgPackage="org.Hs.eg.db")$MF
fisherGOProfiles(prof1, prof2, commProf, method="holm")
```

`fitGOProfile` *Does a "sample" GO profile 'pn', observed in a sample of 'n' genes, fit a "population" or "model" p0?*

## Description

'fitGOProfile' implements some inferential procedures to solve the preceding question. These procedures are based on asymptotic properties of the squared euclidean distance between the contracted versions of pn and p0

## Usage

```
fitGOProfile(pn, p0, n = ngenes(pn), method = "lcombChisq",
ab.approx = "asymptotic", confidence = 0.95, simplify = T)
```

## Arguments

| | |
|---|---|
| `pn` | an object of class ExpandedGOProfile representing one or more "sample" expanded GO profiles for a fixed ontology (see the 'Details' section) |
| `p0` | an object of class ExpandedGOProfile representing one or more "population" or "theoretical" expanded GO profiles (see also the 'Details' section) |
| `n` | a numeric vector with the number of genes profiled in each column of pn. This parameter is included to allow the possibility of exploring the consequences of varying sample sizes, other than the true sample size in pn |
| `method` | the approximation method to the sampling distribution under the null hypothesis "p = p0", where p is the 'true' population profile originating each column of pn. See the 'Details' section below |
| `ab.approx` | the method used to compute the constants 'a' and 'b' described in the paper. See the 'Details' section |
| `confidence` | the confidence level of the confidence interval in the result |
| `simplify` | should the result be simplified, if possible? See the 'Details' section |

## Details

An object of class 'ExpandedGOProfile' is, essentially, a 'data.frame' object with each column representing the relative frequencies in all observed node combinations, resulting from profiling a set of genes, for a given and fixed ontology. The row.names attribute codifies the node combinations and each data.frame column (say, each profile) has an attribute, 'ngenes', indicating the number of profiled genes. (Actually, the 'ngenes' attribute of each 'p0' column is ignored and is taken as if it were infinite, 'Inf'.) The arguments 'pn' and 'p0' are compared in a column by column wise, recycling columns, if necessary, in order to perform max(ncol(pn),ncol(p0)) comparisons (each comparison resulting in an object of class 'htest'). In order to be properly compared, 'pn' and 'p0' are expanded by row, according to their row names. That is, both arguments can have unequal row numbers. Then, they are expanded adding rows with zero frequencies, in order to make them comparable.

In the i-th comparison (i from 1 to max(ncol(pn),ncol(p0))), if p stands for the profile originating the sample profile pn[,i] and d(,) for the squared euclidean distance, if p != p0[,i], the distribution of sqrt(n)(d(pn[,i],p0[,i]) - d(p,p0[,i]))/se is approximately standard normal, N(0,1). This provides the basis for the confidence interval in the result field conf.int. When p==p0[,i], the asymptotic distribution of n d(pn[,i],p0[,i]) is the distribution of a linear combination of independent chi-square random variables, each one with one degree of freedom. This sampling distribution may be directly computed (approximating it by simulation, method="lcombChisq") or approximated by a chi-square distribution, based on two correcting constants a and b (method="chi-square"). These constants are chosen to equate the first two moments of both distributions (the distribution of a linear combination of chi square variables and the approximating chi-square distribution). When method="chi-square", the returned test statistic value is the chi-square approximation (n d(pn,p0) - b) / a. Then, the result field 'parameter' is a vector containing the 'a' and 'b' values and the number of degrees of freedom, 'df'. Otherwise, the returned test statistic value is n d(pn,p0) and 'parameter' contains the coefficients of the linear combination of chi-squares

## Value

A list containing max(ncol(pn),ncol(p0)) objects of class 'htest', or a single 'htest' object if ncol(pn)==1 and ncol(p0)==1 and simplify == T. Each 'htest' object has the following fields:

| | |
|---|---|
| `statistic` | test statistic; its meaning depends on the value of "method", see the 'Details' section |
| `parameter` | parameters of the sample distribution of the test statistic, see the 'Details' section |
| `p.value` | associated p-value to test the null hypothesis "pn[,i] is a random sample taken from p0[,i]" |
| `conf.int` | asymptotic confidence interval for the squared euclidean distance. Its attribute "conf.level" contains its nominal confidence level |
| `estimate` | squared euclidean distance between the contracted pn and p0 profiles. Its attribute "se" contains its standard error estimate |
| `method` | a character string indicating the method used to perform the test |
| `data.name` | a character string giving the names of the data |
| `alternative` | a character string describing the alternative hypothesis |

## Author(s)

Jordi Ocana

## References

Sanchez-Pla, A., Salicru, M. and Ocana, J. Statistical methods for the analysis of high-throughput data based on functional profiles derived from the gene ontology. Journal of Statistical Planning and Inference, 2007.

## See Also

compareGOProfiles

## Examples

```
#data(sampleProfiles)
#comparedMF <-fitGOProfile(pn=expandedWelsh01[['MF']],
#                          p0  = expandedSingh01[['MF']])
#print(comparedMF)
#print(compSummary(comparedMF))
```

---

`GOTermsList` *Functions to create and manage lists of GO terms associated with a vector of 'Entrez' identifiers*

---

## Description

These functions prepare data to be processed by the 'basicProfile' function. To create a profile a set of GOterms belonging to one or more ontologies is needed The terms belonging to each gene must be given separately so that they can be counted. This function queries the environment 'GOENTREZID2GO' with the vector of Entrez terms and formats the output into a list whose components -one per Entrez term- contain the most specific GO identifiers associated with this term.

## Usage

```
GOTermsList(LLids, onto = "any", evid = "any", na.rm = TRUE, orgPkg )
getAncestorsLst(GOtermslist, onto, unique.ancestor=TRUE, na.rm=TRUE, combine=TRUE)
getGOLevel(onto, level)
```

## Arguments

| | |
|---|---|
| `LLids` | Character vector of `Entrez` (formerly `Locuslink` identifiers) |
| `onto` | ontology to be queried using the genes list |
| `evid` | type of evidence supporting the selected GO Terms |
| `na.rm` | flag indicating if those ids returning NA must be removed from the output |
| `orgPkg` | Organism annotation package ('org.Xx.eg.db') required to obtain the GO terms associated with the Entrez identifiers |
| `GOtermslist` | List produced by a call to function `GOTermsList` |
| `unique.ancestor` | Flag to remove repeated ancestor identifiers |
| `combine` | Flag to combine ancestors |
| `level` | GO level at which the profile is built |

## Details

During the call to this function there may appear two types of NAs.

By one side if a name is not mapped in LocusLink this yields an NA that must be eliminated because nothing can be found through LL about this name

By another side if a gene is identified in LL but yields NA it seems to mean that it is not mapped in the GO

This may be eliminated but it may be worth the pity to keep track of them and to put these terms in an 'Seemingly unnanotated' category. In the case that its number was very high it migt suggest reviewing the list or reconsidering the results.

## Value

A list whose components -one per Entrez term- are character vectors with the most specific GO identifiers associated with this term

## Author(s)

Alex Sanchez

## See Also

`getAncestorsLst`

## Examples

```
#data(CD4Ids)
#simpleLLids<- as.character(c(2189,5575,5569,11)) #1 is not a Locuslink identifier
#simpleGOlist<- GOTermsList (simpleLLids, orgPkg="org.Hs.eg.db")
#print(simpleGOlist.CC<-GOTermsList (simpleLLids,"CC", orgPkg="org.Hs.eg.db"))
#print(simpleGOlist.IEA<-GOTermsList (simpleLLids,evid="IEA",na.rm=TRUE, orgPkg="org.Hs.eg.db"))
```

---

# hugoIds *Entrez Identifiers obtained from the Human Genome Organization*

---

## Description

Entrez identifiers obtained from the Human Genome Organization. They correspond to the column named 'Entrez Gene Id (mapped)' in the 'All data' table in the Hugo Genome Nomenclature web site (http://www.genenames.org/index.html)

## Usage

```
data(hugoIds)
```

## References

http://www.genenames.org/cgi-bin/hgnc_downloads.cgi

## Examples

```
data(hugoIds)
```

---

`iterEquivClust` *For each combination of the specified levels in the choosen GO ontologies, compute the equivalence threshold distance matrix and generate a dendrogram from it.*

---

## Description

For each combination of the specified levels in the choosen GO ontologies, compute the equivalence threshold distance matrix and generate a dendrogram from it.

## Usage

```
iterEquivClust(geneLists, ontos = c("BP", "MF", "CC"), ontoLevels = c(2, 3),
  trace = TRUE, onTheFlyDev = NULL, method = "complete",
  jobName = "Equivalence clustering",
  ylab = "Equivalence threshold distance", alpha = 0.05, precis = 0.001,
  ...)
```

## Arguments

| | |
|---|---|
| `geneLists` | list of character vectors, each vector stands for the gene names in a given gene set |
| `ontos` | character vector, (e.g. c("BP","MF")) indicating the GO ontologies to be analysed |
| `ontoLevels` | integer vector (e.g. 2:4) indicating the GO levels in these ontologies where the GO profiles are built |
| `trace` | boolean, the full process must be traced? Defaults to TRUE |
| `onTheFlyDev` | character, name of the graphical device where to immediately display the resulting diagrams. The appropriate names depend on the operating system. Defaults to `NULL` and then nothing is displayed. Otherwise, successive graphical windows are opened and the successive diagrams are displayed in them |
| `method` | character, one of the admissible methods in function `hclust`. Defaults to "complete" |
| `jobName` | character, main plot name, defaults to "Equivalence clustering" |
| `ylab` | character, label of the vertical axis of the plot, defaults to "Equivalence threshold distance" |
| `alpha` | simultaneous nominal significance level for the equivalence tests to be repeteadly performed, defaults to 0.05 |
| `precis` | numerical precission in the iterative search of the equivalence threshold distances, defaults to 0.001 |
| `...` | additional arguments to `hclust` |

### Value

An object of class `iterEquivCluster`. It is a list of `length(ontos)`, one element for each ontology under study. Each element of this list is itself a list of `length(ontoLevels)` with elements of class `equivClust`, standing for the cluster equivalence analysis performed for each ontology and level analysed

### Examples

```
## Not run:
data(kidneyGeneLists)
kidneyGeneLists
genListsClusters <- iterEquivClust(kidneyGeneLists, ontoLevels = 2:3,
                          jobName = "Kidney Gene Lists_Equivalence Clustering (complete)",
                                    ylab = "Equivalence threshold distance",
                                    orgPackage="org.Hs.eg.db", method = "complete")
genListsClusters[["BP"]][["Level 3"]]
class(genListsClusters[["BP"]][["Level 3"]])

## End(Not run)
```

---

## kidneyGeneLists *Gene-lists related to kidney transplantation rejection*

---

### Description

An object of class "list" containing a selected subset of 5 gene-lists related to kidney transplantation rejection, described generically as "PBTS" (Patogenic Based Transcript Sets). Each gene-list is a character vector containing "Entrez" identifiers (integer numbers) for all the genes it contains.

### Usage

```
kidneyGeneLists
```

### Format

An object of class "list" with 5 character vectors:

### Source

https://www.ualberta.ca/medicine/institutes-centres-groups/atagc/research/gene-list

`mergeProfilesLists` *Combines two lists of profiles into one*

## Description

Combines two lists of profiles, that is two lists with three components, 'MF', 'BP', 'CC' into a single one.

## Usage

```
mergeProfilesLists(profilesList1, profilesList2, emptyCats = F, profNames = NULL)
```

## Arguments

| | |
|---|---|
| `profilesList1` | First list to combine |
| `profilesList2` | Second list to combine |
| `emptyCats` | Boolean. Set to TRUE if there are empty categories that should be accounted for in any of the profiles |
| `profNames` | Names for the profiles (optional). If missing they are set to 'Frequency-1', 'Frequency-2',etc. |

## Value

A list of profiles with more than one column each.

## Author(s)

Alex Sanchez

## Examples

```
require(goProfiles)
data(prostateIds)
welsh.MF <- basicProfile (welsh01EntrezIDs[1:100], onto="MF", level=2, orgPackage="org.Hs.eg.db")
singh.MF <- basicProfile (singh01EntrezIDs[1:100], onto="MF", level=2, orgPackage="org.Hs.eg.db")
plotProfiles(welsh.MF,'Functional profiles for Welsh dataset',percentage=TRUE)
welsh.singh.MF <-mergeProfilesLists(welsh.MF, singh.MF, profNames=c("Welsh", "Singh"))
```

ngenes *Returns the number of genes that lead to this GO profile (an object of class ExpandedGOProfile, BasicGOProfile or assimilable to them)*

## Description

The information contained in one or more lists of genes may be summarized by their GO profiles, that is to say, the absolute or relative frequencies of annotations or hits in all the classes or nodes of a given leven in a given GO ontology, or by the corresponding frequencies in a selected set of nodes (possibly belonging to more than one GO level but not hierarchicaly related). This function returns the number of genes in each list that were annotated to compute the profiles

## Usage

```
ngenes(pn, i=NULL)
## Default S3 method:
ngenes(pn, i=NULL)
## S3 method for class 'numeric'
ngenes(pn, i=NULL)
## S3 method for class 'matrix'
ngenes(pn, i=NULL)
## S3 method for class 'ExpandedGOProfile'
ngenes(pn, i=NULL)
## S3 method for class 'BasicGOProfile'
ngenes(pn, i=NULL)
```

## Arguments

| | |
|---|---|
| pn | an object of class ExpandedGOProfile or BasicGOProfile representing one or more "sample" expanded GO profiles for a fixed ontology, or a numeric vector interpretable as a GO profile (expanded or not), or a frequency matrix (see the 'Details' section) |
| i | i-th profile in the case of more than one profiles. A vector with the number of genes of all profiles is returned if this argument is absent |

## Details

Given a list of n genes, and a set of s GO nodes X, Y, Z, ... in a given ontology (BP, MF or CC), its associated (contracted) "basic profile" is the frequencies vector (either absolute or relative frequencies) of annotations or hits of the n genes in each node. For a given node, say X, this frequency includes all annotations for X alone, for X and Y, for X and Z and so on. Thus, as relative frequencies, its sum is not necessarily one, or as absolute frequencies their sum is not necessarily n. On the other hand, an "expanded profile" corresponds to the frequencies in ALL OBSERVED NODE COMBINATIONS. That is, if n genes have been profiled, the expanded profile stands for the frequency of all hits EXCLUSIVELY in nodes X, Y, Z, ..., jointly with all hits simultaneously in nodes X and Y (and only in X and Y), simultaneously in X and Z, in Y and Z, ... , in X and Y and Z (and only in X,Y,Z), and so on. Thus, their sum is one.

An object of S3 class 'ExpandedGOProfile' is, essentially, a 'data.frame' object with each column representing an expanded profile. The row.names attribute codifies the node combinations and each data.frame column (say, each profile) has an attribute, 'ngenes', indicating the number of profiled genes.

## Value

A vector with the number of genes annotated in one or more GO profiles

## Author(s)

Jordi Ocana

## See Also

BasicGOProfile object, ExpandedGOProfile object

## Examples

```
require("org.Hs.eg.db")
data(prostateIds)
# To improve speed, use only the first 100 genes:
list1 <- welsh01EntrezIDs[1:100]
prof1 <- expandedProfile(list1, onto="MF", level=2, orgPackage="org.Hs.eg.db", na.rm=TRUE)$MF
length(list1)
# Only a subset of the initial gene list are annotated in the profile
ngenes(prof1)
```

---

# omimIds *Entrez identifiers for disease-related genes in the OMIM database*

---

## Description

Entrez identifiers for several lists of genes related with human disease.

`diseaseIds` contains the Entrez identifiers corresponding to disease-related genes found in the OMIM database. This list has been manually curated by Nuria Lopez-Bigas et al. who kindly provided it to us.

`morbidmapIds` contains the Entrez identifiers for all the genes in the morbidmap table. This list would correspond to disease-related genes if there had been no manual curation, as in the previous list ('diseaseIds').

`dominantIds` ontains the Entrez identifiers for dominant genes after manual curation by Nuria Lopez-Bigas who has kindly allowed us to include them in the package.

`recessiveIds` contains the Entrez identifiers for recessive genes after manual curation by Nuria Lopez-Bigas who has kindly allowed us to include them in the package.

`dominantIdsEBI` contains the Entrez identifiers for dominant genes in the EBI version of the OMIM database recovered using SRS with the term 'dominant' in the KEYWORDS field.

`recessiveIdsEBI` contains the Entrez identifiers for recessive genes in the EBI version of the OMIM database recovered using SRS with the term 'recessive' in the KEYWORDS field.

`dominantIdsNCBI` contains the Entrez identifiers for dominant genes in the NCBI version of the OMIM database recovered using ENTREZ with the term 'dominant' in the CLINICAL field.

`recessiveIdsNCBI` contains the Entrez identifiers for recessive genes in the NCBI version of the OMIM database recovered using ENTREZ with the term 'recessive' in the CLINICAL field.

## Usage

```
data(omimIds)
```

## Format

Each dataset is a character vector with a different number of elements which (should) correspond to valid Entrez identifiers

## Details

Lopez-Bigas et al. analyzed the distribution of functional categories in genes causing disease in human. They did several comparisosn which can also be done using `goProfiles`. In order to perform these comparisons we first tried to obtain the same lists of genes using standard database browsers, such as 'SRS', at the European Bioinformatics Institute, or 'Entrez', at the National Center for Biotechnological Information. Curiously both approaches provided very different lists so we asked the authors for their data and they kindly provided them to us. In order to facilitat the use of functions included in `goProfiles` we have trimmed the list of recessive and dominant genes so that (i) They become exclussive (no gene belows to both lists) (2) They are both included in the `diseaseIds` list. This eliminated 39 genes (out of 639) from the list of recessive genes and 52 genes (out of 414) from the list of dominant genes

## References

Lopez-Bigas, N., Blencowe, B.J. and Ouzounis, C.A., Highly consistent patterns for inherited human diseases at the molecular level, Bioinformatics, 2006, 22 (3), 269-277.

## Examples

```
data(omimIds)
```

---

# plotProfiles *Plot functional profiles*

---

## Description

Plots basic functional profiles created with the 'basicProfile' instruction. If several profiles have to be plot together they must be first merged using the 'mergeProfiles' function. The labels of the Y-axis of the plots are the descriptions of the GO Terms. If the label is longer than 20 characters it is truncated and ended by three dots.

## Usage

```
plotProfiles(aProf, aTitle = "Functional Profile", anOnto = NULL, percentage = FALSE,
HORIZVERT = TRUE, legendText = NULL, colores = c("white", "red"), multiplePlots = F,
multipleWindows = T, labelWidth=25,...)
```

## Arguments

| | |
|---|---|
| aProf | Functional profile to plot |
| aTitle | Title for the figures |
| anOnto | Ontology (to appear in the title) |
| percentage | Plot absolute or relative frequencies (not summing to 100) |
| HORIZVERT | Plot horizontal or vertical bars |
| legendText | Text of the legend for the plot |
| colores | Colors to be used |
| multiplePlots | Plot all profiles for a given dataset in one figure |
| multipleWindows | Open a new window after each plot |
| labelWidth | Width of Y axis labels (Names of GO categories) in the plot |
| ... | Other graphical parameters that should be passed for plotting |

## Value

The plot

## Author(s)

Alex Sanchez

## Examples

```
require(goProfiles)
data(prostateIds)
welsh.MF <- basicProfile (welsh01EntrezIDs[1:100], onto="MF", level=2, orgPackage="org.Hs.eg.db")
singh.MF <- basicProfile (singh01EntrezIDs[1:100], onto="MF", level=2, orgPackage="org.Hs.eg.db")
plotProfiles(welsh.MF,'Functional profiles for Welsh dataset',percentage=TRUE)
welsh.singh.MF <-mergeProfilesLists(welsh.MF, singh.MF, profNames=c("Welsh", "Singh"))
plotProfiles(welsh.singh.MF , percentage=TRUE, multiplePlots=TRUE, labelWidth=30)
```

printProfiles *Print functional profiles*

## Description

Prints basic functional profiles created with the 'basicProfile' instruction. Allows for several formatting operations such as truncating long labels, removing empty categories or choosing between absolute or relative frequencies. If several profiles have to be printed together they must be first merged using the 'mergeProfiles' function.

## Usage

```
printProfiles(aProf, aTitle = "Functional Profile", anOnto = NULL, percentage = FALSE,
      Width=25, emptyCats=FALSE)
```

## Arguments

| | |
|---|---|
| aProf | Functional profile to plot |
| aTitle | Title for the figures |
| anOnto | Ontology (to appear in the title) |
| percentage | Plot absolute or relative frequencies (not summing to 100) |
| Width | Maximum width for the description of GO categories |
| emptyCats | Set to 'TRUE' if empty categories should appear in the profile |

## Value

The printout

## Author(s)

Alex Sanchez

## Examples

```
require(goProfiles)
data(prostateIds)
welsh.MF <- basicProfile (welsh01EntrezIDs[1:100], onto="MF", level=2, orgPackage="org.Hs.eg.db")
singh.MF <- basicProfile (singh01EntrezIDs[1:100], onto="MF", level=2, orgPackage="org.Hs.eg.db")
printProfiles(welsh.MF,'Functional profiles for Welsh dataset',percentage=TRUE, anOnto='MF')
welsh.singh.MF <-mergeProfilesLists(welsh.MF, singh.MF, profNames=c("Welsh", "Singh"))
printProfiles(welsh.singh.MF, percentage=TRUE, emptyCats=TRUE)
```

prostateIds *Prostate cancer-related genes*

## Description

Entrez identifiers for genes related with Prostate Cancer selected from two datasets analyzed by Welsh et al. (2001) and Singh et al. (2002) respectively. The genes have been selected from freely available datasets in the internet using a standard workflow for selecting differentially expressed genes. The dataset contains 4 character vectors, each corresponding to the entrez identifiers of the genes selected at a 5% and 1% significance level from the Welsh and Singh dataset respectively.

`welsh05EntrezIDs` List of genes selected from Welsh et al. study at a 0.05 significance level.

`welsh01EntrezIDs` List of genes selected from Welsh et al. study at a 0.01 significance level.

`singh05EntrezIDs` List of genes selected from Singh et al. study at a 0.05 significance level.

`singh01EntrezIDs` List of genes selected from Singh et al. study at a 0.01 significance level.

## Usage

```
data(prostateIds)
```

## Format

Each dataset is a character vector with a different number of elements which (should) correspond to valid Entrez identifiers

## Source

- John B. Welsh, Lisa M. Sapinoso, Andrew I. Su, Suzanne G. Kern, Jessica Wang-Rodriguez, Christopher A. Moskaluk, Jr. Frierson, Henry F., and Garret M. Hampton. Analysis of Gene Expression Identifies Candidate Markers and Pharmacological Targets in Prostate Cancer. Cancer Res, 61(16):5974-5978, 2001.
- Singh, Dinesh and Febbo, Phillip G and Ross, Kenneth and Jackson, Donald G and Manola, Judith and Ladd, Christine and Tamayo, Pablo and Renshaw, Andrew A and D'Amico, Anthony V and Richie, Jerome P and Lander, Eric S and Loda, Massimo and Kantoff, Philip W and Golub, Todd R and Sellers, William R.Gene expression correlates of clinical prostate cancer behavior. Cancer Cell, 2002, Mar., 1(2) 203-209, 2002.

## Examples

```
data(prostateIds)
```

# Index


∗ **datasets**
CD4Ids, 4
clustKidneyMF2, 5
drosophila, 14
hugoIds, 29
kidneyGeneLists, 31
omimIds, 34
prostateIds, 38
∗ **hplot**
plotProfiles, 35
printProfiles, 37
∗ **htest**
basicProfile, 3
compareGeneLists, 5
compareGOProfiles, 7
compareProfilesLists, 9
compSummary, 11
equivalentGOProfiles, 15
equivSummary, 20
expandedProfile, 22
fitGOProfile, 26
∗ **manip**
conversionFunctions, 13
expandedLevel, 21
GOTermsList, 28
∗ **package**
goProfiles-package, 2
∗ **utilities**
mergeProfilesLists, 32

as.GOTerms.frame *(*conversionFunctions*)*, 13
as.GOTerms.list *(*conversionFunctions*)*, 13

basicProfile, 3, *6*
BioCpack2EntrezIDS *(*conversionFunctions*)*, 13
BioCpack2Profiles *(*conversionFunctions*)*, 13
BioCprobes2Entrez *(*conversionFunctions*)*, 13

CD4GOTermsFrame *(*CD4Ids*)*, 4
CD4GOTermsList *(*CD4Ids*)*, 4
CD4Ids, 4
CD4LLids *(*CD4Ids*)*, 4
clustKidneyMF2, 5
compareGeneLists, 5, *10*
compareGOProfiles, *6*, 7
compareProfilesLists, 9
compSummary, 11
contractedProfile, 12
conversionFunctions, 13

diseaseIds *(*omimIds*)*, 34
dominantIds *(*omimIds*)*, 34
dominantIdsEBI *(*omimIds*)*, 34
dominantIdsNCBI *(*omimIds*)*, 34
drosophila, 14
drosophilaIds *(*drosophila*)*, 14

equivalentGOProfiles, 15
equivClust, 17
equivClust2pdf, 19
equivSummary, 20
expandedLevel, 21
expandedProfile, *10*, 22
expandTerm *(*expandedLevel*)*, 21

fisherGOProfiles, 23
fitGOProfile, 26

getAncestorsLst *(*GOTermsList*)*, 28
getGOLevel *(*GOTermsList*)*, 28
goProfiles *(*goProfiles-package*)*, 2
goProfiles-package, 2
GOTermsFrame2GOTermsList *(*conversionFunctions*)*, 13
GOTermsList, 28

hugoIds, 29

iterEquivClust, 30

kidneyGeneLists, 31

mergeProfilesLists, 32
michaudIds *(*drosophila*)*, 14
morbidmapIds *(*omimIds*)*, 34

ngenes, 33

omimIds, 34
ostrinIds *(*drosophila*)*, 14

plotProfiles, 35
printProfiles, 37
prostateIds, 38

recessiveIds *(*omimIds*)*, 34
recessiveIdsEBI *(*omimIds*)*, 34
recessiveIdsNCBI *(*omimIds*)*, 34

singh01EntrezIDs *(*prostateIds*)*, 38
singh05EntrezIDs *(*prostateIds*)*, 38

welsh01EntrezIDs *(*prostateIds*)*, 38
welsh05EntrezIDs *(*prostateIds*)*, 38