funOmics

Elisa Gómez de Lope

2024-02-16

Introduction

The `funOmics` R package is a collection of functions designed to aggregate omics data into higher-level functional representations such as pathways, protein complexes, and cellular locations. This vignette provides a detailed guide on how to use the package.

Omics data analysis is a critical component of modern biomedical research. The `funOmics` package provides a tool for aggregating omics data from high-throughput experiments (e.g. transcriptomics, metabolomics, proteomics) into higher-level functional activity scores that can then be used for further analysis and modeling. This capability provides a more global view of the biological systems, reduces the dimensionality, and facilitates biological interpretation of results.

The package provides different pooling operators, such as aggregation statistics (mean, median, standard deviation, min, max), dimension-reduction derived scores (PCA, NMF, MDS, *pathifier* deregulation scores from the `pathifier` package), or test statistics (t-test, Wilcoxon test, Kolmogorov–Smirnov test) with options for adjusting parameters and settings to suit specific research questions and data types. The package is also well-documented, with detailed descriptions of each function and several examples of usage.

`funOmics` distinguishes itself from existing Bioconductor packages dedicated to pathway or gene set analysis such as GSEA and ORA (`clusterProfiler`, `fgsea`, `GSEAset`), or `GSVA`, by offering a comprehensive tool for directly aggregating diverse omics data types into higher-level functional representations, allowing the analysis of such functional representations as functional activity scores that can be modeled as input features for identifying candidate biomarkers, or in clustering strategies for patient identification. Unlike GSEA and ORA, which primarily focus on gene expression and predefined gene sets, `funOmics` accommodates various omics modalities (e.g., metabolomics, transcriptomics, proteomics), and allows users to define custom molecular sets for aggregation. Additionally, `funOmics` goes beyond `GSVA` by providing flexibility in the choice of aggregation operators, enabling users to derive interpretable functional activity scores tailored to their specific research questions. By offering a flexible and user-friendly, alternative tool for functional analysis, `funOmics` aims to contribute to the diverse array of Bioconductor packages and enhance the capabilities of the community.

Installation

Install `funOmics` from Bioconductor (release 3.19 onwards) via:

```r
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("funOmics")
```

or the pre-release and latest development version from GitHub:

```r
if (!require("devtools", quietly = TRUE))
    install.packages("devtools")
```

```
devtools::install_github("elisagdelope/funOmics")
```

Usage

Loading the Package

To use the `funOmics` R package, load it with the following command:

```
library(funOmics)
```

```
##
```

Functions `get_kegg_sets` and `short_sets_detail`

The function `get_kegg_sets` retrieves KEGG pathway gene sets for a specified organism. It fetches all pathways available for the specified organism from the KEGG database and maps the genes involved in each pathway. Currently, the function only supports choice of gene identifiers (entrez IDs, gene symbols and Ensembl IDs) for Homo sapiens (organism = "hsa") using the `org.Hs.eg.db` package.

`get_kegg_sets` has two parameters: `organism` and `geneid_type`. The parameter `organism` provides the organism abbreviation for which KEGG pathway gene sets are to be retrieved (e.g., "ecj" for E. coli). Default is "hsa" (Homo sapiens). `geneid_type` provides the type of gene IDs to provide and is only used when the organism is "hsa" (Homo sapiens). The default is "entrez"; options are "entrez", "symbol", or "ensembl".

The function `get_kegg_sets()` returns a list where each element represents a KEGG pathway gene set (i.e., a list of lists). The names of the inner lists correspond to the pathway names. For further details, the function documentation can be accessed by running the following command:

```
?funOmics::get_kegg_sets
```

The function `short_sets_detail` identifies molecular sets with sizes less than a specified threshold, and returns information about these short sets. It has two parameters: `sets` and `minsize`. The parameter `sets` is a list of molecular sets, like that obtained from the function `get_kegg_sets`. The parameter `minsize` provides the minimum size threshold for sets to be categorized as "short" and subsequently processed to extract information. Function documentation can be accessed by running:

```
?funOmics::short_sets_detail
```

Examples usage

Let's retrieve KEGG pathway gene sets for Homo sapiens with entrez IDs (default):

```
hsa_kegg_sets_entrez <- get_kegg_sets()
head(hsa_kegg_sets_entrez)
```

```
## $`2-Oxocarboxylic acid metabolism`
##  [1] "100526760" "137362"    "1431"      "162417"    "1629"      "1737"
##  [7] "1738"      "1743"      "189"       "2805"      "2806"      "2875"
## [13] "3417"      "3418"      "3419"      "3420"      "3421"      "48"
## [19] "4967"      "50"        "51166"     "5160"      "5161"      "5162"
## [25] "55526"     "55753"     "586"       "587"       "593"       "594"
## [31] "64902"     "84706"     "95"
##
## $`ABC transporters`
##  [1] "10057"  "10058"  "10060"  "10257"  "10347"  "10349"  "10350"  "10351"
##  [9] "1080"   "11194"  "1244"   "154664" "1672"   "19"     "20"     "21"
## [17] "215"    "22"     "225"    "23456"  "23457"  "23460"  "23461"  "24"
## [25] "26154"  "340273" "368"    "4363"   "5243"   "5244"   "5825"   "5826"
## [33] "64137"  "64240"  "64241"  "6833"   "6890"   "6891"   "85320"  "8647"
## [41] "8714"   "89845"  "94160"  "9429"   "9619"
##
## $`Acute myeloid leukemia`
##  [1] "10000"     "1050"      "1053"      "110117499" "11040"     "1147"
##  [7] "1436"      "1437"      "1848"      "1978"      "207"       "208"
## [13] "2209"      "2322"      "2475"      "2885"      "3265"      "3551"
## [19] "3562"      "3684"      "369"       "3728"      "3815"      "3845"
## [25] "4353"      "4609"      "4790"      "4893"      "51176"     "5290"
## [31] "5291"      "5292"      "5293"      "5295"      "5296"      "5371"
## [37] "5467"      "5594"      "5595"      "5604"      "5605"      "572"
## [43] "5894"      "5914"      "595"       "597"       "5970"      "6198"
## [49] "6199"      "6654"      "6655"      "6688"      "673"       "6774"
## [55] "6776"      "6777"      "6932"      "6934"      "7704"      "83439"
## [61] "8503"      "8517"      "861"       "862"       "8864"      "890"
## [67] "8900"      "929"
##
## $`Adherens junction`
##  [1] "1003"   "10163"  "103910" "10398"  "10458"  "10580"  "10627"  "10810"
##  [9] "11235"  "117178" "1387"   "1457"   "1459"   "1460"   "1495"   "1496"
## [17] "1499"   "1500"   "1956"   "2033"   "2064"   "2241"   "2260"   "2534"
## [25] "25945"  "283106" "29119"  "29895"  "3480"   "3643"   "387"    "4008"
## [33] "4088"   "4089"   "4233"   "4301"   "4633"   "4636"   "51176"  "51701"
## [41] "52"     "54922"  "55698"  "5594"   "5595"   "56288"  "57154"  "57493"
## [49] "5770"   "5777"   "5787"   "5792"   "5795"   "5797"   "5818"   "5819"
## [57] "58498"  "5879"   "5880"   "5881"   "5906"   "5908"   "60"     "6093"
## [65] "64750"  "6591"   "6615"   "6714"   "6885"   "6932"   "6934"   "7046"
## [73] "7048"   "7082"   "71"     "7414"   "7454"   "7525"   "81"     "81607"
## [81] "83439"  "83605"  "87"     "8826"   "889"    "8936"   "8976"   "93408"
## [89] "9411"   "9475"   "9855"   "998"    "999"
##
## $`Adipocytokine signaling pathway`
##  [1] "10000"  "10645"  "10891"  "1147"   "126129" "1374"   "1375"   "181"
##  [9] "207"    "208"    "2180"   "2181"   "2182"   "23205"  "23305"  "2475"
## [17] "2538"   "26060"  "32"     "3551"   "3667"   "3717"   "3952"   "3953"
## [25] "4790"   "4792"   "4793"   "4794"   "4852"   "5105"   "5106"   "51094"
## [33] "51422"  "51703"  "53632"  "5443"   "5465"   "5562"   "5563"   "5564"
## [41] "5565"   "5571"   "5588"   "5599"   "5601"   "5602"   "5781"   "57818"
## [49] "5970"   "6256"   "6257"   "6258"   "6513"   "6517"   "6774"   "6794"
## [57] "7124"   "7132"   "7133"   "7186"   "79602"  "81616"  "8471"   "8517"
## [65] "8660"   "8717"   "9021"   "92579"  "9370"   "948"
##
## $`Adrenergic signaling in cardiomyocytes`
##   [1] "10000"     "102723475" "10368"     "10369"     "10411"     "10488"
##   [7] "107"       "108"       "109"       "11069"     "111"       "11149"
##  [13] "112"       "113"       "114"       "115"       "1385"      "1386"
##  [19] "1388"      "1390"      "1432"      "146"       "146850"    "147"
##  [25] "148"       "148327"    "153"       "154"       "163688"    "183"
##  [31] "185"       "186"       "196883"    "207"       "208"       "23236"
##  [37] "23439"     "23533"     "27091"     "27092"     "2770"      "2771"
##  [43] "2773"      "2776"      "2778"      "28227"     "3753"      "3776"
##  [49] "3784"      "4624"      "4625"      "4633"      "4634"      "4635"
##  [55] "468"       "476"       "477"       "478"       "480"       "481"
##  [61] "482"       "483"       "486"       "487"       "488"       "489"
##  [67] "490"       "491"       "492"       "493"       "51806"     "5294"
##  [73] "5330"      "5331"      "5332"      "5350"      "5499"      "5500"
##  [79] "5501"      "55012"     "5502"      "5515"      "5516"      "5518"
##  [85] "5519"      "5520"      "5521"      "5522"      "5523"      "5525"
##  [91] "5526"      "5527"      "5528"      "5529"      "5566"      "5567"
##  [97] "5568"      "5578"      "55799"     "55844"     "5594"      "5595"
## [103] "5600"      "5603"      "59283"     "59284"     "59285"     "596"
## [109] "6262"      "6300"      "6324"      "6330"      "6331"      "6332"
## [115] "64091"     "64208"     "64764"     "6543"      "6546"      "6547"
## [121] "6548"      "70"        "7134"      "7137"      "7139"      "7168"
## [127] "7169"      "7170"      "7171"      "775"       "776"       "778"
## [133] "779"       "781"       "782"       "783"       "784"       "785"
## [139] "786"       "801"       "805"       "808"       "810"       "815"
## [145] "816"       "817"       "818"       "84699"     "90993"     "91860"
## [151] "9252"      "9254"      "93589"     "9586"
```

The KEGG molecular sets can also be retrieved for gene symbols with the `geneid_type = "symbol"` flag:

```
hsa_kegg_sets_symbol <- get_kegg_sets(geneid_type = "symbol")
hsa_kegg_sets_symbol[1]
```

```
## $`2-Oxocarboxylic acid metabolism`
##  [1] "ABHD14A-ACY1" "GOT1L1"       "CS"           "NAGS"         "DBT"
##  [6] "DLAT"         "DLD"          "DLST"         "AGXT"         "GOT1"
## [11] "GOT2"         "GPT"          "IDH1"         "IDH2"         "IDH3A"
## [16] "IDH3B"        "IDH3G"        "ACO1"         "OGDH"         "ACO2"
## [21] "AADAT"        "PDHA1"        "PDHA2"        "PDHB"         "DHTKD1"
## [26] "OGDHL"        "BCAT1"        "BCAT2"        "BCKDHA"       "BCKDHB"
## [31] "AGXT2"        "GPT2"         "ACY1"
```

And similarly for Ensembl IDs with the `geneid_type = "ensembl"` flag:

```
hsa_kegg_sets_ensembl <- get_kegg_sets(geneid_type = "ensembl")
hsa_kegg_sets_ensembl[1]
```

```
## $`2-Oxocarboxylic acid metabolism`
##  [1] "ENSG00000114786" "ENSG00000169154" "ENSG00000062485" "ENSG00000161653"
##  [5] "ENSG00000137992" "ENSG00000150768" "ENSG00000091140" "ENSG00000119689"
##  [9] "ENSG00000172482" "ENSG00000120053" "ENSG00000125166" "ENSG00000167701"
## [13] "ENSG00000138413" "ENSG00000182054" "ENSG00000166411" "ENSG00000101365"
## [17] "ENSG00000067829" "ENSG00000122729" "ENSG00000105953" "ENSG00000100412"
## [21] "ENSG00000109576" "ENSG00000131828" "ENSG00000163114" "ENSG00000168291"
## [25] "ENSG00000181192" "ENSG00000197444" "ENSG00000060982" "ENSG00000105552"
## [29] "ENSG00000248098" "ENSG00000083123" "ENSG00000113492" "ENSG00000166123"
## [33] "ENSG00000243989"
```

`get_kegg_sets` can also be used to retrieve KEGG pathway gene sets for another organism (e.g., Escherichia coli). Note that the choice of gene identifier is currently not supported for organisms other than Homo sapiens, hence the gene type is that stored by the KEGG database.

```
ecoli_kegg_sets <- get_kegg_sets(organism = "ecj")
head(ecoli_kegg_sets)
```

```
## $`2-Oxocarboxylic acid metabolism`
##  [1] "JW0070" "JW0071" "JW0073" "JW0076" "JW0077" "JW0110" "JW0111" "JW0112"
##  [9] "JW0114" "JW0710" "JW0715" "JW0716" "JW0911" "JW1122" "JW1268" "JW1372"
## [17] "JW2287" "JW2786" "JW3322" "JW3396" "JW3645" "JW3646" "JW3742" "JW3747"
## [25] "JW3929" "JW3930" "JW3984" "JW5103" "JW5553" "JW5605" "JW5606" "JW5807"
##
## $`ABC transporters`
##   [1] "JW0065" "JW0066" "JW0067" "JW0147" "JW0148" "JW0149" "JW0154" "JW0193"
##   [9] "JW0194" "JW0195" "JW0254" "JW0255" "JW0357" "JW0358" "JW0359" "JW0438"
##  [17] "JW0580" "JW0581" "JW0582" "JW0584" "JW0647" "JW0648" "JW0649" "JW0743"
##  [25] "JW0746" "JW0747" "JW0748" "JW0794" "JW0795" "JW0796" "JW0815" "JW0816"
##  [33] "JW0838" "JW0840" "JW0841" "JW0844" "JW0845" "JW0846" "JW0847" "JW0848"
##  [41] "JW0863" "JW0869" "JW0870" "JW0897" "JW0916" "JW0919" "JW1104" "JW1109"
##  [49] "JW1110" "JW1111" "JW1112" "JW1235" "JW1236" "JW1237" "JW1238" "JW1239"
##  [57] "JW1283" "JW1284" "JW1285" "JW1286" "JW1287" "JW1311" "JW1322" "JW1506"
##  [65] "JW1507" "JW1508" "JW1509" "JW1699" "JW1701" "JW1847" "JW1848" "JW1887"
##  [73] "JW1888" "JW1889" "JW1902" "JW1903" "JW1905" "JW2116" "JW2117" "JW2118"
##  [81] "JW2119" "JW2135" "JW2136" "JW2137" "JW2165" "JW2166" "JW2167" "JW2168"
##  [89] "JW2186" "JW2187" "JW2188" "JW2199" "JW2303" "JW2304" "JW2305" "JW2306"
##  [97] "JW2307" "JW2415" "JW2416" "JW2417" "JW2418" "JW2530" "JW2531" "JW2532"
## [105] "JW2652" "JW2653" "JW2654" "JW3159" "JW3160" "JW3161" "JW3162" "JW3168"
## [113] "JW3236" "JW3239" "JW3415" "JW3416" "JW3417" "JW3418" "JW3419" "JW3420"
## [121] "JW3421" "JW3422" "JW3423" "JW3425" "JW3427" "JW3428" "JW3441" "JW3442"
## [129] "JW3443" "JW3444" "JW3445" "JW3509" "JW3510" "JW3511" "JW3512" "JW3513"
## [137] "JW3538" "JW3539" "JW3540" "JW3635" "JW3703" "JW3704" "JW3705" "JW3706"
## [145] "JW3728" "JW3729" "JW3730" "JW3888" "JW3992" "JW3993" "JW3994" "JW3995"
## [153] "JW4047" "JW4048" "JW4049" "JW4066" "JW4067" "JW4186" "JW4218" "JW4247"
## [161] "JW4248" "JW4249" "JW4250" "JW5061" "JW5092" "JW5111" "JW5121" "JW5161"
## [169] "JW5162" "JW5242" "JW5366" "JW5535" "JW5544" "JW5545" "JW5752" "JW5753"
## [177] "JW5754" "JW5760" "JW5818" "JW5831" "JW5857" "JW5897"
##
## $`Acarbose and validamycin biosynthesis`
## [1] "JW2024" "JW2026" "JW3763" "JW5598"
##
## $`Alanine, aspartate and glutamate metabolism`
##  [1] "JW0030" "JW0031" "JW0474" "JW0660" "JW0812" "JW0911" "JW0999" "JW1117"
##  [9] "JW1295" "JW1488" "JW1517" "JW1750" "JW1756" "JW2287" "JW2309" "JW2558"
## [17] "JW2636" "JW2637" "JW2924" "JW3140" "JW3179" "JW3180" "JW3485" "JW3707"
## [25] "JW3722" "JW3841" "JW3932" "JW4099" "JW4135" "JW4203" "JW4204" "JW5019"
## [33] "JW5247"
##
## $`alpha-Linolenic acid metabolism`
## [1] "JW2339" "JW3794" "JW5578"
##
## $`Amino sugar and nucleotide sugar metabolism`
##  [1] "JW0264" "JW0385" "JW0663" "JW0664" "JW0665" "JW0675" "JW0740" "JW0741"
##  [9] "JW0742" "JW1087" "JW1093" "JW1105" "JW1224" "JW1605" "JW1613" "JW1632"
## [17] "JW1806" "JW1807" "JW1808" "JW2010" "JW2021" "JW2027" "JW2033" "JW2034"
## [25] "JW2037" "JW2038" "JW2248" "JW2249" "JW2250" "JW2385" "JW2410" "JW2421"
## [33] "JW2422" "JW3143" "JW3156" "JW3192" "JW3194" "JW3300" "JW3393" "JW3707"
## [41] "JW3708" "JW3940" "JW3985" "JW5372" "JW5538" "JW5599" "JW5600"
```

Main Function: `summarize_pathway_level`

You can then access the main function provided by the package, *summarize_pathway_level* with the type of pooling operator desired to be applied for each molecular set. This function has several options for adjusting parameters and settings to suit specific research questions and data types. The available aggregation operators and other parameters options are described in detail in the package documentation. You can also see the documentation for this function with the command:

```
?funOmics::summarize_pathway_level
```

Find below some examples of usage with transcriptomics data from the `airway` dataset.

Summarizing omics data in `SummarizedExperiment` format from `airway` into KEGG pathway level functional activity scores

Here we illustrate through some examples how to apply the function `summarize_pathway_level` to aggregate data in `SummarizedExperiment` format into KEGG pathway-level functional activity scores. Note that the function can also be used to summarize other types of omics data into any higher-level functional representations beyond pathways, such as protein complexes or cellular locations.

Let's first get an example dataset stored as a `SummarizedExperiment` from the `airway` package. This data represents an actual RNA sequencing experiment on four human airway smooth muscle cell lines.

```
library(SummarizedExperiment)
```

```
## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## The following object is masked from 'package:Biobase':
##
##     rowMedians

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: S4Vectors
```

```
##
## Attaching package: 'S4Vectors'
## The following object is masked from 'package:utils':
##
##     findMatches
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
## Loading required package: IRanges
## Loading required package: GenomeInfoDb
```

```
library(airway)
data(airway)
airway
```

```
## class: RangedSummarizedExperiment
## dim: 63677 8
## metadata(1): ''
## assays(1): counts
## rownames(63677): ENSG00000000003 ENSG00000000005 ... ENSG00000273492
##   ENSG00000273493
## rowData names(10): gene_id gene_name ... seq_coord_system symbol
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(9): SampleName cell ... Sample BioSample
```

The measurement data can be accessed by assay and assays. Note that `SummarizedExperiment` object can contain multiple measurement matrices (all of the same dimension), but in this case `airway` contains only one matrix of RNA sequencing data named `counts`:

```
assayNames(airway)
```

```
## [1] "counts"
```

```
head(assay(airway, "counts"))
```

```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003        679        448        873        408       1138
## ENSG00000000005          0          0          0          0          0
## ENSG00000000419        467        515        621        365        587
## ENSG00000000457        260        211        263        164        245
## ENSG00000000460         60         55         40         35         78
## ENSG00000000938          0          0          2          0          1
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003       1047        770        572
## ENSG00000000005          0          0          0
## ENSG00000000419        799        417        508
## ENSG00000000457        331        233        229
## ENSG00000000460         63         76         60
## ENSG00000000938          0          0          0
```

```
dim(assay(airway, "counts"))
```

```
## [1] 63677     8
```

The data matrix contains 63677 genes (or transcripts) and 8 samples. The features names are Ensembl identifiers, let's get a list of KEGG gene sets with Ensembl IDs through the function `get_kegg_sets` provided by `funOmics` package. Note that `get_kegg_sets` can be used to retrieve a list of KEGG gene sets from any organism available, given its abbreviation (e.g., “hsa” for Homo sapiens or “ecj” for Escherichia coli).

Since `airway` data corresponds to human samples, the parameter `geneid_type` in `get_kegg_sets` can be used to retrieve the molecular sets with Ensembl IDs, and the organism is set to default (“hsa”):

```r
kegg_sets <- get_kegg_sets(geneid_type = "ensembl")
head(kegg_sets)
```

```
## $`2-Oxocarboxylic acid metabolism`
##  [1] "ENSG00000114786" "ENSG00000169154" "ENSG00000062485" "ENSG00000161653"
##  [5] "ENSG00000137992" "ENSG00000150768" "ENSG00000091140" "ENSG00000119689"
##  [9] "ENSG00000172482" "ENSG00000120053" "ENSG00000125166" "ENSG00000167701"
## [13] "ENSG00000138413" "ENSG00000182054" "ENSG00000166411" "ENSG00000101365"
## [17] "ENSG00000067829" "ENSG00000122729" "ENSG00000105953" "ENSG00000100412"
## [21] "ENSG00000109576" "ENSG00000131828" "ENSG00000163114" "ENSG00000168291"
## [25] "ENSG00000181192" "ENSG00000197444" "ENSG00000060982" "ENSG00000105552"
## [29] "ENSG00000248098" "ENSG00000083123" "ENSG00000113492" "ENSG00000166123"
## [33] "ENSG00000243989"
##
## $`ABC transporters`
##  [1] "ENSG00000114770" "ENSG00000115657" "ENSG00000069431" "ENSG00000125257"
##  [5] "ENSG00000064687" "ENSG00000154263" "ENSG00000154258" "ENSG00000141338"
##  [9] "ENSG00000001626" "ENSG00000197150" "ENSG00000023839" "ENSG00000179869"
## [13] "ENSG00000164825" "ENSG00000165029" "ENSG00000107331" "ENSG00000167972"
## [17] "ENSG00000101986" "ENSG00000131269" "ENSG00000173208" "ENSG00000135776"
## [21] "ENSG00000150967" "ENSG00000154262" "ENSG00000154265" "ENSG00000198691"
## [25] "ENSG00000144452" "ENSG00000004846" "ENSG00000091262" "ENSG00000103222"
## [29] "ENSG00000085563" "ENSG00000005471" "ENSG00000117528" "ENSG00000119688"
## [33] "ENSG00000172350" "ENSG00000138075" "ENSG00000143921" "ENSG00000006071"
## [37] "ENSG00000168394" "ENSG00000204267" "ENSG00000121270" "ENSG00000073734"
## [41] "ENSG00000108846" "ENSG00000124574" "ENSG00000140798" "ENSG00000118777"
## [45] "ENSG00000160179"
##
## $`Acute myeloid leukemia`
##  [1] "ENSG00000117020" "ENSG00000245848" "ENSG00000092067" "ENSG00000278139"
##  [5] "ENSG00000102096" "ENSG00000213341" "ENSG00000182578" "ENSG00000164400"
##  [9] "ENSG00000139318" "ENSG00000187840" "ENSG00000142208" "ENSG00000105221"
## [13] "ENSG00000150337" "ENSG00000122025" "ENSG00000198793" "ENSG00000177885"
## [17] "ENSG00000174775" "ENSG00000104365" "ENSG00000164399" "ENSG00000169896"
## [21] "ENSG00000078061" "ENSG00000173801" "ENSG00000157404" "ENSG00000133703"
## [25] "ENSG00000005381" "ENSG00000136997" "ENSG00000109320" "ENSG00000213281"
## [29] "ENSG00000138795" "ENSG00000121879" "ENSG00000051382" "ENSG00000137193"
## [33] "ENSG00000171608" "ENSG00000145675" "ENSG00000105647" "ENSG00000140464"
## [37] "ENSG00000112033" "ENSG00000100030" "ENSG00000102882" "ENSG00000169032"
## [41] "ENSG00000126934" "ENSG00000002330" "ENSG00000132155" "ENSG00000131759"
## [45] "ENSG00000110092" "ENSG00000140379" "ENSG00000173039" "ENSG00000108443"
## [49] "ENSG00000175634" "ENSG00000115904" "ENSG00000100485" "ENSG00000066336"
## [53] "ENSG00000157764" "ENSG00000168610" "ENSG00000126561" "ENSG00000173757"
## [57] "ENSG00000081059" "ENSG00000148737" "ENSG00000109906" "ENSG00000152284"
## [61] "ENSG00000117461" "ENSG00000269335" "ENSG00000159216" "ENSG00000079102"
## [65] "ENSG00000132326" "ENSG00000145386" "ENSG00000133101" "ENSG00000170458"
##
## $`Adherens junction`
##  [1] "ENSG00000179776" "ENSG00000158195" "ENSG00000118680" "ENSG00000101335"
##  [5] "ENSG00000175866" "ENSG00000095637" "ENSG00000101608" "ENSG00000132970"
##  [9] "ENSG00000114209" "ENSG00000117155" "ENSG00000005339" "ENSG00000101266"
## [13] "ENSG00000070770" "ENSG00000204435" "ENSG00000044115" "ENSG00000066032"
## [17] "ENSG00000168036" "ENSG00000198561" "ENSG00000146648" "ENSG00000100393"
## [21] "ENSG00000141736" "ENSG00000151422" "ENSG00000077782" "ENSG00000010810"
## [25] "ENSG00000177707" "ENSG00000254598" "ENSG00000183230" "ENSG00000180209"
## [29] "ENSG00000140443" "ENSG00000171105" "ENSG00000067560" "ENSG00000136153"
## [33] "ENSG00000166949" "ENSG00000141646" "ENSG00000105976" "ENSG00000130396"
## [37] "ENSG00000111245" "ENSG00000215375" "ENSG00000138795" "ENSG00000087095"
## [41] "ENSG00000143727" "ENSG00000105538" "ENSG00000157927" "ENSG00000100030"
## [45] "ENSG00000102882" "ENSG00000148498" "ENSG00000198742" "ENSG00000173706"
## [49] "ENSG00000196396" "ENSG00000111679" "ENSG00000127329" "ENSG00000142949"
## [53] "ENSG00000149177" "ENSG00000173482" "ENSG00000110400" "ENSG00000130202"
## [57] "ENSG00000106631" "ENSG00000136238" "ENSG00000128340" "ENSG00000169750"
## [61] "ENSG00000116473" "ENSG00000127314" "ENSG00000075624" "ENSG00000067900"
## [65] "ENSG00000108854" "ENSG00000019549" "ENSG00000124216" "ENSG00000197122"
## [69] "ENSG00000135341" "ENSG00000081059" "ENSG00000148737" "ENSG00000106799"
## [73] "ENSG00000163513" "ENSG00000104067" "ENSG00000184009" "ENSG00000035403"
## [77] "ENSG00000015285" "ENSG00000176105" "ENSG00000130402" "ENSG00000143217"
## [81] "ENSG00000152284" "ENSG00000136280" "ENSG00000072110" "ENSG00000140575"
## [85] "ENSG00000001631" "ENSG00000112290" "ENSG00000106299" "ENSG00000106436"
## [89] "ENSG00000137962" "ENSG00000134318" "ENSG00000006607" "ENSG00000070831"
## [93] "ENSG00000039068"
##
## $`Adipocytokine signaling pathway`
##  [1] "ENSG00000117020" "ENSG00000110931" "ENSG00000109819" "ENSG00000213341"
##  [5] "ENSG00000169169" "ENSG00000110090" "ENSG00000205560" "ENSG00000159723"
##  [9] "ENSG00000142208" "ENSG00000105221" "ENSG00000151726" "ENSG00000123983"
## [13] "ENSG00000068366" "ENSG00000103740" "ENSG00000164398" "ENSG00000198793"
## [17] "ENSG00000131482" "ENSG00000157500" "ENSG00000076555" "ENSG00000104365"
## [21] "ENSG00000169047" "ENSG00000096968" "ENSG00000174697" "ENSG00000116678"
## [25] "ENSG00000109320" "ENSG00000100906" "ENSG00000104825" "ENSG00000146232"
## [29] "ENSG00000122585" "ENSG00000124253" "ENSG00000100889" "ENSG00000159346"
## [33] "ENSG00000106617" "ENSG00000197142" "ENSG00000115592" "ENSG00000115138"
## [37] "ENSG00000186951" "ENSG00000132356" "ENSG00000162409" "ENSG00000111725"
## [41] "ENSG00000131791" "ENSG00000181929" "ENSG00000065675" "ENSG00000107643"
## [45] "ENSG00000050748" "ENSG00000109339" "ENSG00000179295" "ENSG00000152254"
## [49] "ENSG00000173039" "ENSG00000186350" "ENSG00000204231" "ENSG00000143171"
## [53] "ENSG00000117394" "ENSG00000181856" "ENSG00000168610" "ENSG00000118046"
## [57] "ENSG00000232810" "ENSG00000067182" "ENSG00000028137" "ENSG00000127191"
## [61] "ENSG00000006831" "ENSG00000130377" "ENSG00000133124" "ENSG00000269335"
## [65] "ENSG00000185950" "ENSG00000102871" "ENSG00000184557" "ENSG00000141349"
## [69] "ENSG00000181092" "ENSG00000135218"
##
## $`Adrenergic signaling in cardiomyocytes`
##   [1] "ENSG00000117020" "ENSG00000276289" "ENSG00000006116" "ENSG00000166862"
##   [5] "ENSG00000079337" "ENSG00000107175" "ENSG00000164742" "ENSG00000078295"
##   [9] "ENSG00000138031" "ENSG00000091428" "ENSG00000173175" "ENSG00000112276"
##  [13] "ENSG00000174233" "ENSG00000121281" "ENSG00000155897" "ENSG00000162104"
##  [17] "ENSG00000118260" "ENSG00000115966" "ENSG00000213676" "ENSG00000095794"
##  [21] "ENSG00000112062" "ENSG00000171873" "ENSG00000276231" "ENSG00000170214"
##  [25] "ENSG00000120907" "ENSG00000143578" "ENSG00000043591" "ENSG00000169252"
##  [29] "ENSG00000169885" "ENSG00000135744" "ENSG00000144891" "ENSG00000180772"
##  [33] "ENSG00000129467" "ENSG00000142208" "ENSG00000105221" "ENSG00000182621"
##  [37] "ENSG00000101892" "ENSG00000141506" "ENSG00000075429" "ENSG00000075461"
##  [41] "ENSG00000127955" "ENSG00000114353" "ENSG00000065135" "ENSG00000156052"
##  [45] "ENSG00000087460" "ENSG00000167393" "ENSG00000180509" "ENSG00000082482"
##  [49] "ENSG00000053918" "ENSG00000197616" "ENSG00000092054" "ENSG00000111245"
##  [53] "ENSG00000160808" "ENSG00000198336" "ENSG00000128272" "ENSG00000163399"
##  [57] "ENSG00000018625" "ENSG00000105409" "ENSG00000132681" "ENSG00000143153"
##  [61] "ENSG00000129244" "ENSG00000069849" "ENSG00000137731" "ENSG00000196296"
##  [65] "ENSG00000174437" "ENSG00000074370" "ENSG00000070961" "ENSG00000157087"
##  [69] "ENSG00000067842" "ENSG00000058668" "ENSG00000178372" "ENSG00000105851"
##  [73] "ENSG00000137841" "ENSG00000149782" "ENSG00000101333" "ENSG00000198523"
##  [77] "ENSG00000172531" "ENSG00000213639" "ENSG00000186298" "ENSG00000092020"
##  [81] "ENSG00000135447" "ENSG00000113575" "ENSG00000104695" "ENSG00000105568"
##  [85] "ENSG00000137713" "ENSG00000221914" "ENSG00000156475" "ENSG00000074211"
##  [89] "ENSG00000073711" "ENSG00000066027" "ENSG00000068971" "ENSG00000078304"
##  [93] "ENSG00000112640" "ENSG00000154001" "ENSG00000072062" "ENSG00000142875"
##  [97] "ENSG00000165059" "ENSG00000154229" "ENSG00000157445" "ENSG00000175470"
## [101] "ENSG00000100030" "ENSG00000102882" "ENSG00000185386" "ENSG00000156711"
## [105] "ENSG00000142408" "ENSG00000105605" "ENSG00000130433" "ENSG00000171791"
## [109] "ENSG00000198626" "ENSG00000188130" "ENSG00000105711" "ENSG00000177098"
## [113] "ENSG00000183873" "ENSG00000136546" "ENSG00000121577" "ENSG00000132429"
## [117] "ENSG00000182158" "ENSG00000118160" "ENSG00000183023" "ENSG00000100678"
## [121] "ENSG00000090020" "ENSG00000159251" "ENSG00000114854" "ENSG00000129991"
## [125] "ENSG00000118194" "ENSG00000140416" "ENSG00000198467" "ENSG00000143549"
## [129] "ENSG00000167460" "ENSG00000151067" "ENSG00000157388" "ENSG00000102001"
## [133] "ENSG00000081248" "ENSG00000153956" "ENSG00000067191" "ENSG00000165995"
## [137] "ENSG00000167535" "ENSG00000182389" "ENSG00000108878" "ENSG00000198668"
## [141] "ENSG00000143933" "ENSG00000160014" "ENSG00000178363" "ENSG00000070808"
## [145] "ENSG00000058404" "ENSG00000145349" "ENSG00000148660" "ENSG00000060566"
## [149] "ENSG00000157613" "ENSG00000129007" "ENSG00000100784" "ENSG00000007402"
## [153] "ENSG00000151062" "ENSG00000146592"
```

Example usage 1: Summarize `airway` omics data into dimension-reduction derived activity scores at KEGG pathway level. The dimension-reduction operators implemented in `funOmics` include PCA (Principal Component Analysis), NMF (Non-Negative Matrix Factorization), MDS (Multidimensional Scaling), and *pathifier* deregulation scores from the `pathifier` package derived from principal curves.

Now, let's summarize the counts data using PCA. The PCA-aggregated activity scores values represent the projection of the overall expression of all genes in each pathway onto the first principal component. For this example, let's use the default minimum size of sets (10). Note that when default value for `minsize` parameter is used, it is not necessary to assign a value for this parameter in the function call:

```
pathway_activity_pca <- summarize_pathway_level(assay(airway, "counts"), kegg_sets, type = "pca")
```

```
##
## 359 functional sets read.

## iteration 100

## iteration 200

## iteration 300

## 344 successful functional aggregations over minsize

## 15 failed functional aggregations under minsize

## Functional activity score matrix has dimensions: 344 , 8
```

From the original `airway` data containing 63677 genes (transcripts) and 8 samples, the function `summarize_pathway_level` has generated a pathway-level activity score for each of the 8 samples, for 343 KEGG pathways containing more than 10 genes (16 failed functional aggregations under `minsize`). Let's see how this matrix looks like:

```r
print(head(pathway_activity_pca))
```

```
##                                           SRR1039508 SRR1039509 SRR1039512
## 2-Oxocarboxylic acid metabolism          -0.06702655   1.980602  -2.253388
## ABC transporters                         -0.62811819   3.493143  -4.375987
## Acute myeloid leukemia                   -1.68843459  -2.745095   1.499643
## Adherens junction                        -3.76905749  -4.338354   2.688612
## Adipocytokine signaling pathway          -1.65951433  -1.936742   1.463798
## Adrenergic signaling in cardiomyocytes -2.43770816  -4.671855   3.295351
##                                          SRR1039513 SRR1039516 SRR1039517
## 2-Oxocarboxylic acid metabolism            6.875470  -2.670499  -8.032398
## ABC transporters                           5.828548  -4.399747  -4.392270
## Acute myeloid leukemia                    -7.342374   2.162537  11.748711
## Adherens junction                         -8.418050   3.224066  15.496050
## Adipocytokine signaling pathway           -7.704634   2.383409  11.647702
## Adrenergic signaling in cardiomyocytes -10.499038   5.729656  15.768629
##                                          SRR1039520 SRR1039521
## 2-Oxocarboxylic acid metabolism            2.713740  1.4534992
## ABC transporters                           1.040788  3.4336426
## Acute myeloid leukemia                    -4.224764  0.5897765
## Adherens junction                         -5.052887  0.1696201
## Adipocytokine signaling pathway           -4.492711  0.2986920
## Adrenergic signaling in cardiomyocytes  -4.075372 -3.1096629
```

The resulting matrix of higher-level functional representations looks very similar to the original one, except that the original had many more features (63677 instead of 343). This reduction in dimensionality can facilitate the interpretation of the data and the identification of patterns in the samples.

In this illustrative example, the RNA sequencing data in `airway` package has been directly summarized or aggregated by the `summarize_pathway_level` function of the `funOmics` package without intermediate processing. Depending on the type of omics data, you may want to apply corresponding processing steps to the omics abundance matrix prior to the aggregation into higher level functional features. For instance, it is common practice to filter out rows (genes or features) with low counts when analyzing transcriptomics data. This filtering step is often performed to remove genes that are expressed at very low levels and may not be biologically relevant or reliable.

Some aggregation methods, may also have specific assumptions or requirements on the input data. Let's see another example where some filtering is indeed necessary.

`funOmics` allows to generate aggregated representations using dimension-reduction derived scores from the NMF (Non-Negative Matrix Factorization) method. The NMF-aggregated activity scores values represent the weight (or contribution) of a single underlying basis component or latent factor contributing to the pathway activity (or higher level functional structure in use) for each sample in your data set. Rank=1 is used for the basis matrix in the internal NMF dimension-reduction.

Notably, the NMF method does not allow for negative values or null rows in the input matrix. Transcriptomics data in the `airway` dataset are measured as counts, hence the matrix presumably does not contain negative values, but it may contain null rows. To avoid errors, we can filter out rows with less than 10 counts across all samples before applying the NMF method:

```r
print(any(assay(airway, "counts")[rowSums(assay(airway, "counts")) <0, ]))
```

```
## [1] FALSE
```

```r
X <- assay(airway, "counts")[rowSums(assay(airway, "counts")) >= 10, ]
```

Let's summarize the filtered counts data using NMF method:

```r
pathway_activity_nmf <- summarize_pathway_level(X, kegg_sets, type = "nmf") # note that the NMF operati
```

```
##
## 359 functional sets read.

## iteration 100

## iteration 200

## iteration 300

## 340 successful functional aggregations over minsize

## 19 failed functional aggregations under minsize

## Functional activity score matrix has dimensions: 340 , 8
```

```r
print(paste("Pathway activity score matrix has dimensions:", nrow(pathway_activity_nmf), ",", ncol(pathw
```

```
## [1] "Pathway activity score matrix has dimensions: 340 , 8"
```

Let's see how this matrix looks like:

```r
head(pathway_activity_nmf)
```

```
##                                         SRR1039508 SRR1039509 SRR1039512
## 2-Oxocarboxylic acid metabolism         0.25510801 0.24907840 0.30336668
## ABC transporters                        0.14657216 0.14153830 0.18241952
## Acute myeloid leukemia                  0.12141031 0.12443504 0.15009114
## Adherens junction                       0.03275327 0.03748509 0.04486956
## Adipocytokine signaling pathway         0.20786710 0.22531725 0.23257091
## Adrenergic signaling in cardiomyocytes 0.02707585 0.02996081 0.03504904
##                                         SRR1039513 SRR1039516 SRR1039517
## 2-Oxocarboxylic acid metabolism         0.18367951 0.31445548 0.38954754
## ABC transporters                        0.11451785 0.20131920 0.22893987
## Acute myeloid leukemia                  0.10556670 0.14733207 0.20353586
## Adherens junction                       0.03481751 0.03822019 0.07716273
## Adipocytokine signaling pathway         0.17018762 0.25451406 0.37516128
## Adrenergic signaling in cardiomyocytes 0.02461770 0.03373750 0.05566273
##                                         SRR1039520 SRR1039521
## 2-Oxocarboxylic acid metabolism         0.22570140 0.25690708
## ABC transporters                        0.15339158 0.18120393
## Acute myeloid leukemia                  0.11304698 0.15061145
## Adherens junction                       0.03185635 0.04591787
## Adipocytokine signaling pathway         0.18211029 0.25379765
## Adrenergic signaling in cardiomyocytes 0.02673446 0.03261790
```

In this example, `summarize_pathway_level` has generated a pathway-level activity score for each of the 8 samples, for 340 KEGG pathways containing more than 10 genes (here 19 failed functional aggregations under `minsize`, slightly more than for the PCA since some genes were removed in the filtering). Note that the resulting matrix looks similar to that of the previous example in terms of shape and format, but the values are derived from the NMF dimension-reduction method instead of the PCA method. Same analogies apply for other types of aggregation operator; only the interpretation of the resulting functional activity scores will change.

Note that beyond pre-processing, you can also post-process the resulting summarized matrix as you see appropriate for your analyses and workflows.

Integrating the pathway-level activity scores with the `airway` `SummarizedExperiment` object in a `MultiAssayExperiment` object The resulting matrix of pathway-level activity scores can be further analyzed as an independent dataset, or can also be integrated with the `airway` `SummarizedExperiment` object in a `MultiAssayExperiment` structure (note that `SummarizedExperiment` can simultaneously manage several experimental assays only if they have the same dimensions, which is not the case here, hence the need for a `MultiAssayExperiment` object). The MultiAssayExperiment library has to be loaded, and a MultiAssayExperiment (`airwayMultiAssay`) can be created and filled with a list of assays-like matrices that may have different dimensions. Here, `airwayMultiAssay` contains the `counts` and the recently generated KEGG pathway activity scores by NMF and PCA pooling.

```
library(MultiAssayExperiment)
assays_list <- list( counts = assay(airway, "counts"), kegg_nmf_agg = pathway_activity_nmf, kegg_pca_agg
airwayMultiAssay <- MultiAssayExperiment(experiments=assays_list)
colData(airwayMultiAssay) <- colData(airway)
airwayMultiAssay
```

```
## A MultiAssayExperiment object of 3 listed
##  experiments with user-defined names and respective classes.
##  Containing an ExperimentList class object of length 3:
##  [1] counts: matrix with 63677 rows and 8 columns
##  [2] kegg_nmf_agg: matrix with 340 rows and 8 columns
##  [3] kegg_pca_agg: matrix with 344 rows and 8 columns
## Functionality:
##  experiments() - obtain the ExperimentList instance
##  colData() - the primary/phenotype DataFrame
##  sampleMap() - the sample coordination DataFrame
##  `$`, `[`, `[[` - extract colData columns, subset, or experiment
##  *Format() - convert into a long or wide DataFrame
##  assays() - convert ExperimentList to a SimpleList of matrices
##  exportClass() - save data to flat files
```

Example usage 2: Summarize `airway` omics data with summary statistics and a minimum size of the KEGG gene sets Here we will apply the function `summarize_pathway_level` to summarize pathway activity using the mean pooling aggregation for those sets containing at least 12 genes. Remember that you can adjust the parameters `minsize` and `type` of aggregation as desired.

```
min <- 12
pathway_activity <- summarize_pathway_level(assay(airway, "counts"), kegg_sets, type = "mean", minsize =
```

```
##
## 359 functional sets read.

## iteration 100

## iteration 200

## iteration 300

## 342 successful functional aggregations over minsize

## 17 failed functional aggregations under minsize

## Functional activity score matrix has dimensions: 342 , 8
```

```
print(paste("Pathway activity score matrix has dimensions:", nrow(pathway_activity), ",", ncol(pathway_a
## [1] "Pathway activity score matrix has dimensions: 342 , 8"
```

Now from the original `airway` data of dimensions 63677 genes (transcripts) x 8 samples, `summarize_pathway_level` has generated through mean-pooling a pathway-level activity score matrix of 341 pathways x 8 samples, for gene sets containing more than 12 genes.

```r
print(head(pathway_activity))
```

```
##                                         SRR1039508 SRR1039509 SRR1039512
## 2-Oxocarboxylic acid metabolism           709.0000   692.2424   843.1818
## ABC transporters                          645.8000   623.6222   803.7556
## Acute myeloid leukemia                    989.1045  1013.7612  1222.8209
## Adherens junction                        3068.6452  3511.9677  4203.8602
## Adipocytokine signaling pathway           961.7143  1042.4143  1075.9286
## Adrenergic signaling in cardiomyocytes   1299.2368  1437.7171  1681.8289
##                                         SRR1039513 SRR1039516 SRR1039517
## 2-Oxocarboxylic acid metabolism           510.4848   873.9394   1082.636
## ABC transporters                          504.5556   886.9778   1008.644
## Acute myeloid leukemia                    860.0448  1200.2985   1658.224
## Adherens junction                        3262.0860  3580.9355   7229.441
## Adipocytokine signaling pathway           787.3714  1177.4857   1735.643
## Adrenergic signaling in cardiomyocytes   1181.2895  1618.9079   2671.007
##                                         SRR1039520 SRR1039521
## 2-Oxocarboxylic acid metabolism           627.2727    714.000
## ABC transporters                          675.8222    798.400
## Acute myeloid leukemia                    921.0448   1227.000
## Adherens junction                        2984.6129   4302.065
## Adipocytokine signaling pathway           842.5571   1174.186
## Adrenergic signaling in cardiomyocytes   1282.8684   1565.164
```

In this example, 18 of the gene sets in `kegg_sets` have size < 12. You can then use the function `short_sets_detail` to get information about which pathways have been left out, how many genes they had, and which genes are involved in these shorter sets:

```r
short_sets <- short_sets_detail(kegg_sets, min)
print(short_sets$short_sets)
```

```
##  [1] "Biotin metabolism"
##  [2] "Caffeine metabolism"
##  [3] "D-Amino acid metabolism"
##  [4] "Neomycin, kanamycin and gentamicin biosynthesis"
##  [5] "Phenylalanine, tyrosine and tryptophan biosynthesis"
##  [6] "Phosphonate and phosphinate metabolism"
##  [7] "Riboflavin metabolism"
##  [8] "Sulfur metabolism"
##  [9] "Sulfur relay system"
## [10] "Ubiquinone and other terpenoid-quinone biosynthesis"
## [11] "Valine, leucine and isoleucine biosynthesis"
## [12] "Virion - Adenovirus"
## [13] "Virion - Flavivirus"
## [14] "Virion - Herpesvirus"
## [15] "Virion - Human immunodeficiency virus"
## [16] "Virion - Rotavirus"
## [17] "Vitamin B6 metabolism"
```

```r
print(short_sets$short_sets_lengths)
```

```
##                                   Biotin metabolism
##                                                   3
##                                 Caffeine metabolism
##                                                   6
##                             D-Amino acid metabolism
##                                                   6
##     Neomycin, kanamycin and gentamicin biosynthesis
##                                                   5
## Phenylalanine, tyrosine and tryptophan biosynthesis
##                                                   6
##              Phosphonate and phosphinate metabolism
##                                                   6
##                               Riboflavin metabolism
##                                                   8
##                                   Sulfur metabolism
##                                                  10
##                                 Sulfur relay system
##                                                   8
## Ubiquinone and other terpenoid-quinone biosynthesis
##                                                  11
##         Valine, leucine and isoleucine biosynthesis
##                                                   4
##                                 Virion - Adenovirus
##                                                   4
##                                 Virion - Flavivirus
##                                                   4
##                                Virion - Herpesvirus
##                                                   9
##               Virion - Human immunodeficiency virus
##                                                   5
##                                  Virion - Rotavirus
##                                                   2
##                               Vitamin B6 metabolism
##                                                   6
```

```r
print(short_sets$short_sets_molecules)
```

```
## $`Biotin metabolism`
## [1] "ENSG00000159267" "ENSG00000151093" "ENSG00000169814"
##
## $`Caffeine metabolism`
## [1] "ENSG00000156006" "ENSG00000140505" "ENSG00000255974" "ENSG00000198077"
## [5] "ENSG00000158125" "ENSG00000171428"
##
## $`D-Amino acid metabolism`
## [1] "ENSG00000110887" "ENSG00000135423" "ENSG00000115419" "ENSG00000167720"
## [5] "ENSG00000133943" "ENSG00000203797"
##
## $`Neomycin, kanamycin and gentamicin biosynthesis`
## [1] "ENSG00000106633" "ENSG00000156515" "ENSG00000159399" "ENSG00000160883"
## [5] "ENSG00000156510"
##
## $`Phenylalanine, tyrosine and tryptophan biosynthesis`
## [1] "ENSG00000169154" "ENSG00000104951" "ENSG00000120053" "ENSG00000125166"
## [5] "ENSG00000171759" "ENSG00000198650"
```

```
##
## $`Phosphonate and phosphinate metabolism`
## [1] "ENSG00000134255" "ENSG00000161217" "ENSG00000111666" "ENSG00000185813"
## [5] "ENSG00000138018" "ENSG00000102230"
##
## $`Riboflavin metabolism`
## [1] "ENSG00000197594" "ENSG00000154269" "ENSG00000143727" "ENSG00000134575"
## [5] "ENSG00000102575" "ENSG00000135002" "ENSG00000090013" "ENSG00000160688"
##
## $`Sulfur metabolism`
##  [1] "ENSG00000162813" "ENSG00000105755" "ENSG00000128309" "ENSG00000104331"
##  [5] "ENSG00000137767" "ENSG00000139531" "ENSG00000128311" "ENSG00000143416"
##  [9] "ENSG00000198682" "ENSG00000138801"
##
## $`Sulfur relay system`
## [1] "ENSG00000124217" "ENSG00000174177" "ENSG00000164172" "ENSG00000128309"
## [5] "ENSG00000128311" "ENSG00000167118" "ENSG00000142544" "ENSG00000244005"
##
## $`Ubiquinone and other terpenoid-quinone biosynthesis`
##  [1] "ENSG00000167186" "ENSG00000196715" "ENSG00000181019" "ENSG00000115486"
##  [5] "ENSG00000173085" "ENSG00000158104" "ENSG00000119723" "ENSG00000132423"
##  [9] "ENSG00000198650" "ENSG00000167397" "ENSG00000110871"
##
## $`Valine, leucine and isoleucine biosynthesis`
## [1] "ENSG00000135094" "ENSG00000139410" "ENSG00000060982" "ENSG00000105552"
##
## $`Virion - Adenovirus`
## [1] "ENSG00000154639" "ENSG00000117335" "ENSG00000121594" "ENSG00000114013"
##
## $`Virion - Flavivirus`
## [1] "ENSG00000104938" "ENSG00000113249" "ENSG00000090659" "ENSG00000092445"
##
## $`Virion - Herpesvirus`
## [1] "ENSG00000121716" "ENSG00000085514" "ENSG00000119912" "ENSG00000197081"
## [5] "ENSG00000161638" "ENSG00000259207" "ENSG00000110400" "ENSG00000130202"
## [9] "ENSG00000157873"
##
## $`Virion - Human immunodeficiency virus`
## [1] "ENSG00000104938" "ENSG00000160791" "ENSG00000090659" "ENSG00000121966"
## [5] "ENSG00000010610"
##
## $`Virion - Rotavirus`
## [1] "ENSG00000164171" "ENSG00000150093"
##
## $`Vitamin B6 metabolism`
## [1] "ENSG00000135069" "ENSG00000138356" "ENSG00000144362" "ENSG00000108439"
## [5] "ENSG00000241360" "ENSG00000160209"
```

Other summary statistics can be used for the aggregation, such as median, standard deviation, min, or max. See below some more examples with varying number of genes in the gene sets:

```
min <- 15
pathway_activity <- summarize_pathway_level(assay(airway, "counts"), kegg_sets, type = "sd", minsize = 
```

```
##
```

```
## 359 functional sets read.

## iteration 100

## iteration 200

## iteration 300

## 339 successful functional aggregations over minsize

## 20 failed functional aggregations under minsize

## Functional activity score matrix has dimensions: 339 , 8
```

```
print(paste("Pathway activity score matrix has dimensions:", nrow(pathway_activity), ",", ncol(pathway_a
```

```
## [1] "Pathway activity score matrix has dimensions: 339 , 8"
```

```
head(pathway_activity)
```

```
##                                           SRR1039508 SRR1039509 SRR1039512
## 2-Oxocarboxylic acid metabolism             897.8148   933.7587   1176.073
## ABC transporters                           1343.7024  1247.1723   1700.936
## Acute myeloid leukemia                     1540.3614  1571.4687   2372.710
## Adherens junction                          6891.7930  8717.6871   9705.484
## Adipocytokine signaling pathway            1311.9107  1537.7371   1451.020
## Adrenergic signaling in cardiomyocytes     3465.9881  5059.8960   4655.470
##                                           SRR1039513 SRR1039516 SRR1039517
## 2-Oxocarboxylic acid metabolism             700.9372   1235.497   1476.768
## ABC transporters                           1034.8811   2075.315   2100.908
## Acute myeloid leukemia                     1613.5125   2178.260   3040.191
## Adherens junction                          8987.2540   7424.228  21500.638
## Adipocytokine signaling pathway            1115.5480   1741.506   2625.365
## Adrenergic signaling in cardiomyocytes     3869.2012   4542.750   9635.093
##                                           SRR1039520 SRR1039521
## 2-Oxocarboxylic acid metabolism             818.0049   958.0687
## ABC transporters                           1482.2698  1751.8586
## Acute myeloid leukemia                     1703.6101  2032.9305
## Adherens junction                          6707.2493 10638.9590
## Adipocytokine signaling pathway            1140.2466  1726.5103
## Adrenergic signaling in cardiomyocytes     3483.4342  4787.7482
```

```
min <- 7
pathway_activity <- summarize_pathway_level(assay(airway, "counts"), kegg_sets, type = "median", minsiz
```

```
##
## 359 functional sets read.

## iteration 100

## iteration 200

## iteration 300

## 347 successful functional aggregations over minsize

## 12 failed functional aggregations under minsize

## Functional activity score matrix has dimensions: 347 , 8
```

```
print(paste("Pathway activity score matrix has dimensions:", nrow(pathway_activity), ",", ncol(pathway_a
```

```
## [1] "Pathway activity score matrix has dimensions: 347 , 8"
```

```r
head(pathway_activity)
```

```
##                                          SRR1039508 SRR1039509 SRR1039512
## 2-Oxocarboxylic acid metabolism               424.0      553.0        444
## ABC transporters                              136.0       92.0        225
## Acute myeloid leukemia                        555.0      566.0        700
## Adherens junction                            1004.0     1073.0       1212
## Adipocytokine signaling pathway               542.5      611.0        623
## Adrenergic signaling in cardiomyocytes        157.0      132.5        182
##                                          SRR1039513 SRR1039516 SRR1039517
## 2-Oxocarboxylic acid metabolism               405.0      518.0      780.0
## ABC transporters                               76.0      224.0      138.0
## Acute myeloid leukemia                        393.0      663.0      767.0
## Adherens junction                             783.0     1232.0     1793.0
## Adipocytokine signaling pathway               442.5      634.5      835.0
## Adrenergic signaling in cardiomyocytes        111.5      198.5      214.5
##                                          SRR1039520 SRR1039521
## 2-Oxocarboxylic acid metabolism               375.0      614.0
## ABC transporters                              155.0       86.0
## Acute myeloid leukemia                        504.0      618.0
## Adherens junction                             980.0     1145.0
## Adipocytokine signaling pathway               480.0      616.5
## Adrenergic signaling in cardiomyocytes        168.5      180.0
```

Example usage 3: Summarize airway omics data with test statistics Using the same `airway` data and gene sets `kegg_sets` from previous examples, let's generate aggregated representations using test statistics. These operators allow to compare the measurements for each sample between the molecules in each functional set and the molecules not in the given functional set. They may help identify functionally related genes/molecules that exhibit coordinated or significant deviations in their expression patterns across samples. Currently, the implemented available statistical tests in `funOmics` are the t-test, Wilcoxon test, and Kolmogorov–Smirnov test.

When using test statistics, one has to be mindful about the assumptions of the test and the distribution of the data. For instance, the t-test assumes that the data is normally distributed and compares the means of two groups; the Wilcoxon test assumes that the data is continuous and symmetric, and compares the medians of two groups; the Kolmogorov–Smirnov test is a non-parametric test that does not assume any distribution and compares the entire cumulative distribution functions (i.e, in this context, the overall shapes of the distributions of values) of two groups. Here we provide an example using the t-test statistic:

```r
pathway_activity <- summarize_pathway_level(assay(airway, "counts"), kegg_sets, type = "ttest", minsize
```

```
##
## 359 functional sets read.

## iteration 100

## iteration 200

## iteration 300

## 339 successful functional aggregations over minsize

## 20 failed functional aggregations under minsize

## Functional activity score matrix has dimensions: 339 , 8
```

```r
print(paste("Pathway activity score matrix has dimensions:", nrow(pathway_activity), ",", ncol(pathway_a
## [1] "Pathway activity score matrix has dimensions: 339 , 8"
```

In this case, `summarize_pathway_level` has generated a pathway-level activity score for each of the 8 samples, for 339 pathways containing more than 15 genes. The resulting test statistic for each sample represents the difference between the two groups (i.e., the functional set and rest of molecules) for each gene or molecule. These statistics can be then be used as features for further analysis or modeling:

```r
print(head(pathway_activity))
```

```
##                                        SRR1039508 SRR1039509 SRR1039512
## 2-Oxocarboxylic acid metabolism          2.454453   2.436486   2.165767
## ABC transporters                         1.603352   1.763264   1.596487
## Acute myeloid leukemia                   3.528012   3.738857   2.841968
## Adherens junction                        3.845327   3.563145   3.786380
## Adipocytokine signaling pathway          4.055114   4.060762   3.889088
## Adrenergic signaling in cardiomyocytes   3.472753   2.788910   3.403428
##                                        SRR1039513 SRR1039516 SRR1039517
## 2-Oxocarboxylic acid metabolism          2.224862   2.271840   2.322606
## ABC transporters                         1.724146   1.624509   1.672874
## Acute myeloid leukemia                   3.153732   3.064161   3.160045
## Adherens junction                        3.249354   4.157697   3.029834
## Adipocytokine signaling pathway          4.110839   3.802921   3.984753
## Adrenergic signaling in cardiomyocytes   3.010759   3.355814   2.804207
##                                        SRR1039520 SRR1039521
## 2-Oxocarboxylic acid metabolism          2.285466   2.280632
## ABC transporters                         1.696842   1.782976
## Acute myeloid leukemia                   2.978309   3.599640
## Adherens junction                        3.864234   3.603268
## Adipocytokine signaling pathway          3.960969   4.073604
## Adrenergic signaling in cardiomyocytes   3.481214   3.179876
```

Importantly, this is just an illustrative example. In real-world experiments, you may have to consider that genes or molecules in a certain set might be longer or naturally more abundant and therefore might have higher overall read counts (abundance) compared to genes or molecules outside the set, even if their expression levels (as a proportion of transcripts) are similar. Normalizing the expression counts for all genes (including those within and outside the sets) using a suitable method (e.g., library size normalization, transcript length normalization) can help make the expression levels more comparable before performing the aggregation and set comparisons.

Molecular sets beyond KEGG and omics matrices beyond `SummarizedExperiment`

The package `funOmics` interoperates with KEGGREST to retrieve molecular sets from the KEGG through the function `get_kegg_sets` (see description and example above). Other real-world molecular sets can be downloaded from several sources. In terms of gene sets, the Gene Ontology is a versatile resource that covers three domains: cellular components, biological processes and molecular functions. Reactome pathways can also be used to generate higher-level functional representations from omics data. Explore the different releases and download the corresponding gene sets for the different types of GO terms, and reactome pathways here. You can also aggregate genes into protein complexes, which you can find in the CORUM database.

Regarding other omics types, such as metabolomics, the function `summarize_pathway_level` can be applied in a similar manner to a metabolomics matrix `X` and KEGG metabolic pathways. Metabolite sets from KEGG pathways can also be downloaded with the KEGG API.

After obtaining the molecular sets information, this data has to be formatted as a list of lists (similar to what is obtained from the `get_kegg_sets` function). In other words, you need a structure where you have a list of multiple molecular sets names, and each of these sets is represented as a list of molecule identifiers, such as entrez IDs, PubChem CIDs, Uniprot IDs, etc. For instance, let's retrieve gene sets from GO terms for cellular compartments. The information can be downloaded from the msigdb link or accessed programmatically as follows:

```
goccdb <- "https://data.broadinstitute.org/gsea-msigdb/msigdb/release/7.5/c5.go.cc.v7.5.entrez.gmt"
downdb <- sapply(readLines(goccdb), function(x) strsplit(x, "\t")[[1]])
gocc_sets <- sapply(as.matrix(downdb), function(x) x[3:length(x)])
names(gocc_sets) = sapply(as.matrix(downdb), function(x) x[1])
gocc_sets[1:3]
```

```
## $GOCC_NUCLEOTIDE_EXCISION_REPAIR_COMPLEX
##  [1] "1069" "1161" "2067" "2071" "2072" "2073" "5424" "5887" "7507" "7508"
## [11] "7515"
##
## $GOCC_HISTONE_DEACETYLASE_COMPLEX
##  [1] "10013"  "10014"  "10284"  "10428"  "10467"  "10524"  "10629"  "10847"
##  [9] "10856"  "10902"  "10933"  "1107"   "1108"   "116092" "1457"   "221037"
## [17] "23186"  "23309"  "23314"  "23468"  "25855"  "25942"  "26038"  "283248"
## [25] "3065"   "3066"   "3094"   "3622"   "473"    "51317"  "51564"  "51780"
## [33] "53615"  "54556"  "54815"  "55758"  "55806"  "55809"  "55818"  "55869"
## [41] "55929"  "57459"  "57504"  "57634"  "57649"  "58516"  "5928"   "5931"
## [49] "64426"  "64431"  "6907"   "7764"   "79595"  "79685"  "79718"  "79885"
## [57] "81611"  "8204"   "8295"   "83933"  "84215"  "84312"  "8607"   "8819"
## [65] "8841"   "90665"  "9112"   "91748"  "9219"   "9611"   "9734"   "9759"
##
## $GOCC_SAGA_COMPLEX
##  [1] "100130302" "10474"     "10629"     "112869"    "117143"    "170067"
##  [7] "23326"     "2648"      "27097"     "55578"     "56943"     "56970"
## [13] "6871"      "6878"      "6880"      "6881"      "6883"      "8295"
## [19] "8464"      "8850"      "9913"
```

As you can see, the resulting `gocc_sets` object is a list of lists where each element represents a GO cellular compartment gene set. Here you can also use the function `short_sets_detail` to get information about which and how many pathways contain less than a given number of molecules:

```
min <- 8
short_sets <- short_sets_detail(gocc_sets, min)
print(head(short_sets$short_sets_molecules))
```

```
## $GOCC_TRANSCRIPTION_FACTOR_TFIIIC_COMPLEX
## [1] "112495" "2975"   "2976"   "9328"   "9329"   "9330"
##
## $GOCC_COMMITMENT_COMPLEX
## [1] "11338" "55015" "6631"  "6632"  "6634"
##
## $GOCC_THO_COMPLEX
## [1] "57187" "79228" "80145" "84321" "8563"  "9984"
##
## $GOCC_EKC_KEOPS_COMPLEX
## [1] "112858" "51002"  "55644"  "8270"   "84520"
##
## $GOCC_GLYCOSYLPHOSPHATIDYLINOSITOL_N_ACETYLGLUCOSAMINYLTRANSFERASE_GPI_GNT_COMPLEX
## [1] "51227" "5277"  "5279"  "5283"  "84992" "8818"  "9091"
##
## $GOCC_HRD1P_UBIQUITIN_LIGASE_ERAD_L_COMPLEX
## [1] "51009" "550"   "57414" "6400"  "79139" "84447" "91319"
```

Notably, omics data does not always come from a `SummarizedExperiment` object. Some times, it is imported from a csv file, generated through other pre-processing steps and packages, or even generated from a simulation. In these cases, the data has to be formatted as a matrix of dimensions `g*s` (`g` molecules and `s` samples).

Let's see an example usage of the GO cellular compartments gene sets `gocc_sets` where omics data is of type matrix. For this purpose, we will create an expression matrix where the expression values are random positive values sampled from a standard normal distribution. Please, note that `funOmics` can be used to aggregate other types of omics data and molecular sets, such as metabolomics or proteomics that may have a similar range of values.

Let's simulate a gene expression matrix `X_expr`, where gene IDs are codes between 1:10000 (to match entrez IDs), and `summarize_pathway_level` can be applied:

```
# Example usage:
set.seed(1)
g <- 10000
s <- 20
X_expr <- matrix(abs(rnorm(g * s)), nrow = g, dimnames = list(1:g, paste0("s", 1:s)))
print(paste("Dimensions of omics matrix X:", dim(X_expr)[1], "*", dim(X_expr)[2]))
```

```
## [1] "Dimensions of omics matrix X: 10000 * 20"
```

```
head(X_expr)
```

```
##          s1        s2        s3        s4        s5        s6        s7
## 1 0.6264538 0.8043316 0.2353485 0.6179223 0.2212571 0.5258908 0.3413341
## 2 0.1836433 1.0565257 0.2448250 0.8935057 0.3517935 0.4875444 0.4136665
## 3 0.8356286 1.0353958 0.6421869 0.4277562 0.1606019 1.1382508 0.1220357
## 4 1.5952808 1.1855604 1.9348085 0.2999012 0.1240523 1.2151344 1.5893806
## 5 0.3295078 0.5004395 1.0386957 0.5319833 0.6598739 0.4248307 0.7874385
## 6 0.8204684 0.5249887 0.2835501 1.7059816 0.5038493 1.4508403 1.5920640
##           s8         s9        s10       s11          s12       s13       s14
## 1 1.00203611 1.55915937 0.09504307 0.7914415 0.0145724495 0.8663644 0.1604426
## 2 0.02590761 0.20166217 0.38805939 0.3921679 1.7854043337 0.9476952 0.9241849
## 3 0.44814178 1.04017610 2.13657003 0.4726670 0.0002997544 0.4522428 1.5561751
## 4 0.84323332 0.07195772 0.55661945 0.4579517 0.4356948690 0.2782408 0.8812202
## 5 0.21846310 0.01526544 0.59094164 0.1681319 1.4076452475 1.4175945 0.5263595
## 6 0.47678629 0.33938598 1.52014345 0.5856737 0.6929698698 0.6329981 0.4627372
##           s15       s16       s17       s18        s19        s20
## 1 0.249371112 1.2520152 2.3150930 0.4414410 0.82485558 1.37086468
## 2 0.335346796 0.3351313 1.0603800 0.4130862 0.74402087 0.54569610
## 3 0.004405287 0.1080678 0.3970672 0.8660777 0.69009734 1.62446330
## 4 0.986768348 0.4717051 0.4840034 2.2615708 1.76900681 0.06247283
## 5 0.543575705 2.5070607 1.3584146 0.1787018 0.55215640 0.57021255
## 6 0.626142823 1.2451344 0.6370574 0.4713582 0.03257056 0.31350574
```

Now, let's summarize the expression data using standard deviation pooling for the GO cellular compartments gene sets. We won't specify a minimum size of sets in this case, so the default `minsize` of 10 is used:

```
sd_gocc_expr <- summarize_pathway_level(X_expr, gocc_sets, type="sd", minsize=8)
```

```
##
## 1006 functional sets read.

## iteration 100

## iteration 200

## iteration 300
## iteration 400

## iteration 500

## iteration 600

## iteration 700

## iteration 800

## iteration 900

## iteration 1000

## 550 successful functional aggregations over minsize

## 456 failed functional aggregations under minsize

## Functional activity score matrix has dimensions: 550 , 20
```

```r
head(sd_gocc_expr)
```

```
##                                                   s1        s2        s3        s4
## GOCC_NUCLEOTIDE_EXCISION_REPAIR_COMPLEX 0.4045096 0.3841496 0.4876056 0.6158703
## GOCC_HISTONE_DEACETYLASE_COMPLEX        0.5961716 0.5524931 0.4077048 0.6363483
## GOCC_SAGA_COMPLEX                       0.6190007 0.4613048 0.4179023 0.5202579
## GOCC_GOLGI_MEMBRANE                     0.6050940 0.5866127 0.6168553 0.6264271
## GOCC_UBIQUITIN_LIGASE_COMPLEX           0.7146701 0.7208348 0.6495247 0.6066239
## GOCC_NUCLEAR_UBIQUITIN_LIGASE_COMPLEX   0.5324259 0.6635115 0.6381171 0.6187512
##                                                   s5        s6        s7        s8
## GOCC_NUCLEOTIDE_EXCISION_REPAIR_COMPLEX 0.5427526 0.8412107 0.6130915 0.5778133
## GOCC_HISTONE_DEACETYLASE_COMPLEX        0.7490914 0.4977517 0.5737141 0.6572306
## GOCC_SAGA_COMPLEX                       0.4176742 0.5104733 0.6437678 0.6215122
## GOCC_GOLGI_MEMBRANE                     0.5953969 0.6037806 0.6283209 0.6215994
## GOCC_UBIQUITIN_LIGASE_COMPLEX           0.6213030 0.6611160 0.6128709 0.5919404
## GOCC_NUCLEAR_UBIQUITIN_LIGASE_COMPLEX   0.6536113 0.6025263 0.6968490 0.4874398
##                                                   s9       s10       s11       s12
## GOCC_NUCLEOTIDE_EXCISION_REPAIR_COMPLEX 0.8376548 0.6066229 0.6922118 0.9214760
## GOCC_HISTONE_DEACETYLASE_COMPLEX        0.6117108 0.7045937 0.3173755 0.6027702
## GOCC_SAGA_COMPLEX                       0.6731533 0.7353114 0.2952001 0.4611380
## GOCC_GOLGI_MEMBRANE                     0.5759732 0.6395941 0.6302452 0.6040828
## GOCC_UBIQUITIN_LIGASE_COMPLEX           0.6589884 0.5835952 0.6676812 0.5805518
## GOCC_NUCLEAR_UBIQUITIN_LIGASE_COMPLEX   0.6787391 0.3562284 0.6064547 0.4911328
##                                                  s13       s14       s15       s16
## GOCC_NUCLEOTIDE_EXCISION_REPAIR_COMPLEX 0.4879332 0.6815041 0.3723823 0.4737714
## GOCC_HISTONE_DEACETYLASE_COMPLEX        0.5398999 0.5982277 0.4842830 0.7086422
## GOCC_SAGA_COMPLEX                       0.6219926 0.4270019 0.9318551 0.4333853
## GOCC_GOLGI_MEMBRANE                     0.5775356 0.5659483 0.5885292 0.6093459
## GOCC_UBIQUITIN_LIGASE_COMPLEX           0.4800778 0.5803810 0.5722157 0.6796406
## GOCC_NUCLEAR_UBIQUITIN_LIGASE_COMPLEX   0.5056348 0.5565119 0.4373176 0.6570645
##                                                  s17       s18       s19       s20
## GOCC_NUCLEOTIDE_EXCISION_REPAIR_COMPLEX 0.8067935 0.3302759 0.5167987 0.7764892
## GOCC_HISTONE_DEACETYLASE_COMPLEX        0.6092391 0.5663914 0.5499079 0.4634814
## GOCC_SAGA_COMPLEX                       0.5610913 0.6366751 0.4932885 0.6815148
## GOCC_GOLGI_MEMBRANE                     0.5888242 0.6110807 0.5752147 0.6149827
## GOCC_UBIQUITIN_LIGASE_COMPLEX           0.5656959 0.6670322 0.5867161 0.6975617
## GOCC_NUCLEAR_UBIQUITIN_LIGASE_COMPLEX   0.5784575 0.4706088 0.5487221 0.5346914
```

GO cellular compartments level expression signatures have been generated via standard deviation aggregation. You can apply similar procedures for other types of molecular sets, aggregation functions and omics types.

The package `funOmics` is conceived to be flexible across omics types and types of molecular sets, so you can also tailor or directly create your own list of molecular sets based on specific criteria of your experiments (e.g., include only protein complexes involved in ubiquitination, or define *ad hoc* metabolic routes involving specific metabolites).

Packages & Session information

The `funOmics` package was developed for R version >= 4.0.3. However, BioConductor release 3.19 runs on R-4.4. See session information and loaded packages below:

```
sI <- sessionInfo()
print(sI, locale = FALSE)
```

```
## R version 4.4.0 (2024-04-24)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.4.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] MultiAssayExperiment_1.30.1 airway_1.24.0
##  [3] SummarizedExperiment_1.34.0 GenomicRanges_1.56.0
##  [5] GenomeInfoDb_1.40.0         IRanges_2.38.0
##  [7] S4Vectors_0.42.0            MatrixGenerics_1.16.0
##  [9] matrixStats_1.3.0           funOmics_0.99.7
## [11] Biobase_2.64.0              BiocGenerics_0.50.0
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1       gridBase_0.4-7         viridisLite_0.4.2
##  [4] dplyr_1.1.4            blob_1.2.4             viridis_0.6.5
##  [7] Biostrings_2.72.0      fastmap_1.1.1          pathifier_1.42.0
## [10] digest_0.6.35          lifecycle_1.0.4        cluster_2.1.6
## [13] KEGGREST_1.44.0        RSQLite_2.3.6          magrittr_2.0.3
## [16] compiler_4.4.0         rlang_1.1.3            rngtools_1.5.2
## [19] tools_4.4.0            utf8_1.2.4             yaml_2.3.8
## [22] NMF_0.30.4.900         knitr_1.46             S4Arrays_1.4.0
## [25] curl_5.2.1             bit_4.0.5              DelayedArray_0.30.0
## [28] plyr_1.8.9             RColorBrewer_1.1-3     abind_1.4-5
## [31] registry_0.5-1         withr_3.0.0            R.oo_1.26.0
## [34] grid_4.4.0             fansi_1.0.6            princurve_2.1.6
## [37] xtable_1.8-4           colorspace_2.1-0       ggplot2_3.5.1
## [40] scales_1.3.0           iterators_1.0.14       cli_3.6.2
## [43] rmarkdown_2.26         crayon_1.5.2           generics_0.1.3
## [46] rstudioapi_0.16.0      httr_1.4.7             reshape2_1.4.4
## [49] BiocBaseUtils_1.6.0    DBI_1.2.2              cachem_1.0.8
## [52] stringr_1.5.1          zlibbioc_1.50.0        assertthat_0.2.1
## [55] parallel_4.4.0         AnnotationDbi_1.66.0   XVector_0.44.0
## [58] vctrs_0.6.5            Matrix_1.7-0           jsonlite_1.8.8
## [61] bit64_4.0.5            dendextend_1.17.1      foreach_1.5.2
## [64] glue_1.7.0              codetools_0.2-20        stringi_1.8.4
## [67] gtable_0.3.5            UCSC.utils_1.0.0        munsell_0.5.1
## [70] tibble_3.2.1            pillar_1.9.0            pkgmaker_0.32.10
## [73] htmltools_0.5.8.1       GenomeInfoDbData_1.2.12 R6_2.5.1
## [76] doParallel_1.0.17       lattice_0.22-6          evaluate_0.23
## [79] R.methodsS3_1.8.2       png_0.1-8               memoise_2.0.1
## [82] Rcpp_1.0.12             SparseArray_1.4.1       gridExtra_2.3
## [85] org.Hs.eg.db_3.19.1     xfun_0.43               pkgconfig_2.0.3
```

Contact Information

Feedback is very welcome! If you have any questions, issues, or suggestions for improving the `funOmics` package, please use the GitHub issues page or contact elisa.gomezdelope@uni.lu.

License

The `funOmics` package is released under the terms of the MIT License. See the LICENSE file for more details.