

# Package: TiDEomics (via r-universe)

October 12, 2026

**Title** Time-course Differential Expression analysis of omics data

**Version** 0.99.5

**Date** 2026-07-31

**Description** TiDEomics provides a comprehensive workflow for multi-group time-course omics data analysis, analysing time-dominant, group-dominant, and group-specific temporal effects through pairwise differential expression, variance decomposition, and co-expression module analysis (WGCNA). The package integrates quality control, data processing, functional enrichment, and extensive visualisation. It supports datasets with missing values (e.g., mass spectrometry-based proteomics), and operates on SummarizedExperiment objects to ensure compatibility with the Bioconductor ecosystem.

**License** GPL (>= 2)

**URL** <https://github.com/hte123/TiDEomics>,  
<https://hte123.github.io/TiDEomics>

**BugReports** <https://github.com/hte123/TiDEomics/issues>

**biocViews** Software, GeneExpression, DifferentialExpression, Proteomics, Transcriptomics, TimeCourse, MultipleComparison, Pathways, Visualization, MassSpectrometry, QualityControl

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**Imports** circlize, clusterProfiler, ComplexHeatmap, dplyr, enrichplot, ggforce, ggh4x, ggplot2, ggplotify, ggpubr, ggrepel, ggridges, ggsci, limma, lme4, methods, patchwork, pbapply, PCAtools, randtests, scales, SummarizedExperiment, tibble, tidyr, Trendy, umap, WGCNA

**Depends** R (>= 4.5.0)

**LazyData** false

**Suggests** knitr, rmarkdown, BiocStyle, testthat (>= 3.1.0), plotly, enrichR, org.Hs.eg.db, org.Mm.eg.db, msigdb, DeeDeeExperiment

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.0.0

**Config/pak/sysreqs** libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev perl python3 zlib1g-dev

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

**Date/Publication** 2026-07-31 18:22:25 UTC

**RemoteUrl** <https://github.com/bioc/TiDEomics>

**RemoteRef** HEAD

**RemoteSha** ca26312c86214fd3ec8d103e330224611463edd3

## Contents

|                                   |    |
|-----------------------------------|----|
| calc_feature_property . . . . .   | 3  |
| calc_mean_sd . . . . .            | 4  |
| create_input . . . . .            | 5  |
| DE_between_group . . . . .        | 6  |
| DE_between_time . . . . .         | 8  |
| decomp_variance . . . . .         | 9  |
| enrich_msigdb . . . . .           | 10 |
| enrichGO_list . . . . .           | 12 |
| enrichGO_rank . . . . .           | 14 |
| enrichR_list . . . . .            | 15 |
| example_go . . . . .              | 16 |
| example_net . . . . .             | 17 |
| example_obj . . . . .             | 17 |
| example_res_list . . . . .        | 18 |
| extract_hubs . . . . .            | 18 |
| extract_segment_trends . . . . .  | 19 |
| flatten_DE . . . . .              | 20 |
| flatten_enrich . . . . .          | 20 |
| get_custom_palette . . . . .      | 21 |
| group_specific_features . . . . . | 22 |
| impute_groups . . . . .           | 23 |
| merge_groups . . . . .            | 24 |
| merge_replicates . . . . .        | 25 |
| normalise_to_start . . . . .      | 26 |
| plot_breakpoints . . . . .        | 26 |
| plot_cor_matrix . . . . .         | 27 |
| plot_cv . . . . .                 | 28 |
| plot_DE_between_group . . . . .   | 29 |
| plot_DE_between_time . . . . .    | 30 |
| plot_distribution . . . . .       | 31 |
| plot_GO . . . . .                 | 32 |
| plot_ID . . . . .                 | 33 |

|                                      |           |
|--------------------------------------|-----------|
| plot_missing . . . . .               | 34        |
| plot_modules_h . . . . .             | 35        |
| plot_modules_v . . . . .             | 37        |
| plot_pca . . . . .                   | 39        |
| plot_pca_3D . . . . .                | 40        |
| plot_pca_arrows . . . . .            | 41        |
| plot_pca_by_group . . . . .          | 42        |
| plot_segments . . . . .              | 43        |
| plot_trend . . . . .                 | 44        |
| plot_umap . . . . .                  | 45        |
| plot_umap_by_group . . . . .         | 46        |
| plot_variance . . . . .              | 47        |
| plot_volcano . . . . .               | 48        |
| plot_WGCNA . . . . .                 | 50        |
| prepare_tide . . . . .               | 51        |
| prepare_WGCNA . . . . .              | 53        |
| run_Trendy . . . . .                 | 55        |
| run_WGCNA . . . . .                  | 56        |
| set_custom_palette . . . . .         | 57        |
| split_groups . . . . .               | 58        |
| summarise_feature_property . . . . . | 58        |
| summarise_module_metrics . . . . .   | 59        |
| summarise_module_pattern . . . . .   | 60        |
| summarise_Trendy . . . . .           | 61        |
| theme_custom . . . . .               | 61        |
| tutorial_data . . . . .              | 62        |
| tutorial_sample_info . . . . .       | 63        |
| WGCNA_module . . . . .               | 64        |
| <b>Index</b>                         | <b>65</b> |

---

calc\_feature\_property *Calculate feature property*

---

## Description

Calculate per-feature temporal properties for each group from the mean of replicates at each time point.

## Usage

```
calc_feature_property(se_obj_merged_list, threshold = NULL)
```

**Arguments**

- `se_obj_merged_list` A list of SummarizedExperiment objects created by `merge_replicates()`, each containing the mean of replicates for each feature at each time point for one group.
- `threshold` A numeric value to determine whether a feature is "expressed" in samples. (default is NULL, meaning any non-NA value is considered expressed).

**Value**

A list of SummarizedExperiment objects, each corresponding to one group, with updated rowData containing the following columns: Feature, Group, Exp\_threshold, T\_exp (number of time points with values > Exp\_threshold), T\_total (total number of time points), P\_trend (p-value from Bartels' test for randomness against a trend), Max\_FC (maximum fold change across time points), Max\_FC\_time (difference between the time points at which maximum and minimum expression occur; positive if max occurs after min, negative otherwise), Exp\_ratio (T\_exp / T\_total), Rho\_time (Spearman correlation with time; positive = up trend), Peak\_ratio (number of local maxima / T\_total; 0 = no peaks detected), Log2\_CV (log2(1 + CV) of expression across time points), AUC (area under the time-0-normalised curve from assay 2; NA if assay 2 is not available).

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)

example_obj_merged_list <-
  calc_feature_property(example_obj_merged_list, threshold = 0)
```

---

|              |                              |
|--------------|------------------------------|
| calc_mean_sd | <i>Calculate mean and SD</i> |
|--------------|------------------------------|

---

**Description**

Calculate mean and standard deviation for each group and time point

**Usage**

```
calc_mean_sd(se_obj)
```

**Arguments**

- `se_obj` A SummarizedExperiment object with assays containing expression data. The first assay should be the original data, and the second (if present) should be normalized data.

Value

A list containing two data frames: one for original values and one for normalized values (if available). Each data frame includes columns for Mean, SD, Group, Time, and Feature.

Examples

```
data(example_obj)
table_mean_sd_list <- calc_mean_sd(example_obj)
table_mean_sd <- table_mean_sd_list$norm
```

---

|              |               |
|--------------|---------------|
| create_input | Create object |
|--------------|---------------|

---

Description

Check format of input data and sample annotation, create a SummarizedExperiment object

Usage

```
create_input(
  data,
  sample_ann,
  subject_col = NULL,
  replicate_col = NULL,
  batch_col = NULL
)
```

Arguments

|               |                                                                                                                                                                                                                                                                                                                                        |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data          | A data frame with rows as features (e.g., genes, proteins) and columns as samples. The first column should contain feature identifiers (e.g., gene symbols) and named as 'Feature'. The data should have been log transformed (and normalised if needed). Missing values are allowed in some of the downstream analyses.               |
| sample_ann    | A data frame containing sample annotations with required columns: 'Sample', 'Group', and 'Time'. 'Replicate' and 'Batch' are optional columns and will be auto-generated if not provided. 'Time', 'Replicate' and 'Batch' should be numeric.                                                                                           |
| subject_col   | Optional: name of a column in sample_ann identifying biological subjects measured repeatedly across time points (e.g., "PatientID"). If provided, the column is renamed to 'Subject' and used for repeated-measures analyses downstream. If NULL (default), all samples are treated as independent, e.g. for cell culture experiments. |
| replicate_col | Optional: name of a column in sample_ann identifying replicate IDs. If provided, the column is renamed to 'Replicate'. If NULL (default), the function looks for a column named 'Replicate'; if absent, replicate IDs are auto-generated within each Group and Time (and Subject, if provided).                                        |

**batch\_col** Optional: name of a column in `sample_ann` identifying batch information. If provided, the column is renamed to 'Batch'. If NULL (default), the function looks for a column named 'Batch'; if absent, all samples are assigned to batch 1.

### Value

A SummarizedExperiment object containing the input data and sample annotations.

### Examples

```
data <- data.frame(Feature = c("Gene1", "Gene2"),
                  Sample1 = c(1, 2), Sample2 = c(3, 4))
sample_ann <- data.frame(
  Sample = c("Sample1", "Sample2"),
  Group = c("A", "A"),
  Time = c(0, 1),
  Rep = c(1, 1),
  BatchID = c(1, 1)
)
se_obj <- create_input(data, sample_ann,
  replicate_col = "Rep", batch_col = "BatchID")
```

---

|                  |                          |
|------------------|--------------------------|
| DE_between_group | <i>DE between groups</i> |
|------------------|--------------------------|

---

### Description

Differential expression analysis between groups at each time point by limma. The function compares each pair of groups at each time point, and returns a nested list of DE analysis results for each pair of groups and each time point including all features, as well as a list of filtered DE results for each pair of groups based on the specified thresholds including only significant features. Use `plot_DE_between_group()` to visualise the number of DE features between groups over time.

### Usage

```
DE_between_group(
  se_obj,
  group = NULL,
  filter = NULL,
  assay,
  adjP_thres = 0.05,
  logFC_thres = 1,
  trend = FALSE
)
```

**Arguments**

|                          |                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>se_obj</code>      | A SummarizedExperiment object                                                                                                                                                                                                                                                                                                                                                                    |
| <code>group</code>       | A character vector specifying which group to be compared to. If NULL, all groups in the 'Group' column will be compared to. (default is NULL)                                                                                                                                                                                                                                                    |
| <code>filter</code>      | Minimum number of replicates required in both conditions for a feature to be tested. If NULL, the minimum number of replicates across all groups and time points will be used. (default is NULL)                                                                                                                                                                                                 |
| <code>assay</code>       | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Required, no default.) The selected assay should contain log-transformed, normalised values (e.g. log2-CPM for RNA-seq, log2-intensity for proteomics). A warning is issued if the data appears to be un-logged raw counts. |
| <code>adjP_thres</code>  | Threshold for adjusted p-value to consider a feature as differentially expressed (default is 0.05)                                                                                                                                                                                                                                                                                               |
| <code>logFC_thres</code> | Threshold for log2 fold change to consider a feature as differentially expressed (default is 1)                                                                                                                                                                                                                                                                                                  |
| <code>trend</code>       | Logical, passed to <code>limma::eBayes()</code> . Set to TRUE for RNA-seq count-derived data to model the mean-variance trend. Leave as FALSE (default) for microarray, proteomics, metabolomics, or other log-intensity data where the mean-variance relationship is typically flat.                                                                                                            |

**Details**

Only time points present in both groups are compared. No multiple-testing correction is applied across the pairwise group-by-time comparisons; each comparison is independent. Apply your own correction (e.g., `stats::p.adjust()`) across combined results if needed.

**Value**

A list with: `all_list` (nested list of DE results per group pair and time point, all features); `de_list` (significant features only); `fit_list` (nested list of limma MArrayLM fit objects, for use with `limma::plotSA()`); `ref_groups` (group to be compared to); `all_groups` (all groups). The output can be passed to `plot_DE_between_group()` for visualisation.

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)

DE_between_group_out <- DE_between_group(example_obj, assay = 2)
```

---

|                 |                               |
|-----------------|-------------------------------|
| DE_between_time | <i>DE between time points</i> |
|-----------------|-------------------------------|

---

## Description

Differential expression analysis between time points within each group by limma

## Usage

```
DE_between_time(
  se_obj,
  group = NULL,
  filter = NULL,
  assay,
  adjP_thres = 0.05,
  logFC_thres = 1,
  trend = FALSE
)
```

## Arguments

|             |                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj      | A SummarizedExperiment object created by create_input                                                                                                                                                                                                                                                                                                                                            |
| group       | A character vector specifying which groups to analyse. If NULL, all groups in the 'Group' column will be used. (default is NULL)                                                                                                                                                                                                                                                                 |
| filter      | Minimum number of replicates required in both conditions for a feature to be tested. If NULL, the minimum number of replicates across all groups and time points will be used. (default is NULL)                                                                                                                                                                                                 |
| assay       | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Required, no default.) The selected assay should contain log-transformed, normalised values (e.g. log2-CPM for RNA-seq, log2-intensity for proteomics). A warning is issued if the data appears to be un-logged raw counts. |
| adjP_thres  | Threshold for adjusted p-value to consider a feature as differentially expressed (default is 0.05)                                                                                                                                                                                                                                                                                               |
| logFC_thres | Threshold for log2 fold change to consider a feature as differentially expressed (default is 1)                                                                                                                                                                                                                                                                                                  |
| trend       | Logical, passed to limma::eBayes(). Set to TRUE for RNA-seq count-derived data to model the mean-variance trend. Leave as FALSE (default) for microarray, proteomics, metabolomics, or other log-intensity data where the mean-variance relationship is typically flat.                                                                                                                          |

## Value

A list with: all\_list (nested list of DE results per group and time comparison, all features); de\_list (nested list of significant features only); fit\_list (nested list of limma MArrayLM fit objects, for use with limma::plotSA()); time\_series (vector of all available time points). The output can be passed to plot\_DE\_between\_time() for visualisation.

## Examples

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)

DE_between_time_out <- DE_between_time(example_obj, assay = 1)
```

---

|                 |                               |
|-----------------|-------------------------------|
| decomp_variance | <i>Variance decomposition</i> |
|-----------------|-------------------------------|

---

## Description

Variance decomposition by linear mixed models (LMM), to estimate the contribution of Group and Time to the variance of each feature. Modified from `PALMO::lmeVariance()` function.

Group and Time are always included as random effects. The formula is extended automatically based on `colData`:

- Subject column present: adds `(1|Subject)` to model between-subject differences
- `interaction = TRUE`: adds `(1|Group:Time)` (or `(1|Subject:Time)` when Subject present) to capture interaction variance. Requires  $\geq 2$  replicates per combination. Default: `FALSE`.

Output always includes: Group, Time, Residual. Subject is included when present. All values are percentages of total variance.

Allow using original data or data normalised to time point 0.

## Usage

```
decomp_variance(
  se_obj,
  features = NULL,
  fixed_effect_var = NULL,
  interaction = FALSE,
  assay,
  core = 1
)
```

## Arguments

|                               |                                                                                                                                                                                                                                                                            |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>se_obj</code>           | A SummarizedExperiment object created by <code>create_input()</code>                                                                                                                                                                                                       |
| <code>features</code>         | A vector of features to include. If <code>NULL</code> , all features.                                                                                                                                                                                                      |
| <code>fixed_effect_var</code> | Column names to include as fixed effects (regressed out, not appearing as variance components). Default is <code>NULL</code> (no fixed effects). Set to <code>"Batch"</code> to regress out batch effects, or provide other columns (e.g., <code>c("Sex", "Age")</code> ). |
| <code>interaction</code>      | Logical. If <code>TRUE</code> , adds <code>(1 Group:Time)</code> (or <code>(1 Subject:Time)</code> when Subject present) to the model to capture interaction variance. Default: <code>FALSE</code> .                                                                       |

|       |                                                                                                                                                                                             |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| assay | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Required, no default.) |
| core  | Number of cores for parallel processing (default: 1)                                                                                                                                        |

**Value**

A data frame with variance decomposition results (percentages).

**References**

<https://github.com/aifimmunology/PALMO/blob/main/R/lmeVariance.R>

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)

var_decomp <- decomp_variance(example_obj, assay = 1)
```

---

enrich\_msigdb

*Gene set enrichment via MSigDB*


---

**Description**

Enrichment analysis against MSigDB gene sets via hypergeometric test. Supports MSigDB categories, subcategories, and specified gene sets. Requires the msigdbR package.

Use cases:

- Drug perturbations: category = "C2", subcategory = "CGP" (chemical and genetic perturbations)
- TF targets: category = "C3", subcategory = "TFT:GTRD"
- Hallmarks: category = "H", subcategory = NULL (no subcategory)
- GO: category = "C5", subcategory = "BP" (GO biological process)

Multiple testing: P-values are adjusted (Benjamini-Hochberg) per module across the gene-set tests within that module.

**Usage**

```
enrich_msigdb(
  gene_list,
  universe,
  category = NULL,
  subcategory = NULL,
  gene_sets = NULL,
  species = "Homo sapiens",
```

```

    db_species = "HS",
    name = NULL,
    pvalueCutoff = 0.05,
    pAdjustMethod = "BH",
    minGSSize = 10,
    maxGSSize = 500
  )

```

## Arguments

|               |                                                                                                                                                                                                                                                                                                                                       |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| gene_list     | A named list of gene vectors, or a data.frame with Feature and Module columns from WGCNA_module().                                                                                                                                                                                                                                    |
| universe      | Background genes (required).                                                                                                                                                                                                                                                                                                          |
| category      | MSigDB category (default: NULL). See msigdb::msigdb_collections(db_species = "HS"/"MM") for available options. Not required when gene_sets is provided; in that case all collections are searched to locate the named gene sets.                                                                                                      |
| subcategory   | MSigDB subcategory (default: NULL). Set to NULL to include all subcategories of category.                                                                                                                                                                                                                                             |
| gene_sets     | Optional character vector of specific MSigDB gene set names to include (e.g., c("HALLMARK_APOPTOSIS", "HALLMARK_P53_PATHWAY")). When provided, msigdb is queried across all collections to find these gene sets, so category and subcategory are not required. If NULL (default), all gene sets in the category/subcategory are used. |
| species       | Species of input genes (default: "Homo sapiens"). See msigdb::msigdb_species() for available options.                                                                                                                                                                                                                                 |
| db_species    | Species of MSigDB genesets (default: "HS"). Use "MM" for mouse-native gene sets. If species and db_species mismatch, db_species MSigDB will be used with ortholog mapping to species.                                                                                                                                                 |
| name          | Name for the output list element. If NULL, auto-derived from category (e.g., "C2" names the output element "C2"). If category is also NULL, it uses "msigdb".                                                                                                                                                                         |
| pvalueCutoff  | Adjusted p-value cutoff (default: 0.05).                                                                                                                                                                                                                                                                                              |
| pAdjustMethod | P-value adjustment method (default: "BH").                                                                                                                                                                                                                                                                                            |
| minGSSize     | Minimum gene set size (default: 10).                                                                                                                                                                                                                                                                                                  |
| maxGSSize     | Maximum gene set size (default: 500).                                                                                                                                                                                                                                                                                                 |

## Value

A named list with one element: a data.frame with columns Cluster, ID, Description, GeneRatio, BgRatio, pvalue, p.adjust, Count, geneID.

## Examples

```

if (requireNamespace("msigdb", quietly = TRUE)) {
  data(example_net)
  example_module <- WGCNA_module(example_net) |>

```

```

    dplyr::filter(Module %in% c("1", "2"))
hallmark_msigdb <- enrich_msigdb(example_module, category = "MH",
  species = "Mus musculus", db_species = "MM",
  minGSSize = 1,
  pvalueCutoff = 0.9,
  universe = WGCNA_module(example_net, exclude_grey = FALSE)$Feature)
}

```

---

enrichGO\_list

*GO enrichment with gene sets*


---

## Description

Gene ontology enrichment analysis of multiple gene sets with clusterProfiler, allowing for using different or same background genes for each gene set

## Usage

```

enrichGO_list(
  gene_list,
  keyType = "SYMBOL",
  OrgDb,
  universe = NULL,
  universe_list = NULL,
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05,
  qvalueCutoff = 0.05,
  category = NULL,
  simplify = FALSE,
  simplify_cutoff = 0.7,
  simplify_by = "p.adjust",
  simplify_select_fun = min,
  simplify_measure = "Wang",
  ...
)

```

## Arguments

|               |                                                                                                    |
|---------------|----------------------------------------------------------------------------------------------------|
| gene_list     | A named list of gene vectors, or a data.frame with Feature and Module columns from WGCNA_module(). |
| keyType       | Available options are AnnotationDbi::keytypes(OrgDb) (default is "SYMBOL")                         |
| OrgDb         | Organism database, e.g. org.Hs.eg.db, org.Mm.eg.db                                                 |
| universe      | Background genes for all input gene sets, used if universe_list is not provided                    |
| universe_list | Background genes for each input gene set, a list of gene vectors with the same names as gene_list  |

|                     |                                                                                                                                  |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------|
| pAdjustMethod       | Parameter of clusterProfiler::enrichGO() (default is "BH")                                                                       |
| pvalueCutoff        | Parameter of clusterProfiler::enrichGO() (default is 0.05)                                                                       |
| qvalueCutoff        | Parameter of clusterProfiler::enrichGO() (default is 0.05)                                                                       |
| category            | GO category to analyze (default is all three of BP, MF, CC)                                                                      |
| simplify            | Whether to simplify the GO terms by removing redundant terms with clusterProfiler::simplify() function. (default is FALSE)       |
| simplify_cutoff     | Parameter of clusterProfiler::simplify(), cutoff for similarity when simplifying GO terms (default is 0.7)                       |
| simplify_by         | Parameter of clusterProfiler::simplify(), method to choose representative term when simplifying GO terms (default is "p.adjust") |
| simplify_select_fun | Parameter of clusterProfiler::simplify(), function to select representative term when simplifying GO terms (default is min)      |
| simplify_measure    | Parameter of clusterProfiler::simplify(), method to calculate similarity when simplifying GO terms (default is "Wang")           |
| ...                 | additional arguments passed to clusterProfiler::enrichGO()                                                                       |

## Value

A nested list of GO enrichment results with sublists: 'all' and 'simplified' (if simplify = TRUE) containing merged results of all or simplified terms across gene sets for each GO category; 'unmerged\_all' and 'unmerged\_simplified' (if simplify = TRUE) including all or simplified terms for each gene set and GO category.

## Examples

```
if (requireNamespace("org.Mm.eg.db", quietly = TRUE)) {
  library(org.Mm.eg.db)
  library(clusterProfiler)
  data(example_net)
  # select two modules for demonstration
  example_module <- WGCNA_module(example_net) |>
    dplyr::filter(Module %in% c("1", "2"))
  # set cutoff to 1 to show all results for demonstration
  example_go_list = enrichGO_list(example_module, OrgDb = org.Mm.eg.db,
    universe = WGCNA_module(example_net, exclude_grey = FALSE)$Feature,
    pvalueCutoff = 1, qvalueCutoff = 1,
    category = "BP", simplify = FALSE)
}
```

enrichGO\_rank

*GO enrichment with ranked gene list***Description**

Gene ontology enrichment analysis of a ranked gene list with clusterProfiler. Genes can be ranked by variance decomposition results.

**Usage**

```
enrichGO_rank(
  rank_table,
  gene_rank_by,
  OrgDb,
  keyType = "SYMBOL",
  go_rank_by = "p.adjust",
  category = NULL,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  ...
)
```

**Arguments**

|               |                                                                                                                                                                                                                      |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rank_table    | A data frame with at least two columns: 'Feature' for gene names and one or more columns of variables for ranking the genes, e.g. output of <code>decomp_variance()</code> containing variance decomposition results |
| gene_rank_by  | Variable in <code>rank_table</code> to rank the genes by, e.g. "Time", "Group" in the output of <code>decomp_variance()</code>                                                                                       |
| OrgDb         | Organism database, e.g. <code>org.Hs.eg.db</code> , <code>org.Mm.eg.db</code>                                                                                                                                        |
| keyType       | Available options are <code>AnnotationDbi::keytypes(OrgDb)</code> (default is "SYMBOL")                                                                                                                              |
| go_rank_by    | Variable in the GO enrichment result to rank the GO terms by (default is "p.adjust", other options include "pvalue", "qvalue", "NES", "setSize", "enrichmentScore", etc.)                                            |
| category      | GO category to analyze (default is all three of BP, MF, CC)                                                                                                                                                          |
| pvalueCutoff  | Parameter of <code>clusterProfiler::gseGO()</code> (default is 0.05)                                                                                                                                                 |
| pAdjustMethod | Parameter of <code>clusterProfiler::gseGO()</code> (default is "BH")                                                                                                                                                 |
| ...           | additional arguments passed to <code>clusterProfiler::gseGO()</code>                                                                                                                                                 |

**Value**

A list of `gseaResult` objects containing the GSEA results

## Examples

```
if (requireNamespace("org.Mm.eg.db", quietly = TRUE)) {
  library(org.Mm.eg.db)
  data(example_obj)
  example_obj <- normalise_to_start(example_obj)

  var_decomp <- decomp_variance(example_obj,
    features = rownames(example_obj)[1:100], assay = 1)
  example_go_rank <- enrichGO_rank(var_decomp, gene_rank_by = "Time",
    OrgDb = org.Mm.eg.db, keyType = "SYMBOL",
    category = "BP")
}
```

---

enrichR\_list

*Gene-set enrichment via enrichR*


---

## Description

Enrichment analysis of multiple gene sets against databases from enrichR (e.g. DSigDB for drug signatures, ChEA for TF targets, KEGG for pathways, DrugBank for drug targets).

## Usage

```
enrichR_list(
  gene_list,
  databases = c("DSigDB", "DrugMatrix"),
  site = "Enrichr",
  universe = NULL,
  universe_list = NULL,
  pvalueCutoff = 0.05,
  include_overlap = FALSE
)
```

## Arguments

|               |                                                                                                                                                                                                                                     |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| gene_list     | A named list of gene vectors, or a data.frame with Feature and Module columns from WGCNA_module(). Gene identifiers must match the convention of the chosen site (e.g. human gene symbols for Enrichr, fly symbols for FlyEnrichr). |
| databases     | Character vector of enrichR database names to query. Available databases can be checked with <code>enrichR::listEnrichrDbs()</code> . (default: <code>c("DSigDB", "DrugMatrix")</code> )                                            |
| site          | enrichR site to query. See <code>enrichR::listEnrichrSites()</code> . (default: "Enrichr")                                                                                                                                          |
| universe      | Background genes for all input gene sets, used if <code>universe_list</code> is not provided.                                                                                                                                       |
| universe_list | Background genes for each input gene set, a list of gene vectors with the same names as <code>gene_list</code> .                                                                                                                    |
| pvalueCutoff  | Adjusted p-value cutoff for filtering enriched terms (default: 0.05)                                                                                                                                                                |

include\_overlap

Parameter passed to `enrichR::enrichr()`. If TRUE, databases are downloaded during each query to output 'Overlap' when analysing with a background. (default: FALSE)

## Details

**Multiple testing:** P-values are adjusted (Benjamini–Hochberg) independently for each module within each database. No correction is applied across databases or across modules.

Requires an internet connection.

## Value

A named list of data.frames, one per database. Each data.frame has columns Cluster, Description, p.adjust (Adjusted.P.value from enrichR output), Combined.Score, Genes, and any additional columns returned by the enrichR API. Compatible with `plot_modules_h(enrich_list = result, enrich_category = "DSigDB")`.

## Examples

```
data(example_net)
library(dplyr)
example_module <- WGCNA_module(example_net) |>
  dplyr::filter(Module %in% c("1", "2"))
# Use high pvalueCutoff for demonstration
# enrichr_out <- enrichR_list(example_module,
#   databases = c("KEGG_2019_Mouse"),
#   universe = WGCNA_module(example_net, exclude_grey = FALSE)$Feature,
#   pvalueCutoff = 0.5)
```

---

example\_go

enrichGO\_list() *output object for runnable examples*

---

## Description

Code for producing the data is in `inst/script/generate_example_go.R`

## Usage

```
data(example_go)
```

## Format

A nested list with GO enrichment results for modules "1" and "2" from 'example\_net', containing categories "BP" and "CC", with `simplify = TRUE` applied:

**all** Merged results across gene sets for each GO category

**simplified** Simplified merged results

**unmerged\_all** Per-gene-set results for each category

**unmerged\_simplified** Simplified per-gene-set results

**Source**

GSE263759

---

example\_netrun\_WGCNA() *output object for runnable examples*

---

**Description**

Code for producing the data is in inst/script/generate\_example\_net.R

**Usage**

```
data(example_net)
```

**Format**

A list including WGCNA module assignments, module eigengenes, dendrogram, input data, sample information, and parameters used

**Source**

GSE263759

---

example\_obj*SummarizedExperiment object for runnable examples*

---

**Description**

A subset of `data_obj <- create_input(data = tutorial_data, sample_ann = tutorial_sample_info)` for use in runnable examples in function documentation.

**Usage**

```
data(example_obj)
```

**Format**

A SummarizedExperiment object with assays of 100 rows and 40 columns, colData of 40 rows and 5 columns:

**colData** tutorial\_sample\_info

**assays** tutorial\_data first 100 rows, "Feature" column as rownames

**Details**

Code for preparing the data is available in inst/script/tutorial\_input.R

**Source**

GSE263759

---

|                  |                                                |
|------------------|------------------------------------------------|
| example_res_list | run_Trendy output object for runnable examples |
|------------------|------------------------------------------------|

---

**Description**

Code for preparing the data is in inst/script/generate\_example\_res\_list.R

**Usage**

```
data(example_res_list)
```

**Format**

A nested list, each element is a list with Trendy results for one group

**Source**

GSE263759

---

|              |                                         |
|--------------|-----------------------------------------|
| extract_hubs | Extract hub features from WGCNA modules |
|--------------|-----------------------------------------|

---

**Description**

Identifies hub features within each module using module membership (kME). For unsigned networks, absolute kME values are used to reproduce the behaviour of `WGCNA::blockwiseModules()`. For signed and signed hybrid networks, signed kME values are retained.

**Usage**

```
extract_hubs(net, top_n = 3, exclude_grey = TRUE)
```

**Arguments**

|              |                                                                      |
|--------------|----------------------------------------------------------------------|
| net          | An output object from <code>run_WGCNA()</code>                       |
| top_n        | Number of hub features to return per module. Default: 3.             |
| exclude_grey | Logical; if TRUE, exclude the grey/unassigned module. Default: TRUE. |

**Value**

A character vector containing the top `top_n` features from each module concatenated together.

**Examples**

```
data(example_net)
hubs <- extract_hubs(example_net, top_n = 3)
```

---

extract\_segment\_trends

*Extract feature trends*

---

**Description**

Extract features with different segment trends from trendy segmented regression results summary

**Usage**

```
extract_segment_trends(trendy.summary)
```

**Arguments**

trendy.summary A data frame, output of summarise\_Trendy(), containing the segmented regression results for all features in all groups.

**Value**

A nested list of features grouped by their segment trend patterns within each group.

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)
example_obj_merged_list <- calc_feature_property(example_obj_merged_list,
  threshold = 0)
# no missing value in the example dataset, so imputation is not necessary
example_obj_merged_imp_list <- impute_groups(example_obj_merged_list)

data("example_res_list")
trendy_summary <- summarise_Trendy(example_res_list)
trendy_list <- extract_segment_trends(trendy_summary)
```

---

|            |                                                       |
|------------|-------------------------------------------------------|
| flatten_DE | <i>Flatten nested differential expression results</i> |
|------------|-------------------------------------------------------|

---

### Description

Flattens the nested DE result structure returned by `DE_between_group()` or `DE_between_time()` into a named list of data frames, one entry per contrast-time combination. This is useful for exporting results to other tools such as DeeDeeExperiment.

### Usage

```
flatten_DE(de_list)
```

### Arguments

`de_list` A list of DE results. Accepts individual output of `DE_between_group()` or `DE_between_time()` with an `all_list` element; or a named list (e.g. `list(between_group = ..., between_time = ...)`), each containing an `all_list` element.

### Value

A named list of data frames with columns renamed for DeeDeeExperiment compatibility (`log2FoldChange`, `pvalue`, `padj`). Returns an empty list if `de_list` is empty.

### Examples

```
data(tutorial_data)
data(tutorial_sample_info)
tide <- prepare_tide(tutorial_data, tutorial_sample_info,
  keep = "threshold", residual_threshold = 100)
tide$DE <- DE_between_group(tide$se, assay = "norm",
  filter = 1, trend = TRUE)
de_flat <- flatten_DE(tide$DE)
str(de_flat, max.level = 1)
```

---

|                |                                          |
|----------------|------------------------------------------|
| flatten_enrich | <i>Flatten nested enrichment results</i> |
|----------------|------------------------------------------|

---

### Description

Flattens the nested enrichment result structure returned by `enrichGO_list()`, `enrichR_list()`, or `enrich_msigdb()` into a flat named list of data frames or `enrichResult` objects. This is useful for exporting results to other tools such as DeeDeeExperiment.

### Usage

```
flatten_enrich(enrich_list)
```

**Arguments**

enrich\_list      Output of `enrichGO_list()`, `enrichR_list()`, or `enrich_msigdb()`; or a named list combining several such outputs (e.g. `list(GO = go_res, MSigDB = msigdb_res)`).

**Value**

A flat named list of data frames or `enrichResult` objects with `DeeDeeExperiment`-compatible names and columns (e.g. `"clusterProfiler_BP_1"`, `"enrichr_DSigDB"`). Returns an empty list if `enrich_list` is empty.

**Examples**

```
data(example_go)
enrich_flat <- flatten_enrich(example_go)
str(enrich_flat, max.level = 1)
```

---

|                    |                                 |
|--------------------|---------------------------------|
| get_custom_palette | <i>Get custom color palette</i> |
|--------------------|---------------------------------|

---

**Description**

Get the custom color palette set by `set_custom_palette`. If no custom palette has been set, the function will return a default color palette based on the number of groups specified in the `groups` argument.

**Usage**

```
get_custom_palette(groups)
```

**Arguments**

groups            A character vector of group names for which to retrieve the colors from the custom palette.

**Value**

A named vector of colors corresponding to the specified group names.

**Examples**

```
get_custom_palette(c("untreated", "IFNbeta"))
```

---

group\_specific\_features

*Group specific features*


---

## Description

Identify features that are unique to selected groups (present in those groups but not in any others) and annotate them with gene names using the `clusterProfiler::bitr()` function. The function also provides an option to perform Gene Ontology (GO) enrichment analysis on the identified unique features using the `enrichGO_list()` function and visualize the results with a dot plot.

## Usage

```
group_specific_features(
  property_random_fc,
  groups = NULL,
  filter_ratio = 0.5,
  group_pct = 1,
  genename = TRUE,
  GO = TRUE,
  OrgDb = NULL,
  keytype = NULL,
  ...
)
```

## Arguments

|                    |                                                                                                                                                                                                                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| property_random_fc | A data frame containing the results of the <code>calc_feature_property()</code> function, which includes the feature names, group names, and the proportion of expressed values for each feature in each group. The data frame should have at least the following columns: "Feature", "Group", "Exp_ratio" and "Exp_threshold". |
| groups             | A character vector of group names to be compared. If NULL, all groups in the input data will be used (default is NULL).                                                                                                                                                                                                         |
| filter_ratio       | A numeric value between 0 and 1 specifying the minimum proportion of expressed time points (minimum Exp_ratio) for a feature to be considered present (default is 0.5, meaning that a feature must have $\geq 50\%$ values $>$ threshold in a group ( $\geq 0.5$ Exp_ratio) to be considered present in that group).            |
| group_pct          | A numeric value between 0 and 1 specifying the percentage of groups in which a feature must be present. (default is 1, meaning that a feature must be present in all specified groups).                                                                                                                                         |
| genename           | A logical value indicating whether to output a table of gene names. If TRUE, the function will use the <code>clusterProfiler::bitr()</code> function to annotate the features with gene names based on the specified OrgDb and keytype. (default is TRUE).                                                                      |

|         |                                                                                                                                                                                                                                                                                                                                                                       |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GO      | A logical value indicating whether to perform Gene Ontology (GO) enrichment analysis on the identified unique features. If TRUE, the function will use the <code>enrichGO_list()</code> function to perform GO enrichment analysis and visualize the results with a dot plot. (default is TRUE).                                                                      |
| OrgDb   | An OrgDb object from the AnnotationDbi package corresponding to the organism of interest (e.g., <code>org.Hs.eg.db</code> for human, <code>org.Mm.eg.db</code> for mouse). This will be used for gene annotation with the <code>clusterProfiler::bitr()</code> function.                                                                                              |
| keytype | A character string specifying the type of gene identifiers used in the row names of the assay data (e.g., "SYMBOL", "ENTREZID", "ENSEMBL"). This will be used for gene annotation with the <code>clusterProfiler::bitr()</code> function. Available key types depend on the OrgDb database and can be checked with the <code>AnnotationDbi::keytypes</code> function. |
| ...     | Additional arguments to be passed to the <code>enrichGO_list()</code> function for GO enrichment analysis (e.g., <code>pvalueCutoff</code> , <code>qvalueCutoff</code> , etc.).                                                                                                                                                                                       |

### Value

A named list with element features (character vector of features identified as unique to the specified groups based on the filtering criteria). If `genename` is TRUE, element `genename` contains a data.frame of gene annotations. If GO is TRUE and enrichment succeeds, element GO contains a dot plot of GO enrichment results. Returns NULL if no unique features pass the filter.

### Examples

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)

example_obj_merged_list <-
  calc_feature_property(example_obj_merged_list, threshold = 0)
property_random_fc <- summarise_feature_property(example_obj_merged_list)

group_specific_features(property_random_fc, groups = c("untreated"),
  genename = FALSE, GO = FALSE)
```

---

impute\_groups

---

*Impute missing values*


---

### Description

Impute missing values for each group of samples in a list of SummarizedExperiment objects. By default, replaces NA with the minimum non-NA value per group / subject. A custom function can be supplied for other strategies.

The input samples can contain replicates, or merged replicates (mean of replicates).

**Usage**

```
impute_groups(se_obj_list, fun = min, impute_by = c("group", "subject"))
```

**Arguments**

|             |                                                                                                                                                                                                                             |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj_list | A list of SummarizedExperiment objects, created by split_groups() or merge_replicates(), each corresponds to one group of samples.                                                                                          |
| fun         | Function applied to the non-NA values to generate the replacement value. Default: min. Use function(x) min(x) / 2 for half-minimum, median for median imputation, etc.                                                      |
| impute_by   | "group" (default): compute replacement value from all non-NA values in the group. "subject": compute per-subject replacement value (requires Subject column). If a subject is all NA, fall back to group-level replacement. |

**Value**

A list of SummarizedExperiment objects with missing values imputed. Each object corresponds to one group of samples.

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)
example_obj_merged_imp_list <- impute_groups(example_obj_merged_list)
```

---

merge\_groups

---

*Merge groups into one object*


---

**Description**

Merge SummarizedExperiment object of different groups into one

**Usage**

```
merge_groups(se_obj_list)
```

**Arguments**

|             |                                                                                                                                                    |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj_list | A list of SummarizedExperiment objects, such as output of split_groups() and merge_replicates(). Names of the list components are the group names. |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

A SummarizedExperiment object containing all samples from the input list. The 'Group' column in the colData will indicate the group of each sample. New sample names will be prefixed with the group name.

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)
example_obj_merged <- merge_groups(example_obj_merged_list)
```

---

|                  |                         |
|------------------|-------------------------|
| merge_replicates | <i>Merge replicates</i> |
|------------------|-------------------------|

---

**Description**

Calculate mean of replicates for each feature at each time point for each group.

When a Subject column is present, merging is done in two stages:

1. Average replicates within each Group-Subject-Time combination.
2. Average across subjects within each Group-Time combination. This gives equal weight to each subject regardless of replicate count.

Without a Subject column, all samples at the same Group-Time are averaged together in a single step.

**Usage**

```
merge_replicates(se_obj_list)
```

**Arguments**

`se_obj_list` A list of SummarizedExperiment objects created by `split_groups()`, each corresponds to one group of samples.

**Value**

A list of SummarizedExperiment objects containing the mean of replicates for each feature at each time point for each group. Each object in the list corresponds to one group of samples.

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)
```

---

|                    |                                         |
|--------------------|-----------------------------------------|
| normalise_to_start | <i>Normalise to starting time point</i> |
|--------------------|-----------------------------------------|

---

### Description

Normalise to starting time point, to make the mean of starting time point samples 0.

Two modes are available:

- `by_subject = FALSE` (default): computes the group-level mean at the first non-NA time point per feature and subtracts it. All samples within a group share the same baseline.
- `by_subject = TRUE`: computes each subject's value at the first non-NA time point per feature and subtracts that subject-specific baseline. Removes between-subject baseline differences. Use when each subject has their own initial condition. Requires a Subject column in `colData`.

### Usage

```
normalise_to_start(se_obj, by_subject = FALSE)
```

### Arguments

|                         |                                                                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <code>se_obj</code>     | A SummarizedExperiment object created by <code>create_input()</code>                                                                  |
| <code>by_subject</code> | If FALSE (default), use group-level baseline. If TRUE, use subject-level baseline (requires Subject column in <code>colData</code> ). |

### Value

A SummarizedExperiment object with normalised data in the second assay slot.

### Examples

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
```

---

|                  |                                     |
|------------------|-------------------------------------|
| plot_breakpoints | <i>Plot breakpoint distribution</i> |
|------------------|-------------------------------------|

---

### Description

Plot breakpoint distribution among time points for each group

### Usage

```
plot_breakpoints(res_list, group = NULL, fontsize = 8, ...)
```

**Arguments**

|          |                                                                                                                       |
|----------|-----------------------------------------------------------------------------------------------------------------------|
| res_list | A list of the Trendy analysis results, output of run_Trendy()                                                         |
| group    | A character vector of group names to be plotted. If NULL, all groups in the input list will be used (default is NULL) |
| fontsize | Font size for the plot (default is 8)                                                                                 |
| ...      | Additional arguments to be passed to Trendy::topTrendy()                                                              |

**Value**

A plot showing the distribution of breakpoints over time for each specified group

**Examples**

```
data("example_res_list")
plot_breakpoints(example_res_list)
```

---

|                 |                                |
|-----------------|--------------------------------|
| plot_cor_matrix | <i>Plot correlation matrix</i> |
|-----------------|--------------------------------|

---

**Description**

Plot correlation matrix between samples as a heatmap

**Usage**

```
plot_cor_matrix(
  se_obj,
  assay = 1,
  use = "pairwise.complete.obs",
  method = c("spearman", "pearson", "kendall"),
  label_group = TRUE,
  label_time = TRUE,
  label_rep = TRUE,
  label_batch = TRUE,
  show_rownames = FALSE,
  show_colnames = FALSE,
  fontsize = 8,
  cellwidth = 1,
  cellheight = 1,
  title = NULL,
  ...
)
```

**Arguments**

|               |                                                                                                                                                                                   |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj        | A SummarizedExperiment object created by <code>create_input()</code>                                                                                                              |
| assay         | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Default: 1.) |
| use           | Parameter of <code>stats::cor()</code> (default is "pairwise.complete.obs")                                                                                                       |
| method        | Parameter of <code>stats::cor()</code> (default is "spearman")                                                                                                                    |
| label_group   | Whether to label Group (default is TRUE)                                                                                                                                          |
| label_time    | Whether to label Time (default is TRUE)                                                                                                                                           |
| label_rep     | Whether to label Replicate (default is TRUE)                                                                                                                                      |
| label_batch   | Whether to label Batch (default is TRUE)                                                                                                                                          |
| show_rownames | Whether to show row names in the heatmap (default is FALSE)                                                                                                                       |
| show_colnames | Whether to show column names in the heatmap (default is FALSE)                                                                                                                    |
| fontsize      | Font size for the plot (default is 8)                                                                                                                                             |
| cellwidth     | Cell width for the heatmap (default is 1)                                                                                                                                         |
| cellheight    | Cell height for the heatmap (default is 1)                                                                                                                                        |
| title         | Title of the heatmap. If NULL, auto-generated as "Sample correlation ()" with the method capitalised (e.g. "Sample correlation (Spearman)") (default: NULL)                       |
| ...           | Additional arguments to be passed to <code>ComplexHeatmap::Heatmap()</code>                                                                                                       |

**Value**

A heatmap showing the correlation between samples.

**Examples**

```
data(example_obj)
plot_cor_matrix(example_obj)
```

---

plot\_cv

*Plot coefficient of variation (CV)*


---

**Description**

Plot the distribution of coefficient of variation (CV) for each feature with replicates. The CV is calculated as the standard deviation divided by the mean of the abundance values.

Note: CV is only meaningful for positive-valued data.

**Usage**

```
plot_cv(se_obj, fontsize = 8)
```

**Arguments**

|          |                                                                                                                                      |
|----------|--------------------------------------------------------------------------------------------------------------------------------------|
| se_obj   | A SummarizedExperiment object, produced by create_input() function, containing the abundance data and associated sample information. |
| fontsize | Font size for the plot (default is 8) (default is 8).                                                                                |

**Value**

A plot showing the distribution of coefficient of variation (CV) for each feature, grouped by Time and Group. If there are no replicates, a message will be printed indicating that CV cannot be calculated.

**Examples**

```
data(example_obj)
plot_cv(example_obj)
```

---

```
plot_DE_between_group  DE number between groups
```

---

**Description**

Plot the number of differentially expressed features between groups over time, using the output of DE\_between\_group().

DE features were pre-filtered with DE\_between\_group(), but by specifying adjP\_thres and logFC\_thres, users can re-filter the DE features for plotting.

**Usage**

```
plot_DE_between_group(
  DE_between_group_out,
  group = NULL,
  fontsize = 8,
  adjP_thres = NULL,
  logFC_thres = NULL
)
```

**Arguments**

|                      |                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DE_between_group_out | Output of DE_between_group(), a list with all_list, de_list, ref_groups, all_groups elements.                                                                 |
| group                | A character vector specifying which groups to use as the reference (Cond1) for plotting. If NULL, all groups in the output will be plotted. (default is NULL) |
| fontsize             | Font size for the plot (default is 8)                                                                                                                         |
| adjP_thres           | Threshold for adjusted p-value to consider a feature as differentially expressed, for re-filtering the DE features. (default is NULL, no re-filtering)        |
| logFC_thres          | Threshold for log2 fold change to consider a feature as differentially expressed, for re-filtering the DE features. (default is NULL, no re-filtering)        |

**Value**

A series of line plots showing the number of DE features over time for each group comparison.

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)

DE_between_group_out <- DE_between_group(example_obj, assay = 2)
plot_DE_between_group(DE_between_group_out, fontsize = 8)
```

---

plot\_DE\_between\_time    *DE number between time points*

---

**Description**

Plot the number of differentially expressed features between time points within each group with heatmaps

DE features were pre-filtered with `DE_between_time()`, but by specifying `adjP_thres` and `logFC_thres`, users can re-filter the DE features for plotting.

**Usage**

```
plot_DE_between_time(
  DE_between_time_out,
  value = TRUE,
  fontsize = 8,
  nrow = 1,
  heatmap_width = 4,
  heatmap_unit = "cm",
  adjP_thres = NULL,
  logFC_thres = NULL
)
```

**Arguments**

|                                  |                                                                                                                                          |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>DE_between_time_out</code> | Output of <code>DE_between_time()</code> , a list with <code>all_list</code> , <code>de_list</code> , <code>time_series</code> elements. |
| <code>value</code>               | Whether to display the actual number of DE features in each heatmap cell (default is TRUE)                                               |
| <code>fontsize</code>            | Font size for the plot (default is 8)                                                                                                    |
| <code>nrow</code>                | Number of rows for arranging the heatmaps (default is 1)                                                                                 |
| <code>heatmap_width</code>       | Width of each heatmap (default is 4)                                                                                                     |
| <code>heatmap_unit</code>        | Unit for the heatmap width (default is "cm")                                                                                             |

|             |                                                                                                                                                        |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| adjP_thres  | Threshold for adjusted p-value to consider a feature as differentially expressed, for re-filtering the DE features. (default is NULL, no re-filtering) |
| logFC_thres | Threshold for log2 fold change to consider a feature as differentially expressed, for re-filtering the DE features. (default is NULL, no re-filtering) |

**Value**

One heatmap per group showing the number of DE features between time points for each group, with the same scale across groups for easy comparison. The heatmap cells are optionally annotated with the number of DE features.

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)

DE_between_time_out <- DE_between_time(example_obj, assay = 1)
plot_DE_between_time(DE_between_time_out,
  fontsize = 8, value = TRUE, nrow = 1, heatmap_width = 3)
```

---

|                   |                                    |
|-------------------|------------------------------------|
| plot_distribution | <i>Abundance distribution plot</i> |
|-------------------|------------------------------------|

---

**Description**

This function generates density plots to visualise the distribution of abundance values across samples. The user can choose to facet the plot by "Group", "Time", or "Sample" to compare distributions across different conditions or samples. If no faceting variable is specified, a single density plot will be generated for all samples combined.

**Usage**

```
plot_distribution(se_obj, facet_by = NULL, fontsize = 8)
```

**Arguments**

|          |                                                                                                                                                                                                                                     |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj   | A SummarizedExperiment object, produced by create_input() function, containing the abundance data and associated sample information.                                                                                                |
| facet_by | A character string specifying the variable to facet the plot by. It can be one of "Group", "Time", or "Sample". If NULL, no faceting will be applied and a single density plot will be generated for all samples (default is NULL). |
| fontsize | Font size for the plot (default is 8) (default is 8).                                                                                                                                                                               |

**Value**

A plot showing the density distribution of abundance values, optionally faceted by the specified variable.

**Examples**

```
data(example_obj)
plot_distribution(example_obj)
plot_distribution(example_obj, facet_by = "Group")
```

plot\_GO

*Plot GO enrichment***Description**

Visualization of GO enrichment analysis with dotplot, cnetplot, or emapplot in clusterProfiler

**Usage**

```
plot_GO(
  go_list,
  plot_dotplot = TRUE,
  plot_cnetplot = FALSE,
  plot_emapplot = FALSE,
  showCategory_dotplot = 5,
  showCategory_cnetplot = 5,
  showCategory_emapplot = 5,
  fontsize = 8,
  label = "genes",
  ...
)
```

**Arguments**

|                       |                                                                                                                                                 |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| go_list               | A list of enriched GO results, output from enrichGO_list()                                                                                      |
| plot_dotplot          | Whether to plot dotplot (default is TRUE)                                                                                                       |
| plot_cnetplot         | Whether to plot cnetplot (default is FALSE)                                                                                                     |
| plot_emapplot         | Whether to plot emapplot (default is FALSE)                                                                                                     |
| showCategory_dotplot  | showCategory parameter of clusterProfiler::dotplot() (default is 5)                                                                             |
| showCategory_cnetplot | showCategory parameter of clusterProfiler::cnetplot() (default is 5)                                                                            |
| showCategory_emapplot | showCategory parameter of clusterProfiler::emapplot() (default is 5)                                                                            |
| fontsize              | Font size for the plots (default is 8)                                                                                                          |
| label                 | Title for the plots (default is "genes")                                                                                                        |
| ...                   | Additional parameters to pass to the plotting functions clusterProfiler::dotplot(), clusterProfiler::cnetplot(), or clusterProfiler::emapplot() |

**Value**

A list of plots visualizing the GO enrichment results.

**Examples**

```
data(example_go)
plot_GO(example_go$all, plot_dotplot = TRUE,
        plot_emapplot = FALSE, plot_cnetplot = FALSE)
```

---

|         |                                           |
|---------|-------------------------------------------|
| plot_ID | <i>Plot number of identified features</i> |
|---------|-------------------------------------------|

---

**Description**

Plot the number of identified features (i.e., features with non-missing values) for each sample

**Usage**

```
plot_ID(se_obj, fontsize = 8, signif = FALSE, ...)
```

**Arguments**

|          |                                                                                                                                                    |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj   | A SummarizedExperiment object, produced by create_input() function, containing the abundance data and associated sample information.               |
| fontsize | Font size for the plot (default is 8) (default is 8).                                                                                              |
| signif   | A logical value indicating whether to perform significance testing between groups and add significance annotations to the plot (default is FALSE). |
| ...      | Additional arguments to be passed to ggpubr::stat_compare_means() when signif is TRUE, for customizing the significance annotations.               |

**Value**

A plot showing the ID number for each sample, with a dashed line indicating the average number across all samples. And a boxplot comparing the ID number between groups, with optional significance annotations.

**Examples**

```
# simulate data with random missing values
na_data <- matrix(rnorm(1500), nrow = 100, ncol = 150)
na_data[sample(length(na_data), size = 2000)] <- NA
na_data <- data.frame(Feature = paste0("Feature", 1:100), na_data)
colnames(na_data)[-1] <- paste0("Sample", 1:150)

na_obj <- create_input(na_data,
  data.frame(Sample = paste0("Sample", 1:150),
    Time = rep(rep(1:10, each = 5), 3),
```

```

Group = rep(c("A", "B", "C"), each = 50),
Replicate = rep(1:5, 30)))
plot_ID(na_obj, signif = TRUE)

```

---

|              |                          |
|--------------|--------------------------|
| plot_missing | <i>Plot missing rate</i> |
|--------------|--------------------------|

---

## Description

Plot the ratio of missing values for each sample

## Usage

```
plot_missing(se_obj, fontsize = 8, signif = FALSE, ...)
```

## Arguments

|          |                                                                                                                                                                |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj   | A SummarizedExperiment object, produced by <code>create_input()</code> function, containing the abundance data and associated sample information.              |
| fontsize | Font size for the plot (default is 8) (default is 8).                                                                                                          |
| signif   | A logical value indicating whether to perform significance testing between groups and add significance annotations to the plot (default is FALSE).             |
| ...      | Additional arguments to be passed to <code>ggpubr::stat_compare_means()</code> when <code>signif</code> is TRUE, for customizing the significance annotations. |

## Value

A plot showing the missing value ratio for each sample, with a dashed line indicating the global missing value ratio across all samples. And a boxplot comparing the missing value ratio between groups, with optional significance annotations.

## Examples

```

# simulate data with random missing values
na_data <- matrix(rnorm(1500), nrow = 100, ncol = 150)
na_data[sample(length(na_data), size = 2000)] <- NA
na_data <- data.frame(Feature = paste0("Feature", 1:100), na_data)
colnames(na_data)[-1] <- paste0("Sample", 1:150)

na_obj <- create_input(na_data,
  data.frame(Sample = paste0("Sample", 1:150),
    Time = rep(rep(1:10, each = 5), 3),
    Group = rep(c("A", "B", "C"), each = 50),
    Replicate = rep(1:5, 30)))
plot_missing(na_obj, signif = TRUE)

```

---

|                |                                         |
|----------------|-----------------------------------------|
| plot_modules_h | <i>Plot modules (horizontal layout)</i> |
|----------------|-----------------------------------------|

---

### Description

Plot WGCNA modules' heatmaps, mean expression profiles, and enrichment terms, aligned horizontally.

Different from running WGCNA, the input data should have the replicates merged, instead of having multiple samples per group, time and feature (gene).

If certain time points are missing in some groups, NA values are added.

### Usage

```
plot_modules_h(
  module,
  se_obj_merged,
  scale = TRUE,
  assay,
  ylabel = "Log2 abundance",
  profile_width = 3,
  profile_link_width = 1,
  enrich_list = NULL,
  enrich_category = "BP",
  enrich_rank_by = "p.adjust",
  enrich_top_n = 3,
  enrich_threshold = 0.05,
  fontsize = 8,
  heatmap_width = 8,
  heatmap_height = 8,
  mark_features = NULL,
  width = 16,
  height = 12,
  res = 300,
  suffix = "",
  device = "png",
  save = NULL
)
```

### Arguments

|               |                                                                                                                                                                                                                                                                                           |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| module        | A data frame with columns "Feature" and "Module"                                                                                                                                                                                                                                          |
| se_obj_merged | A SummarizedExperiment object, with one value for each feature at each time point in each group (replicates merged). The colData of the object should contain columns "Sample", "Group", and "Time". The object can be produced by split_groups(), merge_replicates() and merge_groups(). |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| scale              | Whether to scale the data (z-score) across samples for each feature (default is TRUE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| assay              | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Required, no default.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| ylabel             | Y axis label prefix (default is "Log2 abundance")                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| profile_width      | Width of the mean expression profile panels in cm (default: 3)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| profile_link_width | Width of the link between heatmap and mean expression profile panels in cm (default: 1)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| enrich_list        | A named enrichment list for modules, as produced by <code>enrichGO_list()</code> \$all. Each named element should be a data.frame with columns Cluster, Description, and the rank column. Accepts any enrichment result with compatible format. When <code>enrich_category</code> is a vector, each category is shown as a separate column in the annotation textbox. If NULL (default), no enrichment terms will be displayed.                                                                                                                                                                                                                                       |
| enrich_category    | Category name(s) of enrichment to display: a single string (e.g. "BP"), or a vector for multiple categories (e.g. <code>c("BP", "CC", "Hub features")</code> ). When "Hub features" is included, <code>mark_features</code> are shown as a column in the textbox.                                                                                                                                                                                                                                                                                                                                                                                                     |
| enrich_rank_by     | Numeric column used to rank enrichment terms within each module. A single string (default: "p.adjust") is recycled for all categories in <code>enrich_category</code> . Supply a character vector of matching length for per-category control. Also used for colour-coding: terms with values below <code>enrich_threshold</code> are shown in black, terms above in grey.                                                                                                                                                                                                                                                                                            |
| enrich_top_n       | Top N enrichment terms per module. A single integer (default: 3) is recycled for all categories in <code>enrich_category</code> . Supply an integer vector of matching length for per-category control.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| enrich_threshold   | P-value / adjusted p-value threshold for colour-coding enrichment terms. A single value (default: 0.05) is recycled for all categories; supply a vector of matching length for per-category control. Terms below threshold are drawn in black, terms above in grey. Set to NULL to skip colour-coding (all terms black). Set to NA for "Hub features".                                                                                                                                                                                                                                                                                                                |
| fontsize           | Base font size (default: 8)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| heatmap_width      | Width of the heatmap body in cm (default: 8)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| heatmap_height     | Height of the heatmap body in cm (default: 8). Set to NULL to auto-size.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| mark_features      | Features to label. Accepts two forms: <ul style="list-style-type: none"> <li>• A <b>character vector</b>: all marked in black on the left.</li> <li>• A <b>named list</b> of character vectors, e.g. <code>list("Hub 1" = c("gene1"), "Hub 2" = c("gene2"))</code>. Names become categories with auto-assigned colours (via <code>ggsci::pal_jco()</code>). Auto-routed: shown as <code>anno_mark</code> (left side, connecting lines) unless "Hub features" is present in <code>enrich_category</code>, in which case an <code>anno_multicol_textbox</code> (right side) is built instead, using the same colours. Set to NULL (default) to skip marking.</li> </ul> |

|        |                                                                                                                              |
|--------|------------------------------------------------------------------------------------------------------------------------------|
| width  | Width of the saved image (default is 16 (cm))                                                                                |
| height | Height of the saved image (default is 12 (cm))                                                                               |
| res    | Resolution of the saved image (except pdf format) (default: 300 (ppi))                                                       |
| suffix | Suffix for the saved image file name (default: "")                                                                           |
| device | Image file format(s) for saving. Can be a character vector with one or more of "png", "pdf", "tiff", "jpeg" (default: "png") |
| save   | Directory to save the plot, no saving if is NULL (default is NULL)                                                           |

**Value**

A combined plot of heatmap, mean expression profile, and enrichment terms for each WGCNA module

**References**

<https://github.com/junjunlab/ClusterGVis>

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)
example_obj_merged <- merge_groups(example_obj_merged_list)

data(example_net)
# select two modules for demonstration
example_module <- WGCNA_module(example_net) |>
  dplyr::filter(Module %in% c("1", "2"))

data(example_go)

plot_modules_h(example_module |> dplyr::filter(Module != '0'),
  example_obj_merged, assay = 2, scale = TRUE,
  ylabel = "Z-score of log2 expression",
  enrich_list = example_go$all,
  enrich_category = "BP",
  heatmap_width = 6, heatmap_height = 4)
```

---

plot\_modules\_v

---

*Plot modules (vertical layout)*


---

**Description**

Plot WGCNA modules' mean expression profiles and heatmaps, align vertically.

Different from running WGCNA, the input data should have the replicates merged, instead of having multiple samples per group, time and feature (gene).

If certain time points are missing in some groups, NA values are added.

**Usage**

```
plot_modules_v(
  module,
  se_obj_merged,
  scale = TRUE,
  assay,
  ylabel = "Log2 abundance",
  suffix = "",
  device = "png",
  save = NULL,
  width = 12,
  height = 8,
  height_ratio = 2,
  fontsize = 8,
  res = 300
)
```

**Arguments**

|               |                                                                                                                                                                                                                                                                                           |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| module        | A data frame with columns "Feature" and "Module"                                                                                                                                                                                                                                          |
| se_obj_merged | A SummarizedExperiment object, with one value for each feature at each time point in each group (replicates merged). The colData of the object should contain columns "Sample", "Group", and "Time". The object can be produced by split_groups(), merge_replicates() and merge_groups(). |
| scale         | Whether to scale the data (z-score) across samples for each feature (default is TRUE)                                                                                                                                                                                                     |
| assay         | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Required, no default.)                                                                                               |
| ylabel        | Y axis label prefix (default is "Log2 abundance")                                                                                                                                                                                                                                         |
| suffix        | Suffix for the saved image file name (default is an empty string)                                                                                                                                                                                                                         |
| device        | Image file format(s) for saving. Can be a character vector, e.g. c("png", "pdf"), to save in multiple formats (default: "png").                                                                                                                                                           |
| save          | Directory to save the plot, no saving if is NULL (default is NULL)                                                                                                                                                                                                                        |
| width         | Width of the saved image (default is 12 (cm))                                                                                                                                                                                                                                             |
| height        | Height of the saved image (default is 8 (cm))                                                                                                                                                                                                                                             |
| height_ratio  | Ratio of height of heatmap to the line plot (default is 2)                                                                                                                                                                                                                                |
| fontsize      | Font size for the plot (default is 8)                                                                                                                                                                                                                                                     |
| res           | Resolution of the saved image (except pdf format) (default is 300 (ppi))                                                                                                                                                                                                                  |

**Value**

Two plots aligned vertically by groups: the top one is a line plot of module feature mean expression profiles, the bottom one is a heatmap of feature expression across time points.

**Examples**

```

data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)
example_obj_merged <- merge_groups(example_obj_merged_list)

data(example_net)
example_module <- WGCNA_module(example_net)

plot_modules_v(example_module |> dplyr::filter(Module != '0'),
  example_obj_merged, assay = 2, scale = TRUE,
  ylabel = "Z-score of log2 (expression)",
  height_ratio = 2,
  fontsize = 6)

```

plot\_pca

*Plot PCA***Description**

Plot principal component analysis (PCA), using features without missing values

The samples are coloured by Group and sized by Time. Ellipses are drawn for each Group.

**Usage**

```

plot_pca(
  se_obj,
  plot_screepplot = TRUE,
  plot_loadings = TRUE,
  plot_morepc = TRUE,
  morepc = seq(1, 5),
  pc1 = 1,
  pc2 = 2,
  circle = FALSE,
  xlim_min = NULL,
  xlim_max = NULL,
  ylim_min = NULL,
  ylim_max = NULL,
  fontsize = 8,
  assay = 1
)

```

**Arguments**

|                 |                                                                                   |
|-----------------|-----------------------------------------------------------------------------------|
| se_obj          | A SummarizedExperiment object created by create_input()                           |
| plot_screepplot | Logical, whether to plot screepplot with PCAtools::screepplot() (default is TRUE) |

|               |                                                                                                                                                                                   |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| plot_loadings | Logical, whether to plot loadings with <code>PCAtools::plotloadings()</code> (default is TRUE)                                                                                    |
| plot_morepc   | Logical, whether to plot pairs of more PCs with <code>PCAtools::pairsplot()</code> (default is TRUE)                                                                              |
| morepc        | A numeric vector of PCs to plot in the pairs plot (default is 1:5)                                                                                                                |
| pc1           | Numeric, which PC to use for the x-axis (default is 1)                                                                                                                            |
| pc2           | Numeric, which PC to use for the y-axis (default is 2)                                                                                                                            |
| circle        | Logical, whether to draw circles (ellipses) around samples of each group (default is FALSE)                                                                                       |
| xlim_min      | Minimum x-axis limit when drawing ellipses (default: NULL, auto-computed with 0.35x padding around the PC1 range)                                                                 |
| xlim_max      | Maximum x-axis limit when drawing ellipses (default: NULL, auto-computed with 0.35x padding around the PC1 range)                                                                 |
| ylim_min      | Minimum y-axis limit when drawing ellipses (default: NULL, auto-computed with 0.35x padding around the PC2 range)                                                                 |
| ylim_max      | Maximum y-axis limit when drawing ellipses (default: NULL, auto-computed with 0.35x padding around the PC2 range)                                                                 |
| fontsize      | Font size for the PCA plot (default is 8)                                                                                                                                         |
| assay         | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Default: 1.) |

### Value

A PCA plot showing the distribution of samples, other plots provided by `PCAtools` package, and `PCAtools` output object for custom plotting.

### Examples

```
data(example_obj)
PC = plot_pca(example_obj, morepc = seq(1, 3))
```

---

plot\_pca\_3D

*Plot PCA in 3D*

---

### Description

Plot principal component analysis (PCA) results in 3D. This function takes the output PCA object of the `plot_pca` function and visualizes the samples in a 3D space defined by the specified principal components.

The samples are coloured by Group and sized by Time.

### Usage

```
plot_pca_3D(pca, pcs = seq(1, 3))
```

**Arguments**

|     |                                                                                       |
|-----|---------------------------------------------------------------------------------------|
| pca | A PCA object returned by the plot_pca() function (\$pca) or PCAtools::pca().          |
| pcs | A numeric vector specifying which three principal components to plot (default is 1:3) |

**Value**

An interactive 3D PCA plot showing the distribution of samples in the space defined by the specified principal components. The samples are coloured by Group and sized by Time.

**Examples**

```
data(example_obj)
PC = plot_pca(example_obj, morepc = seq(1, 3))
if (requireNamespace("plotly", quietly = TRUE)) {
  plot_pca_3D(PC$pca, pcs = seq(1, 3))
}
```

---

|                 |                             |
|-----------------|-----------------------------|
| plot_pca_arrows | <i>Plot PCA with arrows</i> |
|-----------------|-----------------------------|

---

**Description**

Plot PCA with arrows indicating trajectory over time, for a single group.

**Usage**

```
plot_pca_arrows(
  se_obj,
  circle = TRUE,
  arrow = TRUE,
  pc1 = 1,
  pc2 = 2,
  fontsize = 8,
  assay = 1
)
```

**Arguments**

|        |                                                                                      |
|--------|--------------------------------------------------------------------------------------|
| se_obj | A SummarizedExperiment object (single group).                                        |
| circle | Logical, whether to draw ellipses around samples of each time point (default: TRUE). |
| arrow  | Logical, whether to draw arrows indicating the trajectory over time (default: TRUE). |
| pc1    | Principal component for the x-axis (default: 1).                                     |
| pc2    | Principal component for the y-axis (default: 2).                                     |

|          |                                                                                                                                                                                   |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fontsize | Font size for the plot (default is 8)                                                                                                                                             |
| assay    | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Default: 1.) |

**Value**

A PCA plot coloured by Time, with optional ellipses and trajectory arrows.

**Examples**

```
data(example_obj)
plot_pca_arrows(example_obj[, example_obj$Group == "IFNbeta"])
```

---

|                   |                          |
|-------------------|--------------------------|
| plot_pca_by_group | <i>Plot PCA by group</i> |
|-------------------|--------------------------|

---

**Description**

Plot PCA for each group separately, with optional circles around time points and arrows indicating trajectory over time. Accepts either a SummarizedExperiment object or a list of them (from `split_groups()`).

**Usage**

```
plot_pca_by_group(
  se_obj,
  circle = TRUE,
  arrow = TRUE,
  pc1 = 1,
  pc2 = 2,
  nrow = 1,
  fontsize = 8,
  assay = 1,
  legend_pos = "right"
)
```

**Arguments**

|        |                                                                                                  |
|--------|--------------------------------------------------------------------------------------------------|
| se_obj | A SummarizedExperiment object, or a named list of them (e.g. from <code>split_groups()</code> ). |
| circle | Logical, whether to draw ellipses around samples of each time point (default: TRUE).             |
| arrow  | Logical, whether to draw arrows indicating the trajectory over time (default: TRUE).             |
| pc1    | Principal component for the x-axis (default: 1).                                                 |
| pc2    | Principal component for the y-axis (default: 2).                                                 |

|            |                                                                                                                                                                                   |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nrow       | Number of rows for arranging the plots (default: 1).                                                                                                                              |
| fontsize   | Font size for the plot (default is 8)                                                                                                                                             |
| assay      | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Default: 1.) |
| legend_pos | Legend position (default: "right").                                                                                                                                               |

**Value**

A series of PCA plots showing the distribution of samples in each group, coloured by Time.

**Examples**

```
data(example_obj)
plot_pca_by_group(example_obj, circle = TRUE, arrow = TRUE)

# Also accepts a list from split_groups()
example_obj_list <- split_groups(example_obj)
plot_pca_by_group(example_obj_list)
```

---

|               |                                  |
|---------------|----------------------------------|
| plot_segments | <i>Plot segmented regression</i> |
|---------------|----------------------------------|

---

**Description**

Plot segmented regression results for specified features in each group, using the `Trendy::plotFeature()` function. The input is a list of SummarizedExperiment objects with no missing values and a list of Trendy analysis results for each group.

**Usage**

```
plot_segments(
  se_obj_imp_list,
  res_list,
  feature,
  group = NULL,
  nrow = NULL,
  ...
)
```

**Arguments**

|                 |                                                                            |
|-----------------|----------------------------------------------------------------------------|
| se_obj_imp_list | A list of SummarizedExperiment objects with no missing values              |
| res_list        | A list of the Trendy analysis results, output of <code>run_Trendy()</code> |
| feature         | A character vector of feature names to be plotted                          |

|       |                                                                                                                                                             |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| group | A character vector of group names to be analysed. If NULL, all groups in the input list will be used (default is NULL)                                      |
| nrow  | Number of rows in the plot layout. If NULL and multiple features are provided, it will be set to half the number of features (rounded up) (default is NULL) |
| ...   | Additional arguments to be passed to the Trendy: <code>plotFeature()</code>                                                                                 |

### Value

Plots of the segmented regression fits for the specified features in each group.

### Examples

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)
example_obj_merged_list <-
  calc_feature_property(example_obj_merged_list, threshold = 0)
# no missing value in the example dataset, so imputation is not necessary
example_obj_merged_imp_list <- impute_groups(example_obj_merged_list)

data("example_res_list")
plot_segments(example_obj_merged_imp_list, example_res_list,
  feature = c("Kif3a"))
```

---

plot\_trend

*Plot feature abundance over time*

---

### Description

Plot trend of feature abundances / expression over time by mean and standard deviation (SD) for each group. Accepts either a SummarizedExperiment object (computing mean/SD internally via `calc_mean_sd()`) or a pre-computed table from `calc_mean_sd()`.

### Usage

```
plot_trend(
  se_obj,
  assay = 1,
  groups = NULL,
  features,
  title = NULL,
  ylabel = "Log2 abundance",
  errorbar = TRUE,
  fontsize = 8
)
```

**Arguments**

|          |                                                                                                                                                                                   |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj   | A SummarizedExperiment object, or a data.frame from calc_mean_sd() with columns: Feature, Time, Group, Mean, SD.                                                                  |
| assay    | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Default: 1.) |
| groups   | Groups to be plotted, if NULL, all groups will be used (default is NULL)                                                                                                          |
| features | Features to be plotted, if NULL, an error will be raised                                                                                                                          |
| title    | Title of the plot. "Feature" if NULL (default).                                                                                                                                   |
| ylabel   | Y axis label of the plot (default is "Log2 abundance")                                                                                                                            |
| errorbar | Whether to plot error bars (default is TRUE)                                                                                                                                      |
| fontsize | Font size for the plot (default is 8)                                                                                                                                             |

**Value**

Plot of feature abundances over time by mean and SD

**Examples**

```
data(example_obj)
plot_trend(example_obj,
  features = sample(rownames(example_obj), 4))
```

---

plot\_umap

*Plot UMAP*


---

**Description**

Plot UMAP of samples, using features without missing values

The UMAP plots are labelled by Group, Time, Replicate (if more than 1), Batch (if more than 1), and number of identified features (non-missing values).

**Usage**

```
plot_umap(
  se_obj,
  seed = 1234,
  plot_ID = FALSE,
  circle = FALSE,
  xlim_min = NULL,
  xlim_max = NULL,
  ylim_min = NULL,
  ylim_max = NULL,
  umap_neighbors = NULL,
  fontsize = 8,
  assay = 1
)
```

**Arguments**

|                |                                                                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj         | A SummarizedExperiment object created by create_input()                                                                                                                           |
| seed           | Random seed for UMAP (default is 1234)                                                                                                                                            |
| plot_ID        | Whether to include a UMAP plot coloured by number of identified features (default is FALSE)                                                                                       |
| circle         | Logical, whether to draw circles (ellipses) around samples of each group (default is FALSE)                                                                                       |
| xlim_min       | Minimum x-axis limit when drawing ellipses (default: NULL, auto-computed with 0.35x padding around the UMAP x range)                                                              |
| xlim_max       | Maximum x-axis limit when drawing ellipses (default: NULL, auto-computed with 0.35x padding around the UMAP x range)                                                              |
| ylim_min       | Minimum y-axis limit when drawing ellipses (default: NULL, auto-computed with 0.35x padding around the UMAP y range)                                                              |
| ylim_max       | Maximum y-axis limit when drawing ellipses (default: NULL, auto-computed with 0.35x padding around the UMAP y range)                                                              |
| umap_neighbors | UMAP n_neighbors parameter (default is selected by .umap_n_neighbors() function based on the number of samples)                                                                   |
| fontsize       | Font size for the plot (default is 8)                                                                                                                                             |
| assay          | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Default: 1.) |

**Value**

A list of UMAP plots showing the distribution of samples. And a data frame containing UMAP coordinates and sample annotations for custom plotting.

**Examples**

```
data(example_obj)
umap_layout <- plot_umap(example_obj)
```

---

plot\_umap\_by\_group      *Plot UMAP by group*

---

**Description**

Plot UMAP for each group separately. Accepts either a SummarizedExperiment object or a named list of them (from split\_groups()).

**Usage**

```
plot_umap_by_group(
  se_obj,
  seed = 1234,
  nrow = 1,
  umap_neighbors = NULL,
  fontsize = 8,
  assay = 1,
  legend_pos = "right"
)
```

**Arguments**

|                |                                                                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| se_obj         | A SummarizedExperiment object, or a named list of them (e.g. from <code>split_groups()</code> ).                                                                                  |
| seed           | Random seed for UMAP (default: 1234).                                                                                                                                             |
| nrow           | Number of rows for arranging the plots (default: 1).                                                                                                                              |
| umap_neighbors | UMAP <code>n_neighbors</code> parameter (default: auto-selected based on sample count).                                                                                           |
| fontsize       | Base font size (default: 8).                                                                                                                                                      |
| assay          | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Default: 1.) |
| legend_pos     | Legend position (default: "right").                                                                                                                                               |

**Value**

A series of UMAP plots showing the distribution of samples in each group, coloured by Time.

**Examples**

```
data(example_obj)
plot_umap_by_group(example_obj)

# Also accepts a list from split_groups()
example_obj_list <- split_groups(example_obj)
plot_umap_by_group(example_obj_list)
```

---

|               |                                    |
|---------------|------------------------------------|
| plot_variance | <i>Plot variance decomposition</i> |
|---------------|------------------------------------|

---

**Description**

Based on `PALMO::variancefeaturePlot()` function

Usage

```
plot_variance(  
  var_decomp,  
  rank = "Group",  
  features = NULL,  
  top_n = 20,  
  show_ylab = TRUE,  
  fontsize = 8  
)
```

Arguments

|            |                                                                                            |
|------------|--------------------------------------------------------------------------------------------|
| var_decomp | A data frame output from decomp_variance()                                                 |
| rank       | Rank the plot by selected variable (default is "Group")                                    |
| features   | Features to be plotted (default is NULL)                                                   |
| top_n      | Top n features ranked by rank to be plotted when features is not specified (default is 20) |
| show_ylab  | Whether to show y axis text (default is TRUE)                                              |
| fontsize   | Font size for the plot (default is 8)                                                      |

Value

A stacked bar plot showing the percentage of variance explained by each model component (Group, Time, Residual, plus Subject and interaction terms when present). Features are ranked by the specified rank variable (Group or Time) and only the top n features are plotted if features is not specified.

Examples

```
data(example_obj)  
example_obj <- normalise_to_start(example_obj)  
  
var_decomp <- decomp_variance(example_obj, assay = 1)  
plot_variance(var_decomp, rank = "Time", top_n = 20)
```

---

|              |                                   |
|--------------|-----------------------------------|
| plot_volcano | <i>Volcano plot of DE results</i> |
|--------------|-----------------------------------|

---

Description

This function creates a volcano plot to visualize the results of differential expression (DE) analysis performed by DE\_between\_group() or DE\_between\_time(). The plot displays log2 fold change on the x-axis and -log10 adjusted p-value on the y-axis, with DE features highlighted based on specified thresholds. If no thresholds are provided, the function will use the thresholds that were applied when running DE\_between\_group() or DE\_between\_time().

**Usage**

```
plot_volcano(
  DE_out,
  group1,
  group2,
  time,
  group,
  time1,
  time2,
  logFC_thres = NULL,
  adjP_thres = NULL,
  label = FALSE,
  fontsize = 8,
  ...
)
```

**Arguments**

|             |                                                                                                                                                 |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| DE_out      | Output from DE_between_group() or DE_between_time()                                                                                             |
| group1      | (Required for DE_between_group() output) Name of the first group for comparison                                                                 |
| group2      | (Required for DE_between_group() output) Name of the second group for comparison                                                                |
| time        | (Required for DE_between_group() output) Time point for comparison                                                                              |
| group       | (Required for DE_between_time() output) Name of the group for comparison                                                                        |
| time1       | (Required for DE_between_time() output) First time point for comparison                                                                         |
| time2       | (Required for DE_between_time() output) Second time point for comparison                                                                        |
| logFC_thres | Log2 fold change threshold for highlighting DE features, default is NULL (using threshold when DE_between_group() or DE_between_time() was run) |
| adjP_thres  | Adjusted p-value threshold for highlighting DE features, default is NULL (using threshold when DE_between_group() or DE_between_time() was run) |
| label       | Whether to label names of DE features on the plot (default is FALSE)                                                                            |
| fontsize    | Font size for the plot (default is 8)                                                                                                           |
| ...         | Additional arguments for ggrepel::geom_text_repel() when label = TRUE                                                                           |

**Value**

A volcano plot of DE results

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)

DE_between_group_out <- DE_between_group(example_obj, assay = 2)
plot_volcano(DE_between_group_out, group1 = "untreated",
```

```
group2 = "IFNbeta", time = 24,
logFC_thres = 0.5, adjP_thres = 0.05, label = TRUE)
```

---

plot\_WGCNA

*Plot WGCNA results*


---

### Description

Plot WGCNA module eigengenes and module-group/time correlations based on output of `run_WGCNA()`

### Usage

```
plot_WGCNA(net, fontsize = 8)
```

### Arguments

|                       |                                                         |
|-----------------------|---------------------------------------------------------|
| <code>net</code>      | WGCNA network object output by <code>run_WGCNA()</code> |
| <code>fontsize</code> | Font size for plots (default is 8)                      |

### Value

Plots of WGCNA module dendrogram, module eigengenes, pairwise scatterplots of eigengenes, clustering of module eigengenes, and module-trait correlation heatmap

### References

[https://github.com/edo98811/WGCNA\\_official\\_documentation/blob/main/FemaleLiver-03-relateModsToExt.R](https://github.com/edo98811/WGCNA_official_documentation/blob/main/FemaleLiver-03-relateModsToExt.R)

### Examples

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)

data("example_net")
plot_WGCNA(example_net, fontsize = 8)
```

---

|              |                                |
|--------------|--------------------------------|
| prepare_tide | <i>Prepare TiDEomics input</i> |
|--------------|--------------------------------|

---

## Description

One-step data preparation wrapper for TiDEomics: creates the SummarizedExperiment object, normalises to starting time point, filters out group-specific features, runs variance decomposition, and filters out noisy features with high residual variance. Returns both unmerged and merged (mean of replicates) outputs, with all-feature and filtered versions of each, ready for downstream analysis.

## Usage

```
prepare_tide(
  data,
  sample_ann,
  subject_col = NULL,
  replicate_col = NULL,
  batch_col = NULL,
  feature_threshold = 0,
  filter_ratio = 0.5,
  min_groups = 2,
  keep = c("below_quantile", "top_n", "threshold"),
  residual_threshold = NULL,
  n_keep = 5000,
  interaction = FALSE,
  n_cores = 1
)
```

## Arguments

|               |                                                                                                                                                                                                                                                                                                                                        |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data          | A data frame with rows as features (e.g., genes, proteins) and columns as samples. The first column should contain feature identifiers (e.g., gene symbols) and be named 'Feature'. The data should have been normalised and log transformed. Missing values are allowed.                                                              |
| sample_ann    | A data frame containing sample annotations with required columns: 'Sample', 'Group', and 'Time'. 'Replicate' and 'Batch' are optional columns and will be auto-generated if not provided. 'Time', 'Replicate' and 'Batch' should be numeric.                                                                                           |
| subject_col   | Optional: name of a column in sample_ann identifying biological subjects measured repeatedly across time points (e.g., "PatientID"). If provided, the column is renamed to 'Subject' and used for repeated-measures analyses downstream. If NULL (default), all samples are treated as independent, e.g. for cell culture experiments. |
| replicate_col | Optional: name of a column in sample_ann identifying replicate IDs. If provided, the column is renamed to 'Replicate'. If NULL (default), the function looks for a column named 'Replicate'; if absent, replicate IDs are auto-generated within each Group and Time (and Subject, if provided).                                        |

|                                 |                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>batch_col</code>          | Optional: name of a column in <code>sample_ann</code> identifying batch information. If provided, the column is renamed to 'Batch'. If NULL (default), the function looks for a column named 'Batch'; if absent, all samples are assigned to batch 1.                                                                                                                                     |
| <code>feature_threshold</code>  | Threshold for <code>calc_feature_property()</code> . Feature values > <code>feature_threshold</code> are considered expressed (default: 0).                                                                                                                                                                                                                                               |
| <code>filter_ratio</code>       | Minimum ratio of time points a feature must be expressed in, to be considered expressed in a group (default: 0.5, meaning that a feature must be > <code>feature_threshold</code> in >=50% time points in a group to be considered expressed). Passed to <code>group_specific_features()</code> .                                                                                         |
| <code>min_groups</code>         | Minimum number of groups a feature must be expressed in, with the above <code>filter_ratio</code> , to pass the cross group filter (default: 2). Passed to <code>group_specific_features()</code> .                                                                                                                                                                                       |
| <code>keep</code>               | How to select features after variance decomposition: "below_quantile" (default): keep features with residual variance below <code>residual_threshold</code> quantile (0-1 scale). "top_n": keep the top <code>n_keep</code> lowest-residual features. "threshold": keep features with residual percentage below <code>residual_threshold</code> (0-100 scale; use 100 to skip filtering). |
| <code>residual_threshold</code> | Numeric: when <code>keep</code> = "below_quantile", the quantile of residual variance to use as cutoff (0-1 scale). When <code>keep</code> = "threshold", the percentage cutoff (0-100 scale, 100 = keep all features). (default: NULL, auto-assigned as 0.75 for "below_quantile", 100 for "threshold")                                                                                  |
| <code>n_keep</code>             | Number of features to keep when <code>keep</code> = "top_n" (default: 5000).                                                                                                                                                                                                                                                                                                              |
| <code>interaction</code>        | Passed to <code>decomp_variance()</code> : if TRUE, adds (1 Group:Time) (or (1 Subject:Time) when Subject present) interaction term (default: FALSE).                                                                                                                                                                                                                                     |
| <code>n_cores</code>            | Number of cores for variance decomposition (default: 1).                                                                                                                                                                                                                                                                                                                                  |

## Details

The pipeline:

1. `create_input()`: validate and build SE
2. `normalise_to_start()`: add time-0-normalised assay
3. `split_groups()` -> `merge_replicates()` -> `calc_feature_property()` -> `summarise_feature_property()`: per-feature expression statistics
4. `group_specific_features()`: filter out group-specific features
5. `decomp_variance()`: LMM variance decomposition on both "orig" and "norm" assays
6. Residual filter: filter out features with high residual variance, calculated from `decomp_variance()` on each assay
7. Produce filtered-unmerged and filtered-merged SEs for each assay

**Value**

A named list with elements. The elements suffixed with "\_filtered" have passed both the cross-group and residual filters, and are split into \$orig and \$norm sub-lists, with variance decomposition and residual filtering done per assay.

1. se (SE with replicates, all features, both assays);
2. se\_filtered (a list of two SEs with replicates, filtered features by orig / norm, use for WGCNA);
3. merged\_list (per-group merged SEs, all features, use for Trendy);
4. merged\_list\_filtered (a list of two per-group merged SEs, filtered features by orig / norm);
5. merged\_se (single merged SE, all groups combined, all features, use for plot\_modules\_v/h);
6. merged\_se\_filtered (a list of two single merged SEs, filtered features by orig / norm, use for plot\_modules\_v/h);
7. variance (a list of two variance decomposition data.frames, computed from orig / norm assay);
8. feature\_property (feature property summary, all features);
9. filter\_summary (per-stage filtering statistics);
10. group\_specific\_filter (a vector of group-specific features, removed by the cross-group filter); 11-13. DE, enrichment, WGCNA (empty on initialization).

**Examples**

```
data(tutorial_data)
data(tutorial_sample_info)
tide <- prepare_tide(tutorial_data, tutorial_sample_info,
  keep = "threshold", residual_threshold = 100)
tide$filter_summary
```

---

```
prepare_WGCNA
```

---



---

*Prepare data and choose power for WGCNA*

---

**Description**

Prepare input data for WGCNA (format, QC), then choose the appropriate soft thresholding power for network construction, by analysing scale free topology with different soft thresholding powers.

**Usage**

```
prepare_WGCNA(
  se_obj,
  assay,
  networkType = "signed",
  RsquaredCut = 0.8,
  MeanConnectivity = 100,
```

```

    powers = NULL,
    fontsize = 8,
    ...
)

```

## Arguments

|                               |                                                                                                                                                                                                                                                                                                      |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>se_obj</code>           | A SummarizedExperiment object. Data normalised to time point 0 can be in the second assay slot, created by <code>normalise_to_start()</code> . Features can be pre-filtered, e.g. by residual variance calculated by <code>decomp_variance()</code> , to remove noisy features before running WGCNA. |
| <code>assay</code>            | The assay to use in the SummarizedExperiment object: a numeric index or character name, e.g. 1 or "orig" for original data, 2 or "norm" for time 0 normalised data. (Required, no default.)                                                                                                          |
| <code>networkType</code>      | Parameter of <code>WGCNA::pickSoftThreshold()</code> (default is "signed")                                                                                                                                                                                                                           |
| <code>RsquaredCut</code>      | Parameter of <code>WGCNA::pickSoftThreshold()</code> (default is 0.8)                                                                                                                                                                                                                                |
| <code>MeanConnectivity</code> | Line of mean connectivity (default is 100)                                                                                                                                                                                                                                                           |
| <code>powers</code>           | Parameter of <code>WGCNA::pickSoftThreshold()</code> (default: NULL, auto-assigned as <code>seq(1, 20)</code> )                                                                                                                                                                                      |
| <code>fontsize</code>         | Font size for the plot (default is 8)                                                                                                                                                                                                                                                                |
| <code>...</code>              | Additional parameters to be passed to <code>WGCNA::pickSoftThreshold()</code>                                                                                                                                                                                                                        |

## Value

A list containing results of the scale-free topology fit indices with different powers, suggested power, network type and prepared input data used for reuse in `run_WGCNA()`

## References

[https://github.com/edo98811/WGCNA\\_official\\_documentation/](https://github.com/edo98811/WGCNA_official_documentation/)

## Examples

```

data(example_obj)
example_obj <- normalise_to_start(example_obj)

wgcna_input <- prepare_WGCNA(example_obj, assay = 2, powers = seq(1, 30),
  networkType = "signed", RsquaredCut = 0.8)
wgcna_input$fitIndices
picked_power <- wgcna_input$powerEstimate

```

run\_Trendy

*Segmented regression analysis***Description**

Run segmented regression analysis with Trendy on imputed data for multiple groups of samples in a list of SummarizedExperiment objects. The results will be returned in a list format. Each element in the list corresponds to one group of samples and contains the Trendy analysis results for that group.

If the feature parameter is not specified, all features expressed in at least minExp time points in each group will be included in the analysis for the group. `calc_feature_property()` should be run before imputation to calculate the expression ratio for each feature in each group. Missing values can be imputed after `calc_feature_property()` using the `impute_groups()` function.

If a group has less than 4 time points available, Trendy analysis cannot be performed and NULL will be returned for the group.

**Usage**

```
run_Trendy(
  se_obj_imp_list,
  group = NULL,
  feature = NULL,
  minExp = 0.5,
  maxK = 1,
  meanCut = 0,
  minNumInSeg = 3,
  NCores = 1,
  ...
)
```

**Arguments**

|                              |                                                                                                                                                                                                                                                                                   |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>se_obj_imp_list</code> | A list of SummarizedExperiment objects with no missing values                                                                                                                                                                                                                     |
| <code>group</code>           | A character vector of group names to be analysed. If NULL, all groups in the input list will be used (default is NULL)                                                                                                                                                            |
| <code>feature</code>         | A character vector of feature names to be included in the analysis. If NULL, all features with expression ratio ( <code>Exp_ratio</code> ) $\geq$ <code>minExp</code> will be used, based on results of <code>calc_feature_property()</code> before imputation. (default is NULL) |
| <code>minExp</code>          | Minimum expression ratio for a feature to be included in the analysis (default is 0.5)                                                                                                                                                                                            |
| <code>maxK</code>            | Parameter of <code>Trendy::trendy()</code> , maximum number of breakpoints allowed in the segmented regression model (default is 1)                                                                                                                                               |
| <code>meanCut</code>         | Parameter of <code>Trendy::trendy()</code> , minimum mean expression required for a feature to be included in the analysis (default is 0)                                                                                                                                         |

|             |                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------|
| minNumInSeg | Parameter of Trendy::trendy(), minimum number of samples required in each segment (default is 3) |
| NCores      | Number of cores to use for parallel processing (default is 1)                                    |
| ...         | Additional arguments to be passed to the Trendy::trendy()                                        |

**Value**

A list of Trendy analysis results, including the fitted model parameters and statistics for each feature in each group.

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)
example_obj_merged_list <- calc_feature_property(example_obj_merged_list,
  threshold = 0)
# no missing value in the example dataset, so imputation is not necessary
example_obj_merged_imp_list <- impute_groups(example_obj_merged_list)

# "untreated" group has only 3 time points, so Trendy analysis will not be
# performed for this group
example_res_list <- run_Trendy(example_obj_merged_imp_list, maxK = 1,
  minNumInSeg = 2, meanCut = 0,
  feature = rownames(example_obj_merged_imp_list[[1]])[1:10])
```

run\_WGCNA

*Weighted gene co-expression network analysis***Description**

Run WGCNA based on output of prepare\_WGCNA(), to identify co-expression modules of features with WGCNA::blockwiseModules()

**Usage**

```
run_WGCNA(wgcna_input, power, numericLabels = TRUE, ...)
```

**Arguments**

|               |                                                                                                                                            |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| wgcna_input   | A list output by prepare_WGCNA(), containing the prepared input data and networkType parameter used in power selection.                    |
| power         | Soft-thresholding power to be used in WGCNA::blockwiseModules(), selected automatically or manually based on the output of prepare_WGCNA() |
| numericLabels | Whether to use numeric labels for modules in the output (default is TRUE)                                                                  |
| ...           | Additional parameters to be passed to WGCNA::blockwiseModules()                                                                            |

**Value**

The built network and parameters of `WGCNA::blockwiseModules()`, and the input data and sample information for use in `plot_WGCNA()`.

**References**

[https://github.com/edo98811/WGCNA\\_official\\_documentation/blob/main/FemaleLiver-03-relateModsToExt.R](https://github.com/edo98811/WGCNA_official_documentation/blob/main/FemaleLiver-03-relateModsToExt.R)

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)

wgcna_input <- prepare_WGCNA(example_obj, assay = 2, powers = seq(1, 30),
  networkType = "signed", RsquaredCut = 0.8)
wgcna_input$fitIndices
picked_power <- wgcna_input$powerEstimate
example_net <- run_WGCNA(wgcna_input,
  power = picked_power,
  minModuleSize = 10, # only 100 genes in the example data
  numericLabels = TRUE)
```

---

|                    |                                 |
|--------------------|---------------------------------|
| set_custom_palette | <i>Set custom color palette</i> |
|--------------------|---------------------------------|

---

**Description**

Set a custom color palette for groups to use in all subsequent plotting functions that support custom palettes.

**Usage**

```
set_custom_palette(palette)
```

**Arguments**

|         |                                                                                                                                                                   |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| palette | A named vector where names correspond to group names and values are the corresponding colors (e.g., <code>c("Group1" = "#FF0000", "Group2" = "#00FF00")</code> ). |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

The function does not return a value but sets the custom palette as an option that can be accessed by other plotting functions. The palette will be stored in the R session options under the name "custom.palette".

**Examples**

```
custom_palette <- c("untreated" = "#1b9e77", "IFNbeta" = "#d95f02",
  "IFNgamma" = "#7570b3", "LPS" = "#e7298a")
set_custom_palette(custom_palette)
```

---

split\_groups

*Split groups*


---

### Description

Split SummarizedExperiment object by groups

### Usage

```
split_groups(se_obj)
```

### Arguments

se\_obj                    A SummarizedExperiment object created by create\_input()

### Value

A list of SummarizedExperiment objects for each group in the input object. Each object in the list corresponds to one group of samples.

### Examples

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
```

---

summarise\_feature\_property

*Summarise feature properties*


---

### Description

This function takes a list of merged SummarizedExperiment objects, which are the output of the calc\_feature\_property() function, and summarizes the feature properties across all groups. It extracts the row data from each SummarizedExperiment object, combines them into a single data frame, and returns this summary data frame. Each row in the resulting data frame represents a feature, and the columns contain the properties of that feature for each group, including the feature name, group name, proportion of NA values for that feature in that group, randomness p-value, and maximum fold change over time.

### Usage

```
summarise_feature_property(se_obj_merged_list)
```

**Arguments**

se\_obj\_merged\_list

A list of merged SummarizedExperiment objects, output of calc\_feature\_property() function

**Value**

A data frame summarizing the feature properties across all groups, with each row representing a feature and columns containing the properties including the feature name, group name, and the proportion of NA values for that feature in that group, randomness p-value, and maximum fold change over time.

**Examples**

```
data(example_obj)
example_obj <- normalise_to_start(example_obj)
example_obj_list <- split_groups(example_obj)
example_obj_merged_list <- merge_replicates(example_obj_list)

example_obj_merged_list <-
  calc_feature_property(example_obj_merged_list, threshold = 0)
property_random_frc <- summarise_feature_property(example_obj_merged_list)
```

---

summarise\_module\_metrics

*Summarise WGCNA module metrics*


---

**Description**

Computes per-module metrics for a WGCNA network: module size, proportion of features assigned, and module membership (kME). Unassigned module is excluded.

**Usage**

```
summarise_module_metrics(net)
```

**Arguments**

net

A WGCNA network object from run\_WGCNA().

**Details**

MeanKME and MeanKME2 are the mean and mean squared module membership ( $kME = \text{cor}(\text{feature}, \text{module eigengene})$ ).  $\text{MeanKME2} = \text{mean}(kME^2)$  is the average proportion of per-feature variance explained by the module eigengene.

**Value**

A data frame with columns: Module, Size, Proportion, MeanKME, MeanKME2, MedianKME, SDKME, MinKME, MaxKME.

**Examples**

```
data(example_net)
summarise_module_metrics(example_net)
```

---

|                                  |
|----------------------------------|
| summarise_module_pattern         |
| <i>Summarise module patterns</i> |

---

**Description**

Summarise module patterns by integrating WGCNA output with Trendy results

**Usage**

```
summarise_module_pattern(module, trendy_summary)
```

**Arguments**

- module            A data frame with columns "Feature" and "Module"
- trendy\_summary   A data frame of trendy results, output of summarise\_Trendy()

**Value**

A list of data frames, each data frame shows the pattern counts in a module

**Examples**

```
data("example_res_list")
trendy_summary <- summarise_Trendy(example_res_list)

data(example_net)
example_module <- WGCNA_module(example_net)
summarise_module_pattern(example_module, trendy_summary)
```

---

|                  |                                 |
|------------------|---------------------------------|
| summarise_Trendy | <i>Summarise Trendy results</i> |
|------------------|---------------------------------|

---

**Description**

Summarise Trendy results into data frame

**Usage**

```
summarise_Trendy(res_list, ...)
```

**Arguments**

- res\_list      A list of Trendy analysis results, output of run\_Trendy()
- ...            Additional arguments to be passed to the Trendy::topTrendy()

**Value**

A data frame of summary results of Trendy, including breakpoints, segment slopes and p-values for each fitted feature in each group. A "Pattern" column is added to show the combination of segment trends (up, down, stable) for each feature.

**Examples**

```
data("example_res_list")
trendy_summary <- summarise_Trendy(example_res_list)
```

---

|              |                             |
|--------------|-----------------------------|
| theme_custom | <i>Custom ggplot2 theme</i> |
|--------------|-----------------------------|

---

**Description**

A minimal theme used by all TiDEomics plotting functions, built on theme\_minimal(). Exported so users can apply it to their own plots for consistent styling.

**Usage**

```
theme_custom(base_size = 8, panel_border = FALSE, legend_position = "right")
```

**Arguments**

- base\_size      Base font size (default: 8).
- panel\_border   Logical, whether to draw a border around the panel (default: FALSE).
- legend\_position      Legend position (default: "right").

**Value**

A ggplot2 theme object.

**Examples**

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_custom(base_size = 10)
```

---

tutorial\_data

*Dataset for TiDEomics tutorial, expression matrix*

---

**Description**

A subset of GSE263759 data set published in [Integrated time-series analysis and high-content CRISPR screening delineate the dynamics of macrophage immune regulation](#)

**Usage**

```
data(tutorial_data)
```

**Format**

A data.frame with 500 rows and 41 variables:

**Feature** Feature ID, e.g. gene symbols

**Sample1, Sample2, ...** log2(CPM + 1) normalised expression values

**Details**

Code for preparing the data is available in `inst/script/tutorial_input.R`

- Ensembl IDs were mapped to symbols, genes with all zero counts were excluded.
- Use time 0 untreated samples for other groups' time 0.
- Include "untreated", "IFNbeta", "IFNgamma", "LPS" groups.
- Sample 500 random genes.
- Normalised to log2(CPM + 1).

**Source**

GSE263759

---

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| tutorial_sample_info | <i>Dataset for TiDEomics tutorial, sample information</i> |
|----------------------|-----------------------------------------------------------|

---

## Description

A subset of GSE263759 data set published in [Integrated time-series analysis and high-content CRISPR screening delineate the dynamics of macrophage immune regulation](#)

## Usage

```
data(tutorial_sample_info)
```

## Format

A data.frame with 40 rows and 5 variables:

**Sample** Sample ID

**Group** Experimental group (untreated, different treatments)

**Time** Time point

**Replicate** Replicate ID for each group and time point

**Batch** Batch information

## Details

Code for preparing the data is available in `inst/script/tutorial_input.R`

- Ensembl IDs were mapped to symbols, genes with all zero counts were excluded.
- Use time 0 untreated samples for other groups' time 0.
- Include "untreated", "IFNbeta", "IFNgamma", "LPS" groups.
- Sample 500 random genes.

## Source

GSE263759

---

|              |                                                          |
|--------------|----------------------------------------------------------|
| WGCNA_module | <i>Convert WGCNA output to feature-module data frame</i> |
|--------------|----------------------------------------------------------|

---

**Description**

Converts the module assignments from `run_WGCNA()` output into two-column data.frame with `Feature` and `Module` columns, expected by `plot_modules_v()`, `plot_modules_h()`, `summarise_module_pattern()`, and all enrichment functions.

**Usage**

```
WGCNA_module(net, exclude_grey = FALSE)
```

**Arguments**

|                           |                                                                                                    |
|---------------------------|----------------------------------------------------------------------------------------------------|
| <code>net</code>          | The output of <code>run_WGCNA()</code> (a list containing <code>\$colors</code> )                  |
| <code>exclude_grey</code> | Logical. If TRUE, features assigned to module 0 (grey / unassigned) are removed.<br>Default: FALSE |

**Value**

A data.frame with columns `Feature` (character) and `Module` (factor ordered by decreasing module size). When `exclude_grey = TRUE`, grey/unassigned features are excluded.

**Examples**

```
data(example_net)
module <- WGCNA_module(example_net)
```

# Index

## \* datasets

- example\_go, [16](#)
- example\_net, [17](#)
- example\_obj, [17](#)
- example\_res\_list, [18](#)
- tutorial\_data, [62](#)
- tutorial\_sample\_info, [63](#)

- calc\_feature\_property, [3](#)
- calc\_mean\_sd, [4](#)
- create\_input, [5](#)

- DE\_between\_group, [6](#)
- DE\_between\_group(), [20](#)
- DE\_between\_time, [8](#)
- DE\_between\_time(), [20](#)
- decomp\_variance, [9](#)

- enrich\_msigdb, [10](#)
- enrich\_msigdb(), [20](#), [21](#)
- enrichGO\_list, [12](#)
- enrichGO\_list(), [20](#), [21](#)
- enrichGO\_rank, [14](#)
- enrichR\_list, [15](#)
- enrichR\_list(), [20](#), [21](#)
- example\_go, [16](#)
- example\_net, [17](#)
- example\_obj, [17](#)
- example\_res\_list, [18](#)
- extract\_hubs, [18](#)
- extract\_segment\_trends, [19](#)

- flatten\_DE, [20](#)
- flatten\_enrich, [20](#)

- get\_custom\_palette, [21](#)
- group\_specific\_features, [22](#)

- impute\_groups, [23](#)

- merge\_groups, [24](#)

- merge\_replicates, [25](#)

- normalise\_to\_start, [26](#)

- plot\_breakpoints, [26](#)
- plot\_cor\_matrix, [27](#)
- plot\_cv, [28](#)
- plot\_DE\_between\_group, [29](#)
- plot\_DE\_between\_time, [30](#)
- plot\_distribution, [31](#)
- plot\_GO, [32](#)
- plot\_ID, [33](#)
- plot\_missing, [34](#)
- plot\_modules\_h, [35](#)
- plot\_modules\_v, [37](#)
- plot\_pca, [39](#)
- plot\_pca\_3D, [40](#)
- plot\_pca\_arrows, [41](#)
- plot\_pca\_by\_group, [42](#)
- plot\_segments, [43](#)
- plot\_trend, [44](#)
- plot\_umap, [45](#)
- plot\_umap\_by\_group, [46](#)
- plot\_variance, [47](#)
- plot\_volcano, [48](#)
- plot\_WGCNA, [50](#)
- prepare\_tide, [51](#)
- prepare\_WGCNA, [53](#)

- run\_Trendy, [55](#)
- run\_WGCNA, [56](#)

- set\_custom\_palette, [57](#)
- split\_groups, [58](#)
- summarise\_feature\_property, [58](#)
- summarise\_module\_metrics, [59](#)
- summarise\_module\_pattern, [60](#)
- summarise\_Trendy, [61](#)

- theme\_custom, [61](#)
- tutorial\_data, [62](#)

tutorial\_sample\_info, [63](#)

WGCNA\_module, [64](#)