Package 'SUMO'

October 16, 2025

Title Generating Multi-Omics Datasets for Testing and Benchmarking

Version 1.2.3

Description
Provides tools to simulate multi-omics datasets with predefined signal structures. The generated data can be used for testing, validating, and benchmarking integrative analysis methods such as factor models and clustering approaches. This version includes enhanced signal customization, visualization tools (scatter, histogram, 3D), MOFA-based analysis pipelines, PowerPoint export, and statistical profiling of datasets. Designed for both method development and teaching, SUMO supports real and synthetic data pipelines with interpretable outputs. Tini, Giulia, et al (2019) <doi:10.1093/bib/bbx167>.

License CC BY 4.0

Encoding UTF-8

Depends R (>= 4.2)

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0), MOFAdata, MOFA2, fabia, basilisk, reticulate, flextable, rvg,

Config/testthat/edition 3

Imports ggplot2, gridExtra, rlang, grid, jsonlite, ragg, stats, graphics, utils, dplyr, readr, readxl, stringr, data.table, magrittr, tidyverse, systemfonts, officer

Collate 'SUMO.R' 'compute_means_vars.R' 'convert_legacy_to_current_std.R' 'demo_multiomics_analysis.R' 'divide_vector.R' 'divide_features_one.R' 'divide_features_two.R' 'divide_samples.R' 'divide_samples_alternative.R' 'feature_selection_one.R' 'feature_selection_two.R' 'globals.R' 'plot_factor.R' 'plot_simData.R' 'plot_weights.R' 'pretrained.R' 'simulateMultiOmics.R' 'simulate_twoOmicsData.R' 'sumo_py.R'

NeedsCompilation no

Author Bernard Isekah Osang'ir [aut, cre] (ORCID: <https://orcid.org/0000-0002-5557-3602>), Ziv Shkedy [ctb],

Surya Gupta [ctb],
Jürgen Claesen [ctb]

Maintainer Bernard Isekah Osang'ir <Bernard.Osangir@sckcen.be>

Repository CRAN

Date/Publication 2025-10-16 16:40:02 UTC

Contents

as_multiomics 2
compute_means_vars 3
demo_multiomics_analysis 4
divide_features_one 6
divide_features_two 6
divide_samples 7
divide_vector 8
feature_selection_one 8
feature_selection_two 9
plot_factor 9
plot_simData 10
plot_weights 11
simulateMultiOmics 13
simulate_twoOmicsData 15
SUMO 16
sumo_load_pretrained_mofa 17
sumo_pretrained_mofa_available 18
sumo_pretrained_mofa_path 18
sumo_setup_mofa 19

Index **20**

`as_multiomics` *Convert legacy objects (e.g., from* `simulate_twoOmicsData()`*) to the current standardized structure used by downstream tools.*

Description

Normalizes outputs that may contain fields like `omic.one`, `omic.two`, `list_betas/beta`, and `list_deltas/delta` into a unified structure with `omics`, `list_betas` (per-omic), `signal_annotation` (samples and features), and a `factor_map`. If the input already matches the current schema, it is returned unchanged.

Usage

```
as_multiomics(x)
```

Arguments

x	A list-like legacy simulation object (e.g., produced by simulate_twoOmicsData() or similar helpers). May also be a list with element omics that is itself a list of matrices.

Details

Coerce legacy simulation outputs to the current multi-omics schema

Value

A standardized list with components:

- concatenated_datasets — list of one matrix cbind(omic1, omic2, ...).
- omics — named list of matrices (samples x features).
- list_alphas — per-factor sample scores (named alpha1, alpha2, ...).
- list_betas — list per-omic with per-factor loadings (named beta1, beta2, ...).
- signal_annotation — list with samples and features indices per factor and per-omic.
- factor_structure — best-effort label: "shared", "unique", or "mixed".
- factor_map — which omics each factor loads on.

---

compute_means_vars *Compute Summary Statistics for a List of Datasets*

Description

Computes overall, row-wise, and column-wise means and standard deviations for each dataset in a list. Also provides average statistics across datasets.

Usage

```
compute_means_vars(data_list)
```

Arguments

data_list	A list of numeric matrices or data frames. Each entry should be a matrix or data frame with numeric values.

Value

A named list containing:

- Overall mean and SD for each dataset.
- Average row-wise mean and SD.
- Average column-wise mean and SD.
- mean_smp: Average row-wise mean across all datasets.
- sd_smp: Average row-wise SD across all datasets.

Examples

```
# Example using simulated matrices
set.seed(123)
dataset1 <- matrix(rnorm(100, mean = 5, sd = 2), nrow = 10, ncol = 10)
dataset2 <- matrix(rnorm(100, mean = 10, sd = 3), nrow = 10, ncol = 10)
data_list <- list(dataset1, dataset2)
results <- compute_means_vars(data_list)
print(results)

## Not run:
# Example using real experimental data (requires MOFAdata)
if (requireNamespace("MOFAdata", quietly = TRUE)) {
  utils::data("CLL_data", package = "MOFAdata")
  CLL_data2 <- CLL_data[c(2, 3)]
  results <- compute_means_vars(CLL_data2)
  print(results)
}

## End(Not run)
```

---

demo_multiomics_analysis

Demonstration of SUMO Utility in Multi-Omics Analysis using MOFA2

Description

Run a complete MOFA2 workflow on either SUMO-generated data or the real-world CLL dataset. The function handles preprocessing and model training (preferring MOFA2's basilisk; falling back to a user reticulate env if configured), or loads a bundled pretrained model. It then creates summary visualizations and can export a multi-slide PowerPoint report.

Usage

```
demo_multiomics_analysis(
  data_type = c("SUMO", "real_world"),
  export_pptx = TRUE,
  verbose = TRUE,
  use_pretrained = c("auto", "always", "never")
)
```

Arguments

data_type	Character. "SUMO" (synthetic) or "real_world" (CLL).
export_pptx	Logical. If TRUE, write a PowerPoint report (multiple slides). Default TRUE.
verbose	Logical. If TRUE, print progress messages. Default TRUE.
use_pretrained	One of "auto", "always", "never".

- "auto": train if a backend is available, otherwise load a pretrained model.
- "always": always load a pretrained model and skip training.
- "never": always train (requires a working Python backend for MOFA2).

Details

Backend selection. Training prefers MOFA2's basilisk backend when available; otherwise a reticulate/conda environment is used if configured via `sumo_setup_mofa()`. If neither is available and `use_pretrained = "auto"`, the function loads a pretrained model shipped under `inst/extdata/`.

PowerPoint contents (when `export_pptx = TRUE`):

- Title slide (dataset label and generation date)
- Data overview: `plot_data_overview()`
- Factor correlation: `plot_factor_cor()`
- Variance explained:
  - by view x factor, and
  - by group x factor (with totals) via `plot_variance_explained()`
- Factor visualizations:
  - beeswarms for factors 1-3 via `plot_factor()`
  - a customized F1 vs F2 plot
  - scatter plots of factor combinations via `plot_factors()`
- Feature weights:
  - `plot_weights()` and `plot_top_weights()` (first view, factor 1)
- Input-data views:
  - heatmap via `plot_data_heatmap()` and
  - feature-factor scatter via `plot_data_scatter()` (second view if present)
- Non-linear embedding (if available): t-SNE via `run_tsne()` + `plot_dimred()`
- Table slides (heads/summaries):
  - sample metadata (head)
  - total R^2 per view/group (head)
  - R^2 per factor x view/group (head)
  - dimensions summary (factors/weights/data)
  - long-format heads from `get_factors()`, `get_weights()`, `get_data()`

Plots are rasterized for portability when embedding in PPT (vector export is used when supported).

Value

Invisibly returns the trained (or loaded) MOFA model object.

See Also

`simulate_twoOmicsData()`, `plot_factor()`, `plot_weights()`, `sumo_setup_mofa()`, `sumo_mofa_backend()`, `sumo_load_pretrained_mofa()`

Examples

```
if (
  interactive() &&
  requireNamespace("MOFA2", quietly = TRUE) &&
  requireNamespace("systemfonts", quietly = TRUE) &&
  utils::packageVersion("systemfonts") >= "1.1.0" &&
  identical(Sys.getenv("NOT_CRAN"), "true")
) {
  # Use pretrained models (no Python needed):
  demo_multiomics_analysis("SUMO",       export_pptx = TRUE, use_pretrained = "always")
  demo_multiomics_analysis("real_world", export_pptx = TRUE, use_pretrained = "always")

  # To train (when basilisk or a reticulate env is available):
  # demo_multiomics_analysis("real_world", export_pptx = TRUE, use_pretrained = "never")
}
```

---

divide_features_one *Dividing features to create vectors with signal in the first omic for single data*

Description

Dividing features to create vectors with signal in the first omic for single data

Usage

```
divide_features_one(n_features_one, num.factor)
```

Arguments

n_features_one	number of features of first omic
num.factor	number of factor = '1'

---

divide_features_two *Dividing features to create vectors with signal in the second omic for single data*

Description

Dividing features to create vectors with signal in the second omic for single data

Usage

```
divide_features_two(n_features_two, num.factor)
```

Arguments

n_features_two	number of features of second omic
num.factor	type of factors - single or multiple

---

divide_samples *Global Variable*

Description

A global variable used in multiple functions.

This utility function divides a sequence of sample indices into num segments ensuring that each segment meets a specified minimum size. It optionally extracts a subset of each segment based on predefined selection logic:

- For a single group (num = 1): selects a random contiguous sub-vector comprising between 10% and 55% of the total samples.
- For multiple groups (num > 1): selects a contiguous sub-vector comprising approximately 75% of each segment.

Usage

```
divide_samples(n_samples, num, min_size)

divide_samples(n_samples, num, min_size)
```

Arguments

n_samples	Integer. Total number of samples to divide.
num	Integer. Number of desired segments or latent factors.
min_size	Integer. Minimum size (length) allowed for each segment.

Details

This function is primarily used for randomized simulation of sample blocks, useful in bootstrapping, subsampling, or simulating latent factor scores across multi-omics datasets.

Value

A list of integer vectors. Each vector contains a sequence of indices representing a subsample of the corresponding segment.

Examples

```
divide_samples(n_samples = 100, num = 3, min_size = 10)
divide_samples(n_samples = 50, num = 1, min_size = 5)
```

`divide_vector` *#' Global Variable #' #' A global variable used in multiple functions. #' #'*

Description

#' Global Variable #' #' A global variable used in multiple functions. #' #'

Usage

```
divide_vector(n_samples, num, min_size)
```

Arguments

`n_samples`	number of samples
`num`	number of factors
`min_size`	Minimum length of any samples scores #' ## ~ Updated IN USE (IN USE): Simulate the samples scores (IN USE) ~

`feature_selection_one` *Dividing features to create vectors with signal in the first omic*

Description

Dividing features to create vectors with signal in the first omic

Usage

```
feature_selection_one(n_features_one, num.factor, no_factor)
```

Arguments

`n_features_one`	number of features of first omic
`num.factor`	type of factors - single or multiple
`no_factor`	number of factors

feature_selection_two *Dividing features to create vectors with signal in the second omic*

Description

Dividing features to create vectors with signal in the second omic

Usage

```
feature_selection_two(n_features_two, num.factor, no_factor)
```

Arguments

n_features_two	number of features of second omic
num.factor	type of factors - single or multiple
no_factor	number of factors

plot_factor *Visualization of factor scores (ground truth)*

Description

Scatter or histogram plots of sample-level factor scores from simulated multi-omics data, using scores from list_alphas and list_gammas.

Usage

```
plot_factor(
  sim_object = NULL,
  factor_num = NULL,
  type = "scatter",
  show.legend = TRUE
)
```

Arguments

sim_object	R object containing simulated data output from `simulate_twoOmicsData` and `simulateMultiOmics`.
factor_num	Integer or "all". Which factor(s) to plot.
type	Character. Either "scatter" (default) or "histogram" for plot type.
show.legend	Logical. Whether to show legend in plots. Default is TRUE.

Examples

```
output_obj <- simulate_twoOmicsData(
  vector_features = c(4000,3000),
  n_samples = 100,
  n_factors = 2,
  snr = 2.5,
  num.factor = 'multiple',
  advanced_dist = 'mixed')

plot_factor(sim_object = output_obj, factor_num = 1)
plot_factor(sim_object = output_obj, factor_num = 'all', type = 'histogram')
```

---

plot_simData *Visualize simulated multi-omics data as a heatmap*

Description

Quick visualization of simulated omics data as a base R heatmap. You can plot the merged/concatenated matrix across all omics or a single omic layer. Optionally permute sample and/or feature order (with a seed) to conceal block structure for sanity checks.

Usage

```
plot_simData(
  sim_object,
  data = "merged",
  type = "heatmap",
  permute = FALSE,
  permute_seed = NULL,
  permute_samples = TRUE,
  permute_features = TRUE
)
```

Arguments

sim_object	List-like simulation result. Must contain omics, a named list of numeric matrices (samples in rows, features in columns).
data	Character. Which matrix to visualize: • "merged" or "concatenated": column-bind all omics (default: "merged"). • a single omic name present in names(sim_object$omics).
type	Character. Plot type. Currently only "heatmap" is supported.
permute	Logical. If TRUE, apply permutations according to permute_samples and permute_features. Default: FALSE.
permute_seed	Integer or NULL. If not NULL, sets RNG seed **once** for reproducible permutations. Default: NULL.

permute_samples
: Logical. If TRUE and permute = TRUE, permute sample order (rows). Default: TRUE.

permute_features
: Logical. If TRUE and permute = TRUE, permute feature order (columns). Default: TRUE.

Details

The function expects sim_object$omics to be a **named list** of numeric matrices with the **same number of rows** (samples). For data = "merged" (or "concatenated"), all omic matrices are column-bound in their current order (subject to optional permutation) and plotted together.

Value

Invisibly returns the numeric matrix that was plotted (after any permutations).

See Also

simulateMultiOmics

Examples

```
set.seed(123)
sim_object <- simulate_twoOmicsData(
  vector_features = c(4000, 3000),
  n_samples = 100,
  n_factors = 2,
  snr = 2.5,
  num.factor = "multiple",
  advanced_dist = "mixed"
)
output_obj = as_multiomics(sim_object)

# Merged (concatenated) heatmap
plot_simData(output_obj, data = "merged", type = "heatmap")

# Single omic with reproducible permutation
plot_simData(output_obj, data = "omic2", permute = TRUE, permute_seed = 123)
```

---

plot_weights *Visualize feature loadings (weights)*

Description

Generate scatter or histogram plots of feature loadings (weights) from simulated or real multi-omics data. Supports per-omic views and, when available, an integrated view.

Usage

```
plot_weights(
  sim_object,
  omic = 1,
  factor_num = 1,
  type = "scatter",
  show.legend = TRUE
)
```

Arguments

sim_object	A multi-omics object (e.g., from simulate_twoOmicsData() and as_multiomics()).
omic	Integer or character. Which view to plot: 1 (omic.one), 2 (omic.two), or "integrated" (if present). Default 1.
factor_num	Integer or "all". Which factor(s) to visualize. Default 1.
type	Character. Plot type: "scatter" or "histogram". Default "scatter".
show.legend	Logical. Whether to show the legend. Default TRUE.

Value

A ggplot object (single plot) or a grob returned by grid.arrange when multiple panels are combined.

Examples

```
output_obj <- simulate_twoOmicsData(
  vector_features = c(4000, 3000),
  n_samples = 100,
  n_factors = 2,
  signal.samples = NULL,
  signal.features.one = NULL,
  signal.features.two = NULL,
  snr = 2.5,
  num.factor = "multiple",
  advanced_dist = "mixed"
)

output_obj <- as_multiomics(output_obj)

plot_weights(
  sim_object = output_obj,
  factor_num = 1,
  omic = 1,
  type = "scatter",
  show.legend = FALSE
)

plot_weights(
  sim_object = output_obj,
  factor_num = 2,
  omic = 2,
```

```
  type = "histogram"
)
```

`simulateMultiOmics` *Simulation of omics with predefined single or multiple latent factors in multi-omics*

Description

Simulate multiple omics (>=2) datasets with predefined sample-level latent factors and corresponding feature-level signal regions. Each omic has unique signal structure, noise profile, and feature space.

Usage

```
simulateMultiOmics(
  vector_features,
  n_samples,
  n_factors,
  snr = 2,
  signal.samples = c(5, 0.05),
  signal.features = NULL,
  factor_structure = "mixed",
  num.factor = "multiple",
  seed = NULL,
  real_stats = FALSE,
  real_means_vars = NULL
)
```

Arguments

`vector_features`	Integer vector of number of features per omic (length k for k omics).
`n_samples`	Total number of samples.
`n_factors`	Number of latent factors.
`snr`	Numeric. Signal-to-noise ratio.
`signal.samples`	Length-2 vector (mean, sd) for sample-level signal values.
`signal.features`	List of length-k vectors (mean, sd) for each omic's feature-level signal.
`factor_structure`	Character. One of: "shared", "unique", "mixed", "partial", "custom".
`num.factor`	Character. Either "multiple" (default) or "single" factor mode.
`seed`	Integer seed for reproducibility (optional).
`real_stats`	Logical. If TRUE, noise variance and mean are derived from `real_means_vars`.
`real_means_vars`	Optional list of named vectors per omic: c(mean=..., var=...). Required if `real_stats = TRUE`.

Details

This function generates synthetic multi-omics datasets for benchmarking integrative methods. Each omic layer has its own feature distribution and noise characteristics.

Key properties:

- Sample signal blocks for each latent factor are non-overlapping and randomly spaced.
- Feature signal blocks per omic are also non-overlapping and assigned per factor.
- Noise can be modeled using either standard SNR scaling (default) or real data statistics (if `real_stats = TRUE`).
- Omics without assigned signal factors still receive background noise.

Value

A list containing:

- `omics`: List of omic matrices.
- `concatenated_datasets`: Merged matrix of all omics.
- `list_alphas`: Sample-level latent factor values.
- `list_betas`: Feature-level loadings for each omic and factor.
- `signal_annotation`: List of feature and sample signal blocks.
- `factor_structure`: Input parameter.
- `factor_map`: Map of which omics each factor affects.

Examples

```
# Example 1: Use standard SNR scaling (default)
sim1 <- simulateMultiOmics(
  vector_features = c(3000, 2500, 2000),
  n_samples = 100,
  n_factors = 3,
  snr = 3,
  signal.samples = c(5, 1),
  signal.features = list(
    c(3, 0.05),
    c(2.5, 0.05),
    c(2, 0.05)
  ),
  factor_structure = "mixed",
  num.factor = "multiple",
  seed = 123
)
plot_simData(sim_object = sim1, data = "merged", type = "heatmap")

# Example 2: Use real stats for noise modeling
sim2 <- simulateMultiOmics(
  vector_features = c(3000, 2500, 2000),
  n_samples = 100,
  n_factors = 3,
```

```
  snr = 3,
  signal.samples = c(5, 1),
  signal.features = list(
    c(3, 0.05),
    c(2.5, 0.05),
    c(2, 0.05)
  ),
  factor_structure = "mixed",
  num.factor = "multiple",
  real_stats = TRUE,
  real_means_vars = list(
    c(mean = 5, var = 1),
    c(mean = 4.5, var = 0.8),
    c(mean = 4.0, var = 0.6)
  ),
  seed = 123
)
plot_simData(sim_object = sim2, data = "merged", type = "heatmap")
```

`simulate_twoOmicsData` *Simulation of omics with predefined single or multiple latent factors in multi-omics*

Description

Simulates two high-dimensional omics datasets with customizable latent factor structures. Users can control the number and type of factors (shared, unique, mixed), the signal-to-noise ratio, and the distribution of signal-carrying samples and features. The function is flexible for benchmarking multi-omics integration methods under various controlled scenarios.

Usage

```
simulate_twoOmicsData(
  vector_features = c(2000, 2000),
  n_samples = 50,
  n_factors = 3,
  signal.samples = NULL,
  signal.features.one = NULL,
  signal.features.two = NULL,
  num.factor = "multiple",
  snr = 1,
  advanced_dist = NULL,
  ...
)
```

Arguments

vector_features	A numeric vector of length two, specifying the number of features in the first and second omics datasets, respectively.
n_samples	Integer. The number of samples shared between both omics datasets.
n_factors	Integer. Number of latent factors to simulate.
signal.samples	Optional numeric vector of length two: the first element is the mean, and the second is the variance of the number of signal-carrying samples per factor. If NULL, signal assignment is inferred from snr.
signal.features.one	Optional numeric vector of length two: the first element is the mean, and the second is the variance of the number of signal-carrying features per factor in the first omic.
signal.features.two	Optional numeric vector of length two: the first element is the mean, and the second is the variance of the number of signal-carrying features per factor in the second omic.
num.factor	Character string. Either 'single' or 'multiple'. Determines whether to simulate a single latent factor or multiple factors.
snr	Numeric. Signal-to-noise ratio used to estimate the background noise. The function uses this value to infer the proportion of signal versus noise in the simulated datasets.
advanced_dist	Character string. Specifies how latent factors are distributed when num.factor = 'multiple'. Options include: '', NULL, 'mixed', 'omic.one', 'omic.two', or 'exclusive'.
...	Additional arguments (not currently used).

SUMO *SUMO: Simulation Utilities for Multi-Omics Data*

Description

It provides tools for simulating complex multi-omics datasets, enabling researchers to generate data that mirrors the biological intricacies observed in real-world omics studies. This package addresses a critical gap in current bioinformatics by offering flexible and customizable methods for synthetic multi-omics data generation, supporting method development, validation, and benchmarking.

Details

Key Features:

- **Multi-Omics Simulation**: Generate multi-layered datasets with shared and modality-specific structures.

- **Flexible Generation Engine**: Fine control over samples, noise levels, signal distributions, and latent factor structures.
- **Pipeline-Friendly Design**: Seamlessly integrates with existing multi-omics analysis workflows and packages (e.g., `SummarizedExperiment`, `MultiAssayExperiment`).
- **Visualization Tools**: Built-in plotting functions for exploring synthetic signals, factor structures, and noise.

Main Functions:

- `simulateMultiOmics()`: Simulates multiple (> two) high-dimensional multi-omics datasets.
- `simulate_twoOmicsData()`: Simulates two high-dimensional multi-omics datasets.
- `plot_simData()`: Visualizes generated data at different levels.
- `plot_factor()`: Displays factor scores across samples for signal inspection.
- `plot_weights()`: Visualizes feature loadings to assess signal versus noise.
- `demo_multiomics_analysis()`: Full demo function for applying MOFA on SUMO-generated or real-world data.
- `compute_means_vars()`: Estimate parameters from the real experimental dataset.

Author(s)

Maintainer: Bernard Isekah Osang'ir <Bernard.Osangir@sckcen.be> (ORCID)

Other contributors:

- Ziv Shkedy [contributor]
- Surya Gupta [contributor]
- Jürgen Claesen [contributor]

`sumo_load_pretrained_mofa` *Load a pretrained MOFA model (no Python required)*

Description

Load a pretrained MOFA model (no Python required)

Usage

```
sumo_load_pretrained_mofa(which = c("SUMO", "CLL"))
```

Arguments

`which`	One of "SUMO" or "CLL".

Value

A MOFA object loaded from the shipped HDF5 file.

sumo_pretrained_mofa_available

List pretrained MOFA models included with SUMO

Description

List pretrained MOFA models included with SUMO

Usage

```
sumo_pretrained_mofa_available()
```

Value

Character vector of file names present in inst/extdata (empty if none).

sumo_pretrained_mofa_path

Path to a pretrained MOFA model shipped with SUMO

Description

Path to a pretrained MOFA model shipped with SUMO

Usage

```
sumo_pretrained_mofa_path(which = c("SUMO", "CLL"))
```

Arguments

which	One of "SUMO" or "CLL".

Value

Full file path to the pretrained model.

`sumo_setup_mofa` *Interactive setup for Python 'mofapy2' via reticulate (fallback when basilisk is unavailable)*

Description

Interactive setup for Python 'mofapy2' via reticulate (fallback when basilisk is unavailable)

Usage

```
sumo_setup_mofa(envname = "r-mofa2", py_version = "3.10")
```

Arguments

`envname`	Name of the conda environment to create/use.
`py_version`	Python version (e.g., "3.10").

Value

`TRUE` on success (and persists the env name in SUMO user config).

Index

∗ **MOFA**
demo_multiomics_analysis, 4
∗ **benchmarking**
SUMO, 16
∗ **demo**
demo_multiomics_analysis, 4
∗ **models**
SUMO, 16
∗ **multi-omics**
demo_multiomics_analysis, 4
SUMO, 16
∗ **synthetic-data**
demo_multiomics_analysis, 4

as_multiomics, 2

compute_means_vars, 3

demo_multiomics_analysis, 4
divide_features_one, 6
divide_features_two, 6
divide_samples, 7
divide_vector, 8

feature_selection_one, 8
feature_selection_two, 9

ggplot, *12*
grid.arrange, *12*

plot_factor, 9
plot_factor(), *5*
plot_simData, 10
plot_weights, 11
plot_weights(), *5*

simulate_twoOmicsData, 15
simulate_twoOmicsData(), *5*
simulateMultiOmics, 13
SUMO, 16
SUMO-package *(*SUMO*)*, 16
sumo_load_pretrained_mofa, 17
sumo_load_pretrained_mofa(), *5*
sumo_mofa_backend(), *5*
sumo_pretrained_mofa_available, 18
sumo_pretrained_mofa_path, 18
sumo_setup_mofa, 19
sumo_setup_mofa(), *5*