Package 'WSSmod'

November 3, 2026

Title Weighted Sum Scores for Protein Network Modules

Version 0.1.0

Description Computes Weighted Sum Score (WSS) module-level summaries from proteomic data. Protein-level values are sign-aligned to their association with core biomarkers and combined within 'STRING' network modules, weighted by network connectivity and association strength. Bundled reference modules and a fitted 'joinet' model also allow all module scores to be predicted from age, sex and a small panel of plasma biomarkers, without measuring the full proteomic panel. The method is described in Butler et al. (2026) <doi:10.64898/2026.08.14.26360394>.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports data.table

Suggests glmnet, joinet, knitr, rmarkdown, RNOmni, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

URL https://github.com/Longo-Lab/WSSmod, https://Longo-Lab.github.io/WSSmod/

BugReports https://github.com/Longo-Lab/WSSmod/issues

NeedsCompilation no

Author Robert Butler [aut, cre] (ORCID: <https://orcid.org/0000-0003-0200-8682>), WSSmod authors [cph]

Maintainer Robert Butler <rrbutler@stanford.edu>

Repository CRAN

Date/Publication 2026-11-03 16:00:38 UTC

Contents

calculate_WSS 2
list_prebuilt_wss 4
load_prebuilt_biomarker_reference 5
load_prebuilt_terms 6
load_prebuilt_wss 6
load_prebuilt_wss_model 7
normalize_wss_biomarkers 8
predict_WSS 9
project_rank_norm 10
wss_example_biomarkers 11
wss_example_expression 12

Index **13**

`calculate_WSS` *Calculate Weighted Sum Scores Using Signed Protein Values*

Description

This function calculates module scores by sign-flipping proteins based on their biomarker associations (mean_beta) and weighting by alpha centrality. This approach aligns proteins in the direction of core biomarker progression and weights by network importance and biomarker association strength.

Usage

```
calculate_WSS(
  expr_matrix,
  result = NULL,
  prebuilt = "core_AD_plasma_biomarkers",
  dedup_by = "mean_alpha_scaled",
  min_module_size = 3
)
```

Arguments

`expr_matrix`	A numeric matrix with samples as rows and genes/proteins as columns, already normalized and scaled (see Details). Column names should match the 'symbol' values in the result table. Proteins used by a module must not have missing values (see Details).
`result`	A data.table containing module information with required columns: module, symbol (gene identifiers), mean_beta (for sign-flipping), mean_alpha_scaled (for weighting). A data.frame is converted to a data.table; any other class is an error. If `NULL` (the default), a prebuilt reference set shipped with the package is used instead; see `prebuilt`.

prebuilt — Name of a prebuilt reference set to use when result is NULL. Defaults to "core_AD_plasma_biomarkers". See list_prebuilt_wss() for available options. Ignored if result is supplied.

dedup_by — Name of the column used to rank duplicate gene/module assignments when a gene appears in result more than once; only the highest-ranked assignment per gene is kept. Defaults to "mean_alpha_scaled".

min_module_size — Minimum number of genes a module must retain (after deduplication) to be included in the output. Defaults to 3, matching the module size filter (N > 2) used to build the original core_AD_plasma_biomarkers reference set.

Details

For each module, proteins are sign-flipped based on sign(mean_beta) to align all proteins in the direction of disease progression. The signed values are then weighted by mean_alpha_scaled (network centrality + association strength) and averaged to produce a module score.

Higher module scores indicate greater module activity in the direction of disease progression.

Input scale. The function does no normalization or scaling of expr_matrix; supply values that are already normalized and scaled. Each module score is a weighted average of sign-aligned protein values, so the scores inherit the scale of the input, whether that is a rank-based inverse normal transformed (INT) scale, z-scaled log2 ratios, or another choice appropriate to the platform. The bundled reference set was built from SomaScan data on an INT scale. Scores computed on differently normalized or scaled input are not directly comparable with each other, and predict_WSS() predicts scores on the INT-based scale of the bundled reference.

Missing values. Modules are scored from the proteins that are present in expr_matrix (see the return value for modules with no proteins present), but missing values (NA) in a protein that a module uses are not supported: the function stops with an error that names the affected proteins. Impute the values, or drop the affected proteins or samples, before scoring. Missing values in proteins that no module uses are ignored.

Value

A list containing:

scores — Matrix of Weighted Sum Scores (samples x modules)

summary_stats — data.table with summary statistics for each module score

module_sd — Named vector of standard deviations for each module

module_proteins — List of protein names in each module

A module with none of its proteins in expr_matrix cannot be scored. It triggers a warning, has all-NA scores and a module_sd of 0, and is left out of summary_stats and module_proteins.

See Also

list_prebuilt_wss(), load_prebuilt_wss(), load_prebuilt_terms()

Examples

```
expr_matrix <- matrix(
  c(1, 2, 3, 4, 5, 6, 7, 8),
  nrow = 2,
  dimnames = list(c("sample1", "sample2"), c("geneA", "geneB", "geneC", "geneD"))
)
cluster_result <- data.table::data.table(
  module = c("mod1", "mod1", "mod1", "mod2"),
  symbol = c("geneA", "geneB", "geneC", "geneD"),
  mean_beta = c(1, -1, 1, 1),
  mean_alpha_scaled = c(1, 1, 2, 1)
)

# Calculate Weighted Sum Scores (min_module_size = 1 to keep this toy example)
module_results <- calculate_WSS(expr_matrix, cluster_result, min_module_size = 1)

# Access score matrix
scores <- module_results$scores

# View summary statistics
print(module_results$summary_stats)

# Check standard deviations
print(module_results$module_sd)

# With no `result`, the default bundled reference set is used. The
# simulated example matrix has columns for all of its modules.
default_results <- calculate_WSS(wss_example_expression)
dim(default_results$scores)
```

---

list_prebuilt_wss *List Available Prebuilt WSS Reference Sets*

Description

List Available Prebuilt WSS Reference Sets

Usage

```
list_prebuilt_wss()
```

Value

A character vector of prebuilt reference set names usable as the `prebuilt` argument of `calculate_WSS()` and `load_prebuilt_wss()`.

Examples

```
list_prebuilt_wss()
```

`load_prebuilt_biomarker_reference` *Load a Prebuilt Biomarker Reference Distribution*

Description

Some prebuilt reference sets ship a baseline biomarker reference distribution used by `normalize_wss_biomarkers()` (`method = "project"`) to rank-normalize new raw biomarker values without needing a sizeable cohort of your own. Each biomarker's reference values are drawn from the real baseline cohort, but **independently randomly shuffled** so that no two values across different biomarkers can be attributed to the same original subject. This preserves each biomarker's marginal distribution exactly (every value, tie, and extreme is real) while destroying the joint/multivariate fingerprint that carries re-identification risk.

Usage

```
load_prebuilt_biomarker_reference(prebuilt = "core_AD_plasma_biomarkers")
```

Arguments

`prebuilt`	Name of a prebuilt reference set. See `list_prebuilt_wss()` for available options.

Details

Because of this, **the returned reference has no valid joint structure** — inter-biomarker correlations are deliberately destroyed. Only use it for independent per-biomarker operations (rank/quantile lookups), never for anything requiring multivariate relationships (e.g. correlation or joint modeling).

Value

A data.table with columns `biomarker` and `value` (long format, one row per reference observation), or `NULL` if the given reference set has no associated biomarker reference.

See Also

`normalize_wss_biomarkers()`, `project_rank_norm()`

Examples

```
ref <- load_prebuilt_biomarker_reference()
table(ref$biomarker)
```

load_prebuilt_terms *Get Module Term Labels for a Prebuilt WSS Reference Set*

Description

Some prebuilt reference sets ship a companion table of human-readable term labels for each module, summarized from the underlying pathway enrichment used to build the module.

Usage

```
load_prebuilt_terms(prebuilt = "core_AD_plasma_biomarkers")
```

Arguments

prebuilt	Name of a prebuilt reference set. See list_prebuilt_wss() for available options.

Value

A data.table of module term labels, or NULL if the given reference set has no associated terms file.

Examples

```
head(load_prebuilt_terms())
```

load_prebuilt_wss *Load a Prebuilt WSS Module/Weighting Table*

Description

Reads one of the module weighting tables shipped with WSSmod. The raw table may assign the same protein to more than one module (e.g. once per biomarker it was associated with); calculate_WSS() deduplicates and filters this table before use via dedup_by and min_module_size.

Usage

```
load_prebuilt_wss(prebuilt = "core_AD_plasma_biomarkers")
```

Arguments

prebuilt	Name of a prebuilt reference set. See list_prebuilt_wss() for available options.

Value

A data.table with (at least) columns `module`, `symbol`, `mean_beta`, and `mean_alpha_scaled`, suitable for use as the `result` argument of `calculate_WSS()`.

See Also

`calculate_WSS()`, `list_prebuilt_wss()`, `load_prebuilt_terms()`

Examples

```
ref <- load_prebuilt_wss()
head(ref[, c("module", "symbol", "mean_beta", "mean_alpha_scaled")])
```

---

load_prebuilt_wss_model

Load a Prebuilt WSS Prediction Model

Description

Some prebuilt reference sets ship a fitted `joinet` model (a two-layer elastic net) that predicts all of that reference set's module scores from a small set of covariates, without needing to measure the full proteomic panel. See `predict_WSS()` to predict from this model directly.

Usage

```
load_prebuilt_wss_model(prebuilt = "core_AD_plasma_biomarkers")
```

Arguments

`prebuilt`	Name of a prebuilt reference set. See `list_prebuilt_wss()` for available options.

Value

A list with components `model` (the fitted `joinet` object), `outcomes` (the module names it predicts), `x_cols` (the required predictor columns, in order), and `variant`; or `NULL` if the given reference set has no associated prediction model.

See Also

`predict_WSS()`, `list_prebuilt_wss()`

Examples

```
model_info <- load_prebuilt_wss_model()
model_info$variant
model_info$x_cols
head(model_info$outcomes)
```

normalize_wss_biomarkers *Normalize Raw Biomarker Values for predict_WSS()*

Description

Builds the rank-based inverse-normal-transformed biomarker columns `predict_WSS()` requires, from raw biomarker values, using one of two methods. Output columns keep the same names as the input columns (e.g. `PlasmaPTau181` stays `PlasmaPTau181`) – `predict_WSS()` reads whatever column names its underlying model actually expects.

Usage

```
normalize_wss_biomarkers(
  raw_biomarkers,
  prebuilt = "core_AD_plasma_biomarkers",
  method = c("project", "self")
)
```

Arguments

`raw_biomarkers`	A data.frame or matrix, one row per sample, with columns named by the raw (unsuffixed) biomarker names (e.g. `PlasmaPTau181`, not `PlasmaPTau181_norm`).
`prebuilt`	Name of a prebuilt reference set to use when `method = "project"`. See `list_prebuilt_wss()` for available options. Ignored when `method = "self"`.
`method`	Normalization method:
	`"project"` (default) Project each raw value onto a bundled reference distribution via `project_rank_norm()`. Works for a single sample or a small/differently-distributed cohort, since each value is scored independently against the fixed reference.
	`"self"` Rank-normalize (via `RNOmni::RankNorm()`) within `raw_biomarkers` itself. Appropriate if you have a sizeable cohort of your own with a distribution you're comfortable normalizing against directly, instead of the bundled reference. Requires at least two samples, since a rank transform needs multiple values to rank against.

Value

A data.frame with one normalized column per input biomarker column (same names as `raw_biomarkers`), row names preserved. Combine with `Age`/`Gender` columns and pass to `predict_WSS()`.

See Also

`predict_WSS()`, `project_rank_norm()`, `load_prebuilt_biomarker_reference()`

Examples

```
raw_biomarkers <- data.frame(PlasmaPTau181 = 1.5, PlasmaNfL = 20)
normalize_wss_biomarkers(raw_biomarkers)
```

`predict_WSS` *Predict WSS Module Scores from Core Biomarkers*

Description

Predicts module scores for a prebuilt reference set's `joinet` model (a two-layer elastic net stack) without needing to measure the full proteomic panel. For `"core_AD_plasma_biomarkers"`, this predicts all 75 module scores from Age, Gender, and the 8 core plasma biomarkers used in the original analysis.

Usage

```
predict_WSS(newdata, prebuilt = "core_AD_plasma_biomarkers", type = "response")
```

Arguments

`newdata`	A data.frame or matrix with one row per sample and (at least) the columns required by the model; see `x_cols` in `load_prebuilt_wss_model()` for the exact set and order. For `"core_AD_plasma_biomarkers"` this is `Age` (raw years), `Gender` (numeric, `1` = male, `0` = not male), and the 8 core biomarkers (`PlasmaPTau181`, `PlasmaAB142P`, `PlasmaAB140P`, `PlasmaABRatio`, `PlasmapTau217`, `PlasmapTau217_AB42Ratio`, `PlasmaGFAP`, `PlasmaNfL`). **The biomarker columns must already be rank-based inverse-normal transformed before calling this function** – see `normalize_wss_biomarkers()` to build them from raw biomarker values, including for a single new patient.
`prebuilt`	Name of a prebuilt reference set. See `list_prebuilt_wss()` for available options.
`type`	Prediction type passed to `predict.joinet()`: `"response"` (the default) or `"link"`.

Value

A list with components `base` (first-layer-only predictions) and `meta` (final stacked predictions), each a matrix with one row per sample (row names taken from `newdata`, if present) and one column per module (named by the model's `outcomes`).

See Also

`normalize_wss_biomarkers()`, `load_prebuilt_wss_model()`, `calculate_WSS()`

Examples

```
if (requireNamespace("glmnet", quietly = TRUE) &&
    requireNamespace("joinet", quietly = TRUE)) {
  # Simulated raw biomarkers; works for a single patient too
  raw_biomarkers <- wss_example_biomarkers[, -(1:2)]
  normalized <- normalize_wss_biomarkers(raw_biomarkers)

  newdata <- cbind(
    Age = wss_example_biomarkers$Age,
    Gender = wss_example_biomarkers$Gender,
    normalized
  )
  pred <- predict_WSS(newdata)
  pred$meta[1:3, 1:4]
}
```

`project_rank_norm` *Project New Values onto a Frozen Rank-Normal Reference*

Description

Projects `new_vals` onto the rank-based inverse-normal transform defined by `train_vals`, without ever refitting or observing `new_vals` when determining the transform. Each new value is treated as a single insertion into the frozen, sorted `train_vals` reference: its rank is the count of reference values at or below it (ties in `train_vals` share the same rank), and that rank is converted to a normal quantile using the same Blom-type constant (`k = 0.375`) as `RNOmni::RankNorm()`.

Usage

```
project_rank_norm(train_vals, new_vals, k = 0.375)
```

Arguments

`train_vals`	Numeric vector, the frozen reference distribution (e.g. a baseline cohort's biomarker values). Must not contain `NA`.
`new_vals`	Numeric vector of new values to project onto `train_vals`. May be a single value.
`k`	Blom-type constant, passed through to match `RNOmni::RankNorm()`'s default.

Details

This is useful for scoring new samples (including a single new sample) against a fixed reference cohort without leaking information from the new samples into the reference's distribution – e.g. avoiding the train/test leakage that results from rank-normalizing a pooled reference+new-sample dataset together.

Note this does **not** reproduce `RNOmni::RankNorm(train_vals)` bit-for-bit when `new_vals` happens to equal `train_vals`: it always treats each new value as an *additional* observation inserted into the `n`-sample reference (using `n + 1` in the denominator), rather than as one of the original `n` samples. The two are close but not identical.

Value

A numeric vector the same length as `new_vals`, giving each value's projected normal quantile.

See Also

`normalize_wss_biomarkers()`

Examples

```
train_vals <- rnorm(200)
project_rank_norm(train_vals, new_vals = c(-1, 0, 1))
```

---

wss_example_biomarkers

Simulated Plasma Biomarkers for WSS Prediction Examples

Description

A small, fully simulated table of raw plasma biomarker values for trying out `normalize_wss_biomarkers()` and `predict_WSS()`. Biomarkers are drawn from log-normal distributions at plausible scales. The data are random draws and contain no participant data; they carry no biological meaning and are for demonstration only.

Usage

```
wss_example_biomarkers
```

Format

A data.frame with 20 rows (named `patient01` to `patient20`) and 10 columns:

Age Age in years.

Gender `1` = male, `0` = not male.

PlasmaPTau181, PlasmaAB142P, PlasmaAB140P, PlasmaABRatio, PlasmapTau217, PlasmapTau217_AB42Ratio, Pl
Raw (un-normalized) plasma biomarker values.

Source

Simulated; see `data-raw/simulate_examples.R` in the package source.

See Also

normalize_wss_biomarkers(), predict_WSS(), wss_example_expression

Examples

```
head(wss_example_biomarkers)
```

---

wss_example_expression

Simulated Expression Matrix for WSS Examples

Description

A small, fully simulated samples-by-proteins matrix for trying out calculate_WSS(). Its columns are the 887 proteins (gene symbols) that survive the default deduplication and module-size filter of the bundled "core_AD_plasma_biomarkers" reference set, so calculate_WSS(wss_example_expression) scores all 75 modules without warnings. Values are on a z-score-like scale, as expected for normalized input.

Usage

```
wss_example_expression
```

Format

A numeric matrix with 50 rows (samples, named sample01 to sample50) and 887 columns (proteins, named by gene symbol).

Details

Each module was given a latent per-sample factor, and its proteins load on that factor in the direction of their mean_beta, plus independent noise. The data are random draws and contain no participant data; they carry no biological meaning and are for demonstration only.

Source

Simulated; see data-raw/simulate_examples.R in the package source.

See Also

calculate_WSS(), wss_example_biomarkers

Examples

```
dim(wss_example_expression)
res <- calculate_WSS(wss_example_expression)
head(res$scores[, 1:3])
```

Index

∗ **datasets**
wss_example_biomarkers, 11
wss_example_expression, 12

calculate_WSS, 2
calculate_WSS(), *4*, *6, 7*, *9*, *12*

list_prebuilt_wss, 4
list_prebuilt_wss(), *3*, *5–9*
load_prebuilt_biomarker_reference, 5
load_prebuilt_biomarker_reference(), *8*
load_prebuilt_terms, 6
load_prebuilt_terms(), *3*, *7*
load_prebuilt_wss, 6
load_prebuilt_wss(), *3, 4*
load_prebuilt_wss_model, 7
load_prebuilt_wss_model(), *9*

normalize_wss_biomarkers, 8
normalize_wss_biomarkers(), *5*, *9*, *11, 12*

predict_WSS, 9
predict_WSS(), *3*, *7, 8*, *11, 12*
project_rank_norm, 10
project_rank_norm(), *5*, *8*

RNOmni::RankNorm(), *8*, *10*

wss_example_biomarkers, 11, *12*
wss_example_expression, *12*, 12