# Package 'weitrix'

October 5, 2020

**Title** Weighted matrix manipulation, finding components of variation with weighted or sparse data

**Version** 1.0.0

**Description** Data type and tools for working with matrices having precision weights and missing data. This package provides a common representation and tools that can be used with many types of high-throughput data. The meaning of the weights is compatible with usage in the base R function ``lm'' and the package ``limma''. Calibrate weights by scaling weights row-wise to account for known predictors of precision. Find PCA-like components of variation even with many missing values, rotated so that individual components may be meaningfully interpreted. DelayedArray matrices and BiocParallel are supported.

**License** LGPL-2.1 | file LICENSE

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 4.0), SummarizedExperiment

**Imports** methods, utils, stats, assertthat, S4Vectors, DelayedArray, DelayedMatrixStats, BiocParallel, BiocGenerics, limma, dplyr, purrr, ggplot2, scales, reshape2, RhpcBLASctl

**Suggests** knitr, rmarkdown, tidyverse, airway, edgeR, topconfects, EnsDb.Hsapiens.v86, org.Sc.sgd.db, AnnotationDbi, testthat (>= 2.1.0)

**RoxygenNote** 7.1.0

**VignetteBuilder** knitr

**biocViews** Software, DataRepresentation, DimensionReduction, GeneExpression, Transcriptomics, RNASeq, SingleCell, Regression

**git_url** https://git.bioconductor.org/packages/weitrix

**git_branch** RELEASE_3_11

**git_last_commit** 626a8e4

**git_last_commit_date** 2020-04-27

**Date/Publication** 2020-10-04

**Author** Paul Harrison [aut, cre] (<https://orcid.org/0000-0002-3980-268X>)

**Maintainer** Paul Harrison <paul.harrison@monash.edu>

# R topics documented:

as_weitrix . . . 2
bless_weitrix . . . 3
components_seq_scree . . . 4
counts_proportions . . . 5
counts_shift . . . 6
matrix_long . . . 7
simwei . . . 7
weitrix_calibrate . . . 8
weitrix_calibrate_trend . . . 8
weitrix_components . . . 9
weitrix_dispersions . . . 11
weitrix_elist . . . 12
weitrix_hill . . . 13
weitrix_randomize . . . 13
weitrix_weights . . . 14
weitrix_x . . . 15

**Index** **16**

---

`as_weitrix` *Convert data to a weitrix*

---

## Description

Ensure data is a weighted matrix or "weitrix". A weitrix is a SummarizedExperiment or subclass thereof with some metadata fields set. If it is ambiguous how to do this, produce an error.

## Usage

```
as_weitrix(object, weights = NULL)
```

## Arguments

| | |
|---|---|
| `object` | Object to convert. |
| `weights` | Optional, weights matrix if not present in `object`. |

## Details

Input can be a matrix or DelayedArray.

Input can be anything the limma package recognizes, notably the limma EList class (for example as output by `voom` or `vooma`).

If weights are not present in "object" and not given with "weights", they default for 0 for NA values and 1 for everything else.

## Value

A SummarizedExperiment object with metadata fields marking it as a weitrix.

## Examples

```
mat <- matrix(c(1,2,NA,3,NA,4), ncol=2)
weitrix <- as_weitrix(mat)

metadata(weitrix)
weitrix_x(weitrix)
weitrix_weights(weitrix)
```

---

`bless_weitrix` *Bless a SummarizedExperiment as a weitrix*

---

## Description

Set metadata entries in a SummarizedExperiment object so that it can be used as a weitrix.

## Usage

```
bless_weitrix(object, x_name, weights_name)
```

## Arguments

| | |
|---|---|
| `object` | A SummarizedExperiment object. |
| `x_name` | Name of the assay containing the observations. |
| `weights_name` | Name of the assay containing the weights. |

## Value

A SummarizedExperiment object with metadata fields marking it as a weitrix.

## Examples

```
mat <- matrix(c(1,2,NA,3,NA,4), ncol=2)
weights <- matrix(c(1,0.5,0,2,0,1), ncol=2)
se <- SummarizedExperiment(assays=list(foo=mat, bar=weights))

weitrix <- bless_weitrix(se, "foo", "bar")

metadata(weitrix)
weitrix_x(weitrix)
weitrix_weights(weitrix)
```

components_seq_scree *Proportion more variance explained by adding components one at a time*

## Description

Based on the output of components_seq, work out how much further variance is explained by adding further components.

## Usage

```
components_seq_scree(comp_seq, rand_comp = NULL)

components_seq_screeplot(comp_seq, rand_comp = NULL)
```

## Arguments

| | |
|---|---|
| comp_seq | A list of Components objects, as produced by components_seq. |
| rand_comp | Optional. A Components object with a single component. This should be based on a randomized version of the weitrix, for example as produced by weitrix_components(weitrix_randomize(my_weitrix),p=1). |

## Details

If rand_comp is given, some possible threshold levels for including further components are also calculated.

The "Parallel analysis" threshold is chosen based on varianced explained by a single component in a randomized weitrix.

The "Optimistic" thresholds are chosen starting from the "Parallel Analysis" threshold. We view the Parallel Analysis threshold as indicating random variance is split amongst an effective number of samples, which will be somewhat smaller than the real number of samples. As each component is accepted, the pool of remaining variance is reduced by its contribution, and also the number of effective samples is reduced by one. The threshold is then the size of the remaining variance pool divided by the effective remaining number of samples. This is a somewhat ad-hoc method, but may indicate more components are justified than by criteria based on a flat threshold.

## Value

components_seq_scree returns a data frame listing the variance explained by each further component.

components_seq_screeplot returns a ggplot2 plot object.

## Examples

```
comp_seq <- weitrix_components_seq(simwei, 4, verbose=FALSE)

components_seq_scree(comp_seq)
components_seq_screeplot(comp_seq)
```

`counts_proportions` *Produce a weitrix of proportions within groups*

## Description

Produce a weitrix of proportions between 0 and 1. The input is read counts at a collection of features in a collection of samples. The features need to be grouped, for example by gene. The proportions will add to 1 within each group.

## Usage

```
counts_proportions(counts, grouping, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `counts` | A matrix of read counts. Rows are peaks and columns are samples. |
| `grouping` | A data frame defining the grouping of features. Should have a column "group" naming the group and a column "name" naming the feature (corresponding to `rownames(counts)`). |
| `verbose` | If TRUE, output some debugging and progress information. |

## Value

A SummarizedExperiment object with metadata fields marking it as a weitrix.

## Examples

```
grouping <- data.frame(
    group=c("A","A","A","B","B"),
    name=c("p1","p2","p3","p4","p5"))

counts <- rbind(
    p1=c(1,2,0),
    p2=c(0,1,0),
    p3=c(1,0,0),
    p4=c(0,0,1),
    p5=c(0,2,1))

wei <- counts_proportions(counts, grouping)

weitrix_x(wei)
weitrix_weights(wei)
rowData(wei)
```

`counts_shift` *Produce a weitrix of shift scores*

## Description

Produce a weitrix of shift scores between -1 and 1. The input is read counts at a collection of peaks (or other features) in a collection of samples. The peaks can be grouped by gene, and are ordered within each gene.

## Usage

```
counts_shift(counts, grouping, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `counts` | A matrix of read counts. Rows are peaks and columns are samples. |
| `grouping` | A data frame defining the grouping of peaks into genes. Should have a column "group" naming the gene and a column "name" naming the peak (corresponding to `rownames(counts)`). Within each group, peak names should be ordered from 5' to 3' position. |
| `verbose` | If TRUE, output some debugging and progress information. |

## Details

For a particular gene, a shift score measures measures the tendency of reads to be upstrand (negative) or downstrand (positive) of the average over all samples. Shift scores range between -1 and 1.

## Value

A SummarizedExperiment object with metadata fields marking it as a weitrix.

## Examples

```
grouping <- data.frame(
    group=c("A","A","A","B","B"),
    name=c("p1","p2","p3","p4","p5"))

counts <- rbind(
    p1=c(1,2,0),
    p2=c(0,1,0),
    p3=c(1,0,0),
    p4=c(0,0,1),
    p5=c(0,2,1))

wei <- counts_shift(counts, grouping)

weitrix_x(wei)
weitrix_weights(wei)
rowData(wei)
```

## matrix_long *Convert a matrix to long form for ggplotting*

### Description

A convenience function which melts the matrix and then joins row and column information.

### Usage

```
matrix_long(
  matrix,
  row_info = NULL,
  col_info = NULL,
  varnames = c("name", "col")
)
```

### Arguments

| | |
|---|---|
| matrix | A matrix, or object that can be converted to a matrix. |
| row_info | Information about rows of the matrix. A data frame, or object that can be converted to a data frame. |
| col_info | Information about columns of the matrix. A data frame, or object that can be converted to a data frame. |
| varnames | Vector of two column names in the output, the first for the row and the second for the column. |

### Value

A data frame containing the matrix and associated information in long format.

### Examples

```
matrix_long(weitrix_x(simwei), rowData(simwei), colData(simwei))
```

## simwei *Simulated weitrix dataset.*

### Description

This is a small simulated weitrix used in examples. There is one component of variation to be found, plus Gaussian noise with variance inversely proportional to the weights.

### Usage

```
simwei
```

### Format

A weitrix object.

## weitrix_calibrate *Adjust weights based on given row dispersions*

### Description

Based on estimated row dispersions, adjust weights in each row.

### Usage

```
weitrix_calibrate(weitrix, dispersions)
```

### Arguments

| | |
|---|---|
| weitrix | A weitrix object, or an object that can be converted to a weitrix with as_weitrix. |
| dispersions | A dispersion for each row. |

### Details

For large numbers of samples this can be based directly on weitrix_dispersions. For small numbers of samples, when using limma, it should be based on a trend-line fitted to known co-variates of the dispersions. This can be done using weitrix_calibrate_trend.

### Value

A SummarizedExperiment object with metadata fields marking it as a weitrix.

### Examples

```
# Adjust weights so dispersion for each row is exactly 1. This is dubious
# for a small dataset, but would be fine for a dataset with many columns.
comp <- weitrix_components(simwei, p=1, verbose=FALSE)
disp <- weitrix_dispersions(simwei, comp)
cal <- weitrix_calibrate(simwei, disp)
weitrix_dispersions(cal, comp)
```

## weitrix_calibrate_trend *Adjust weights by fitting a trend to estimated dispersions*

### Description

Dispersions are estimated using weitrix_dispersions. A trend line is then fitted to log dispersions using a linear model. Weitrix weights are calibrated based on this trend line. Any zero or very-near-zero dispersions are ignored when fitting this model.

### Usage

```
weitrix_calibrate_trend(weitrix, design = ~1, trend_formula = NULL)
```

### Arguments

| | |
|---|---|
| weitrix | A weitrix object, or an object that can be converted to a weitrix with as_weitrix. |
| design | A formula in terms of colData(weitrix or a design matrix, which will be fitted to the weitrix on each row. Can also be a pre-existing Components object, in which case the existing fits (design$row) are used. |
| trend_formula | A formula specification for predicting log dispersion from columns of rowData(weitrix). If absent, metadata(weitrix)$weitrix$trend_formula is used. |

### Value

A SummarizedExperiment object with metadata fields marking it as a weitrix.

### Examples

```
rowData(simwei)$total_weight <- rowSums(weitrix_weights(simwei))

# To estimate dispersions, use a simple model containing only an intercept
# term. Model log dispersion as a straight line relationship with log total
# weight and adjust weights to remove any trend.
cal <- weitrix_calibrate_trend(simwei,~1,trend_formula=~log(total_weight))

# This dataset has few rows, so calibration like this is dubious.
# Predictors in the fitted model are not significant.
summary( metadata(cal)$weitrix$trend_fit )

# Information about the calibration is added to rowData
rowData(cal)


# A Components object may also be used as the design argument.
comp <- weitrix_components(simwei, p=1, verbose=FALSE)
cal2 <- weitrix_calibrate_trend(simwei,comp,trend_formula=~log(total_weight))

rowData(cal2)
```

---

## weitrix_components *Principal components of a weitrix*

---

### Description

Finds principal components of a weitrix. If varimax rotation is enabled, these are then rotated to enhance interpretability.

### Usage

```
weitrix_components(
  weitrix,
  p,
  design = ~1,
  n_restarts = 3,
  max_iter = 1000,
```

```
  tol = 1e-05,
  use_varimax = TRUE,
  initial = NULL,
  verbose = TRUE
)

weitrix_components_seq(
  weitrix,
  p,
  design = ~1,
  n_restarts = 3,
  max_iter = 1000,
  tol = 1e-05,
  use_varimax = TRUE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| weitrix | A weitrix object, or an object that can be converted to a weitrix with as_weitrix. |
| p | Number of components to find. |
| design | A formula referring to colData(weitrix) or a matrix, giving predictors of a linear model for the experimental design. By default only an intercept term is used, i.e. rows are centered before finding components. A more complex formula might be used to account for batch effects. ~0 can be used if rows are already centered. |
| n_restarts | Number of restarts of the iteration to use. |
| max_iter | Maximum iterations. |
| tol | Stop iterating if R-squared increased by less than this amount in an iteration. |
| use_varimax | Use varimax rotation to enhance interpretability of components. |
| initial | Optional, an initial guess for column components (scores). Can have fewer columns than p, in which remaining components are initialized randomly. Can have more columns than p, in which case a randomly chosen subspace is used in each restart. |
| verbose | Show messages about the progress of the iterations. |

## Details

Note that this is a slow numerical method to solve a gnarly problem, for the case where weights are not uniform. The case of uniform weights or weights that can be written as an outer product of row and column weights is somewhat faster, however there are much faster algorithms for this available elsewhere.

An iterative method is used, starting from a random initial set of components. It is possible for this to get stuck at a local minimum. To ameliorate this, the iteration is initially run n_restarts times and the best result used. This is then iterated further. Examine all_R2s in the output to see if this is happening – if the values are not all nearly identital, the iteration is sometimes getting stuck at local minima. Increase n_restarts to increase the odds of finding the global minimum.

## Value

A "Components" object with the following elements accessible using `$`.

**row** Row matrix, aka loadings. Rows are rows in the weitrix, and columns contain the experimental design (usually just an intercept term), and components.

**col** Column matrix, aka scores. Rows are columns in the weitrix, and columns contain fitted coefficients for the experimental design, and components.

**R2** Weighted R squared statistic. The proportion of total variance explained by the components.

**all_R2s** R2 statistics from all restarts. This can be used to check how consistently the iteration finds optimal components.

**ind_design** Column indices associated with experimental design.

**ind_components** Column indices associated with components.

For a result `comp`, the original measurements are approximated by `comp$row %*% t(comp$col)`.

`weitrix_components_seq` returns a list of Components objects, with increasing numbers of components from 1 up to p.

## Functions

- `weitrix_components`: Find a matrix decomposition with the specified number of components.
- `weitrix_components_seq`: Produce a sequence of weitrix decompositions with 1 to p components.

## Examples

```
# Variables in rows, observations in columns, as per Bioconductor convention
dat <- t(iris[,1:4])

# Find two components
comp <- weitrix_components(dat, p=2, max_iter=5, n_restart=1)

# Examine row and col matrices
pairs(comp$row, panel=function(x,y) text(x,y,rownames(comp$row)))
pairs(comp$col)
```

---

# `weitrix_dispersions` *Calculate row dispersions*

---

## Description

Calculate the dispersion of each row. For each observation, this value divided by the weight gives the observation's variance.

## Usage

```
weitrix_dispersions(weitrix, design = ~1)
```

### Arguments

| | |
|---|---|
| weitrix | A weitrix object, or an object that can be converted to a weitrix with as_weitrix. |
| design | A formula in terms of colData(weitrix or a design matrix, which will be fitted to the weitrix on each row. Can also be a pre-existing Components object, in which case the existing fits (design$row) are used. |

### Value

A numeric vector.

### Examples

```
# Using a model just containing an intercept
weitrix_dispersions(simwei, ~1)

# Allowing for one component of variation, the dispersions are lower
comp <- weitrix_components(simwei, p=1, verbose=FALSE)
weitrix_dispersions(simwei, comp)
```

---

## weitrix_elist *Convert a weitrix object to a limma EList object*

---

### Description

The resulting object can be used as input to limma::lmFit for a limma analysis.

### Usage

```
weitrix_elist(weitrix)
```

### Arguments

| | |
|---|---|
| weitrix | A weitrix object. |

### Value

A limma EList object.

### Examples

```
library(limma)

elist <- weitrix_elist(simwei)
design <- model.matrix(~true_score, data=colData(simwei))
fit <- lmFit(elist, design)
# ...perform further limma analysis steps as desired...
```

## weitrix_hill *Calculate Hill numbers (effective number of observations) for rows or columns*

### Description

Effective numbers of observations. `order=0` produces count of non-zero weights. `order=1` produces exp(entropy). `order=2` produces the inverse Simpson index.

### Usage

```
weitrix_hill(weitrix, what = c("row", "col"), order = 2)
```

### Arguments

| | |
|---|---|
| `weitrix` | A weitrix object. |
| `what` | Calculate for rows ("row") (default) or columns ("col")? |
| `order` | Order of the Hill numbers. |

### Value

A numeric vector of effective numbers of observations.

### Examples

```
weitrix_weights(simwei)

weitrix_hill(simwei, what="row", order=0)
weitrix_hill(simwei, what="row", order=1)
weitrix_hill(simwei, what="row", order=2)

weitrix_hill(simwei, what="col", order=0)
weitrix_hill(simwei, what="col", order=1)
weitrix_hill(simwei, what="col", order=2)
```

## weitrix_randomize *Generate a random normally distributed version of a weitrix*

### Description

Values are generated with variance equal to 1/weight. This can be used to see what R-squared would be achieved with purely random data, and therefore an appropriate number of components to use. This is known as Parallel Analysis.

### Usage

```
weitrix_randomize(weitrix)
```

### Arguments

| | |
|---|---|
| weitrix | A weitrix object, or an object that can be converted to a weitrix with as_weitrix. |

### Value

A SummarizedExperiment object with metadata fields marking it as a weitrix.

### See Also

components_seq_screeplot

### Examples

```
weitrix_randomize(simwei)
```

---

## weitrix_weights *Get or set a weitrix object's "weights" matrix*

---

### Description

Gets or sets the appropriate assay in the SummarizedExperiment object.

### Usage

```
weitrix_weights(weitrix)

weitrix_weights(x) <- value
```

### Arguments

| | |
|---|---|
| weitrix | A weitrix object. |
| x | The weitrix to modify. |
| value | The new matrix. |

### Value

A matrix-like object such as a matrix or a DelayedArray.

### Examples

```
weitrix_weights(simwei)
```

`weitrix_x` *Get or set a weitrix object's "x" matrix*

## Description

Gets or sets the appropriate assay in the SummarizedExperiment object.

## Usage

```
weitrix_x(weitrix)

weitrix_x(x) <- value
```

## Arguments

| | |
|---|---|
| `weitrix` | A weitrix object. |
| `x` | The weitrix to modify. |
| `value` | The new matrix. |

## Value

A matrix-like object such as a matrix or a DelayedArray.

## Examples

```
weitrix_x(simwei)

simwei2 <- simwei
weitrix_x(simwei2) <- weitrix_x(simwei2) * 2
```

# Index

∗ **datasets**
simwei, 7

as_weitrix, 2

bless_weitrix, 3

components_seq_scree, 4
components_seq_screeplot, *14*
components_seq_screeplot
*(*components_seq_scree*)*, 4
counts_proportions, 5
counts_shift, 6

matrix_long, 7

simwei, 7

weitrix_calibrate, 8
weitrix_calibrate_trend, 8
weitrix_components, 9
weitrix_components_seq
*(*weitrix_components*)*, 9
weitrix_dispersions, 11
weitrix_elist, 12
weitrix_hill, 13
weitrix_randomize, 13
weitrix_weights, 14
weitrix_weights<- *(*weitrix_weights*)*, 14
weitrix_x, 15
weitrix_x<- *(*weitrix_x*)*, 15