Package ‘Wrench’

July 10, 2025

Depends R (>= 3.5.0)

Type Package

Title Wrench normalization for sparse count data

Version 1.26.0

Description Wrench is a package for normalization sparse genomic count data, like that arising from 16s metagenomic surveys.

Imports limma, matrixStats, locfit, stats, graphics

License Artistic-2.0

LazyData TRUE

RoxygenNote 6.1.0

Suggests knitr, rmarkdown, metagenomeSeq, DESeq2, edgeR

VignetteBuilder knitr

biocViews Normalization, Sequencing, Software

URL `https://github.com/HCBravoLab/Wrench`

BugReports `https://github.com/HCBravoLab/Wrench/issues`

git_url https://git.bioconductor.org/packages/Wrench

git_branch RELEASE_3_21

git_last_commit 369d5b1

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-07-09

Author Senthil Kumar Muthiah [aut],
Hector Corrada Bravo [aut, cre]

Maintainer Hector Corrada Bravo <`hcorrada@gmail.com`>

Contents

`.estimSummary` *Obtain robust means. .*

Description

Obtain robust means. .

Usage

```
.estimSummary(res, estim.type = "s2.w.mean", ...)
```

Arguments

`res`	result structure of `wrench`
`estim.type`	estimator type
`...`	other parameters

Value

a chosen summary statistic

`.getCondLogWeights` *Log Postive-conditional weight computations for wrench estimators.*

Description

Log Postive-conditional weight computations for wrench estimators.

Usage

```
.getCondLogWeights(res)
```

Arguments

`res`	result structure of `wrench`

Value

inverse variance weights when using positive conditional models.

`.getCondWeights` *Postive-conditional weight computations for wrench estimators.*

Description

Postive-conditional weight computations for wrench estimators.

Usage

```
.getCondWeights(res)
```

Arguments

`res`	result structure of `wrench`

Value

positive conditional weights for each sample

`.getHurdle` *Obtains logistic fits for presence/absence and fitted probabilities of a zero occurring.*

Description

This function is used to derive weights for feature-wise compositional estimates. Our (default) intention is to derive these based on average occurrences across the dataset, as just a function of sample depth, and not with particular relevance to groups.

Usage

```
.getHurdle(mat, hdesign = model.matrix(~-1 + log(colSums(mat))),
  pres.abs.mod = TRUE, thresh = FALSE, thresh.val = 1e-08, ...)
```

Arguments

`mat`	count matrix
`hdesign`	design matrix for the logistic; the default is usually sufficient.
`pres.abs.mod`	TRUE if glm regression is for presence or absence. FALSE if glm regression is for counts.
`thresh`	TRUE if numerically one/zero probability occurrences must be thresholded
`thresh.val`	if thresh is true, the numerically one/zero probability occurrences is thresholded to this value
`...`	other parameters

Value

A list with components:

- pi0.fit - list with feature-wise glm.fit objects
- pi0 - matrix with fitted probabilities

`.getMargWeights` *Marginal weight computations for wrench estimators.*

Description

Marginal weight computations for wrench estimators.

Usage

```
.getMargWeights(res, z.adj, ...)
```

Arguments

`res`	result structure of `wrench`
`z.adj`	TRUE if the result structure was generated with `wrench` with `z.adj` set to TRUE.
`...`	other parameters

Value

inverse marginal variances for robust mean computing

`.getReference` *This function generates the reference.*

Description

This function generates the reference.

Usage

```
.getReference(mat, ref.est = "sw.means", ...)
```

Arguments

`mat`	count matrix; rows are features and columns are samples
`ref.est`	reference estimate method
`...`	other parameters

Value

the reference to be used for normalization

`.gets2` *Obtain variances of logged counts.*

Description

Obtain variances of logged counts.

Usage

```
.gets2(mat, design = model.matrix(mat[1, ] ~ 1), plot = FALSE,
  ebs2 = TRUE, smoothed = FALSE, ...)
```

Arguments

`mat`	count matrix; rows are features and columns are samples.
`design`	model matrix for the count matrix
`plot`	if the mean-variance trend function (the same as that of voom) needs to be plot.
`ebs2`	if regularization of variances needs to be performed.
`smoothed`	TRUE if all the variance estimates must be based on the mean-variance trend function.
`...`	other parameters

Value

a vector with variance estimates for logged feature-wise counts.

`.getWeightedMean` *Get weighted means for matrix*

Description

Get weighted means for matrix

Usage

```
.getWeightedMean(mat, w = rep(1, nrow(mat)))
```

Arguments

`mat`	input matrix
`w`	weights

Value

column-wise weighted means.

`.getWeightedMedian` *Get weighted median for matrix*

Description

Get weighted median for matrix

Usage

```
.getWeightedMedian(mat, w = rep(1, nrow(mat)))
```

Arguments

`mat`	input matrix
`w`	weights

Value

column-wise weighted means.

`wrench` *Normalization for sparse, under-sampled count data.*

Description

Obtain normalization factors for sparse, under-sampled count data that often arise with metagenomic count data.

Usage

```
wrench(mat, condition, etype = "w.marg.mean", ebcf = TRUE,
  z.adj = FALSE, phi.adj = TRUE, detrend = FALSE, ...)
```

Arguments

`mat`	count matrix; rows are features and columns are samples
`condition`	a vector with group information on the samples
`etype`	weighting strategy with the following options: • hurdle.w.mean, the W1 estimator in manuscript. • w.marg.mean, the W2 estimator in manuscript. These are appropriately computed depending on whether `z.adj`=TRUE (see below) • s2.w.mean, weight by inverse of feature-variances of logged count data.
`ebcf`	TRUE if empirical bayes regularization of ratios needs to be performed. Default recommended.

`z.adj`	TRUE if the feature-wise ratios need to be adjusted by hurdle probabilities (arises when taking marginal expectation). Default recommended.
`phi.adj`	TRUE if estimates need to be adjusted for variance terms (arises when considering positive-part expectations). Default recommended.
`detrend`	FALSE if any linear dependence between sample-depth and compositional factors needs to be removed. (setting this to TRUE reduces variation in compositional factors and can improve accuracy, but requires an extra assumption that no linear dependence between compositional factors and sample depth is present in samples).
`...`	other parameters

Value

a `list` with components:

- `nf`, *normalization factors* for samples passed. Samples with zero total counts are removed from output.
- `ccf`, *compositional correction factors*. Samples with zero total counts are removed from output.
- `others`, a `list` with results from intermediate computations.
  - `qref`, reference chosen.
  - `design`, design matrix used for computation of positive-part parameters.
  - `s2`, feature-wise variances of logged count data.
  - `r`, (regularized) ratios of feature-wise proportions.
  - `radj`, adjustments made to the regularized ratios based on z.adj and phi.adj settings.

Author(s)

M. Senthil Kumar

Examples

```
#Obtain counts matrix and some group information
require(metagenomeSeq)
data(mouseData)
cntsMatrix <- MRcounts(mouseData)
group <- pData(mouseData)$diet
#Running wrench with defaults
W <- wrench( cntsMatrix, condition=group  )
compositionalFactors <- W$ccf
normalizationFactors <- W$nf

#Introducing the above normalization factors for the most
# commonly used tools is shown below.

#If using metagenomeSeq
normalizedObject <- mouseData
normFactors(normalizedObject) <- normalizationFactors
```

```
#If using edgeR, we must pass in the compositional factors
require(edgeR)
edgerobj <- DGEList( counts=cntsMatrix,
                     group = as.matrix(group),
                     norm.factors=compositionalFactors )

#If using DESeq/DESeq2
require(DESeq2)
deseq.obj <- DESeqDataSetFromMatrix(countData = cntsMatrix,
                                   DataFrame(group),
                                   ~ group )
DESeq2::sizeFactors(deseq.obj) <- normalizationFactors
```

Index