

# Package ‘zitools’

May 2, 2026

**Title** Analysis of zero-inflated count data

**Version** 1.7.0

**Description** zitools allows for zero inflated count data analysis by either using down-weighting of excess zeros or by replacing an appropriate proportion of excess zeros with NA. Through overloading frequently used statistical functions (such as mean, median, standard deviation), plotting functions (such as boxplots or heatmap) or differential abundance tests, it allows a wide range of downstream analyses for zero-inflated data in a less biased manner. This becomes applicable in the context of microbiome analyses, where the data is often overdispersed and zero-inflated, therefore making data analysis extremely challenging.

**License** BSD\_3\_clause + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Imports** phyloseq, pscl, ggplot2, MatrixGenerics, SummarizedExperiment, stats, VGAM, matrixStats, tidyr, tibble, dplyr, DESeq2, reshape2, RColorBrewer, magrittr, BiocGenerics, graphics, utils

**Suggests** knitr, rmarkdown, BiocStyle, testthat (>= 3.0.0), tidyverse, microbiome

**Config/testthat/edition** 3

**Collate** 'data.R' 'globals.R' 'ziMain.R' 'helper.R'  
'inherited\_functions.R' 'plots.R' 'utils.R' 'zitools-package.R'

**Depends** R (>= 4.4.0), methods

**VignetteBuilder** knitr

**biocViews** Software, StatisticalMethod, Microbiome

**URL** <https://github.com/kreutz-lab/zitools>

**BugReports** <https://github.com/kreutz-lab/zitools/issues>

**git\_url** <https://git.bioconductor.org/packages/zitools>

**git\_branch** devel

**git\_last\_commit** e866d12

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-01

**Author** Carlotta Meyring [aut, cre] (ORCID:  
<<https://orcid.org/0009-0000-6201-7615>>)

**Maintainer** Carlotta Meyring <carlotta.meyring@uniklinik-freiburg.de>

## Contents

zitools-package . . . . .	3
* . . . . .	3
+ . . . . .	4
/ . . . . .	5
assays . . . . .	5
boxplot . . . . .	6
colData . . . . .	7
colMeans2 . . . . .	7
colMedians . . . . .	8
cor . . . . .	9
cov . . . . .	10
deinflatedcounts . . . . .	10
heatmap . . . . .	11
inputcounts . . . . .	12
inputdata . . . . .	12
log1p . . . . .	13
log2p . . . . .	14
mean . . . . .	14
median . . . . .	15
MissingValueHeatmap . . . . .	16
model . . . . .	16
mtx . . . . .	17
otu_table . . . . .	17
plot . . . . .	18
quantile . . . . .	19
resample_deinflatedcounts . . . . .	19
rowData . . . . .	20
rowQuantiles . . . . .	21
rowSds . . . . .	22
rowVars . . . . .	23
rowWeightedMeans . . . . .	23
rowWeightedSds . . . . .	24
sample_data . . . . .	25
sd . . . . .	26
show . . . . .	27
subset_feature . . . . .	27
subset_sample . . . . .	28
t . . . . .	29
tax_table . . . . .	30
var . . . . .	30
weighted.mean . . . . .	31
weightedSd . . . . .	32
weights . . . . .	33
Zi-class . . . . .	33

<i>zitoools-package</i>	3
zi2deseq2 . . . . .	34
zi2phyloseq . . . . .	35
ziMain . . . . .	35

**Index** **38**

*zitoools-package*                      *zitoools: Analysis of zero-inflated count data*

**Description**

*zitoools* allows for zero inflated count data analysis by either using down-weighting of excess zeros or by replacing an appropriate proportion of excess zeros with NA. Through overloading frequently used statistical functions (such as mean, median, standard deviation), plotting functions (such as boxplots or heatmap) or differential abundance tests, it allows a wide range of downstream analyses for zero-inflated data in a less biased manner. This becomes applicable in the context of microbiome analyses, where the data is often overdispersed and zero-inflated, therefore making data analysis extremely challenging.

**Author(s)**

**Maintainer:** Carlotta Meyring <carlotta.meyring@uniklinik-freiburg.de> ([ORCID](#))

**See Also**

Useful links:

- <https://github.com/kreutz-lab/zitoools>
- Report bugs at <https://github.com/kreutz-lab/zitoools/issues>

\*                                      *Arithmetic Operators*

**Description**

Arithmetic operators for a Zi-class object

**Usage**

```
## S4 method for signature 'Zi,ANY'
e1 * e2
```

**Arguments**

e1                      [Zi-class object](#), matrix or number  
e2                      [Zi-class object](#), matrix or number

**Value**

a [Zi-class object](#) after a specific arithmetic operation is performed

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
Zi*Zi
Zi*2
```

+

*Arithmetic Operators***Description**

Arithmetic operators for a Zi-class object

Arithmetic operators for a Zi-class object

**Usage**

```
## S4 method for signature 'Zi,ANY'
e1 + e2
```

```
## S4 method for signature 'Zi,ANY'
e1 - e2
```

**Arguments**

e1 [Zi-class object](#), matrix or number

e2 [Zi-class object](#), matrix or number

**Value**

a [Zi-class object](#) after a specific arithmetic operation is performed

a [Zi-class object](#) after a specific arithmetic operation is performed

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
Zi+Zi
Zi+2
data(mtx)
Zi <- ziMain(mtx)
Zi+Zi
Zi+2
```

---

/ *Arithmetic Operators*


---

**Description**

Arithmetic operators for a Zi-class object

**Usage**

```
## S4 method for signature 'Zi,ANY'
e1 / e2
```

**Arguments**

e1                    [Zi-class object](#), matrix or number  
e2                    [Zi-class object](#), matrix or number

**Value**

a [Zi-class object](#) after a specific arithmetic operation is performed

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
Zi/Zi
Zi/2
```

---

assays                    *Access assays*


---

**Description**

access [assays](#) of an [Zi-class object](#) if the inputdata is an object of the class SummarizedExperiment

**Usage**

```
## S4 method for signature 'Zi'
assays(x, withDimnames = TRUE, ...)
```

**Arguments**

x                    [Zi-class object](#)  
withDimnames    A logical, indicating whether the dimnames of the SummarizedExperiment object should be applied (i.e. copied) to the extracted assays. see [assays](#)  
...                see [assays](#)

**Value**

list

**Examples**

```
data(mtx)
colData <- data.frame(SampleID = c('Sample1', 'Sample2', 'Sample3',
  'Sample4', 'Sample5', 'Sample6', 'Sample7', 'Sample8', 'Sample9',
  'Sample10'),
  Group = factor(x = c(1,1,1,1,1,2,2,2,2,2)))
rowData <- data.frame(Kingdom = c(rep('Bacteria', times = 100)),
  Phylum = c(rep('Bacteroidetes', times = 50),
  rep('Firmicutes', times = 50)))
se <- SummarizedExperiment::SummarizedExperiment(assays = list(counts = mtx),
  colData = colData, rowData = rowData)
Zi <- ziMain(se)
assays(Zi)
```

---

boxplot

*Create boxplots of a 'Zi'-class object*

---

**Description**

Create boxplots of a 'Zi'-class object.

**Usage**

```
## S3 method for class 'Zi'
boxplot(x, ...)
```

**Arguments**

x	'Zi'-class object
...	see <a href="#">boxplot.default</a>

**Value**

A List with all information to create a boxplot see [boxplot.default](#)

**See Also**

[boxplot.default](#)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
boxplot(Zi)
boxplot(log1p(Zi))
```

---

colData	<i>Access the col Data</i>
---------	----------------------------

---

### Description

access the `colData` of an `Zi`-class object if the input data is an object of the class `SummarizedExperiment`

### Usage

```
## S4 method for signature 'Zi'
colData(x, ...)
```

### Arguments

<code>x</code>	<code>Zi</code> -class object
<code>...</code>	<code>colData</code>

### Value

DFrame

### See Also

[colData](#)

### Examples

```
data(mtx)
colData <- data.frame(SampleID = c('Sample1', 'Sample2', 'Sample3', 'Sample4',
  'Sample5', 'Sample6', 'Sample7', 'Sample8', 'Sample9', 'Sample10'),
  Group = factor(x = c(1,1,1,1,1,2,2,2,2,2)))
rowData <- data.frame(Kingdom = c(rep('Bacteria', times = 100)),
  Phylum = c(rep('Bacteroidetes', times = 50),
  rep('Firmicutes', times = 50)))
se <- SummarizedExperiment::SummarizedExperiment(assays = list(counts = mtx),
  colData = colData, rowData = rowData)
Zi <- ziMain(se)
colData(Zi)
```

---

colMeans2	<i>Calculate the row or column means of zero-inflated count data</i>
-----------	--

---

### Description

Calculate row and column means of zero-inflated count data taking weights for structural zeros into account.

**Usage**

```
## S4 method for signature 'Zi'
colMeans2(x, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)

## S4 method for signature 'Zi'
rowMeans2(x, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)
```

**Arguments**

x	A <a href="#">Zi</a> -class object
rows, cols	A <a href="#">vector</a> indicating the subset of rows and/or columns to operate over. If <a href="#">NULL</a> (default), no subsetting is done
na.rm	<a href="#">logical</a> If <a href="#">TRUE</a> NAs are excluded, otherwise not. default = <a href="#">FALSE</a>
useNames	<a href="#">logical</a> If <a href="#">TRUE</a> (default), names attributes of result are set. Else if <a href="#">FALSE</a> , no naming support is done.

**Value**

a numeric [vector](#) of row/column length

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
colMeans2(Zi)
rowMeans2(Zi)
```

---

colMedians	<i>Calculate the row or column median of zero-deinflated count data</i>
------------	---

---

**Description**

Calculate the row or column median of zero-deinflated data of a [Zi](#)-class object. To calculate the median, the `deinflatedcounts` matrix will be extracted.

**Usage**

```
## S4 method for signature 'Zi'
colMedians(x, rows = NULL, cols = NULL, na.rm = TRUE, ..., useNames = TRUE)

## S4 method for signature 'Zi'
rowMedians(x, rows = NULL, cols = NULL, na.rm = TRUE, ..., useNames = TRUE)
```

**Arguments**

x	<a href="#">Zi</a> -class object
rows, cols	A <a href="#">vector</a> indicating the subset of rows (and/or columns) to operate over. If <a href="#">NULL</a> , no subsetting is done
na.rm	<a href="#">logical</a> If <a href="#">TRUE</a> NAs are excluded, otherwise not. default = <a href="#">TRUE</a>
...	see <a href="#">colMedians</a>
useNames	<a href="#">logical</a> . If <a href="#">TRUE</a> (default), names attributes of result are set. Else if <a href="#">FALSE</a> , no naming support is done.

**Value**

returns a numeric vector of row/column length

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
colMedians(Zi, useNames = TRUE)
rowMedians(Zi, useNames = TRUE)
```

---

cor

*Calculate weighted Pearson Correlation coefficients*

---

**Description**

calculate the weighted pearson correlation coefficients of a count matrix of an Zi object taking weights for zero counts into account

**Usage**

```
## S4 method for signature 'Zi,ANY'
cor(x, y = NULL, use = "everything", method = "pearson")
```

**Arguments**

x	'Zi'-class object
y	'Zi'-class object
use	'everything' see <a href="#">cor</a>
method	default = 'pearson', weighted correlation only implemented for person correlation

**Value**

correlation matrix

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
cor(Zi)
```

---

cov	<i>Calculate weighted Covariance</i>
-----	--------------------------------------

---

**Description**

calculate the weighted covariance of the columns of the count matrix of an Zi object taking weights for possible structural zero counts into account

**Usage**

```
## S4 method for signature 'Zi,ANY'
cov(x, y = NULL, use = "everything")
```

**Arguments**

x	'Zi'-class object
y	'Zi'-class object
use	'everything'

**Value**

covariance matrix

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
cov(Zi)
```

---

deinflatedcounts	<i>Access the model</i>
------------------	-------------------------

---

**Description**

access the `deinflatedcounts` of an Zi-class object

**Usage**

```
deinflatedcounts(x)

## S4 method for signature 'Zi'
deinflatedcounts(x)

deinflatedcounts(x) <- value

## S4 replacement method for signature 'Zi'
deinflatedcounts(x) <- value
```

**Arguments**

x                    [Zi](#)-class object  
value                deinflatedcounts object

**Value**

deinflatedcounts

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
deinflatedcounts(Zi)
```

---

heatmap	<i>Draw a Heat Map</i>
---------	------------------------

---

**Description**

draw a heatmap of a given 'Zi'-class object, heatmap.Zi uses the deinflatedcounts matrix (drawn structural zeros) to produce a heatmap. NA values are white

**Usage**

```
## S3 method for class 'Zi'
heatmap(x, ...)
```

**Arguments**

x                    'Zi'-class object  
...                  see [heatmap](#)

**Value**

heatmap

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
#heatmap(Zi) # Error, clustering not possible
heatmap(Zi, Rowv=NA) # no clustering of rows
heatmap(Zi, Rowv=NA, Colv=NA) # no clustering of rows and cols
```

---

inputcounts	<i>Access the inputcounts</i>
-------------	-------------------------------

---

**Description**

access the inputcounts of an [Zi](#)-class object

**Usage**

```
inputcounts(x)

## S4 method for signature 'Zi'
inputcounts(x)

inputcounts(x) <- value

## S4 replacement method for signature 'Zi'
inputcounts(x) <- value
```

**Arguments**

x	<a href="#">Zi</a> -class object
value	inputcounts object

**Value**

inputcounts

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
inputcounts(Zi)
```

---

inputdata	<i>Access and Set the inputdata</i>
-----------	-------------------------------------

---

**Description**

access the inputdata of an [Zi](#)-class object

**Usage**

```
inputdata(x)

## S4 method for signature 'Zi'
inputdata(x)

inputdata(x) <- value

## S4 replacement method for signature 'Zi'
inputdata(x) <- value
```

**Arguments**

x	Zi-class object
value	inputdata object

**Value**

inputdata

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
inputdata(Zi)
```

---

log1p	$\log(1+x)$
-------	-------------

---

**Description**

Calculate  $\log(1+x)$  of all 'matrix' objects of a 'Zi'-class object, log calculates by default natural logarithms

**Usage**

```
## S4 method for signature 'Zi'
log1p(x)
```

**Arguments**

x	Zi-class object
---	-----------------

**Value**

a Zi-class object where the  $\log(1+x)$  values of inputcounts, deinflatedcounts and weights are calculated.

**See Also**

[log1p](#), [log2p](#)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
log1p(Zi)
```

---

log2p	<i>log2p(x+1)</i>
-------	-------------------

---

**Description**

Calculate  $\log_2(x+1)$  of all 'matrix' objects of a 'Zi'-class object

**Usage**

```
log2p(x)
```

**Arguments**

x                    [Zi-class object](#)

**Value**

a [Zi-class object](#) where the  $\log_2(1+x)$  values of inputcounts, deinflatedcounts and weights are calculated.

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
log2p(Zi)
```

---

mean	<i>Arithmetic Mean</i>
------	------------------------

---

**Description**

Calculate the arithmetic mean of zero inflated data taking weights for structural zeros into account

**Usage**

```
## S3 method for class 'Zi'
mean(x, ...)
```

**Arguments**

x                    A [Zi-class object](#)  
 ...                    [mean.default](#)

**Value**

mean value

**See Also**

[weighted.mean](#), [colMeans2](#), [rowMeans2](#)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
mean(Zi)
```

---

median

*Calculate the median of zero-deinflated count data*

---

**Description**

Calculate the median of zero-deinflated data of a 'Zi'-class object. To calculate the median, the deinflatedcounts matrix will be extracted

**Usage**

```
## S3 method for class 'Zi'
median(x, na.rm = TRUE, ...)
```

**Arguments**

`x` [Zi-class object](#)

`na.rm` [logical](#) If `TRUE` NAs are excluded, otherwise not. default = `TRUE`

`...` see [median.default](#)

**Value**

median value

**See Also**

[median](#), [colMedians](#), [rowMedians](#)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
median(Zi)
```

---

MissingValueHeatmap    *Missing Value Heatmap*

---

**Description**

Missing Value Heatmap

**Usage**

```
MissingValueHeatmap(ZiObject, title = "", xlab = "", ylab = "")
```

**Arguments**

ZiObject	ZiObject, result of the ziMain function
title	Title of the plot .
xlab	Title of the x axis.
ylab	Title of the y axis.

**Value**

heatmap

**Examples**

```
data(mtx)
```

---

model                    *Access the model*

---

**Description**

access the model of an [Zi](#)-class object

**Usage**

```
model(x)

## S4 method for signature 'Zi'
model(x)

model(x) <- value

## S4 replacement method for signature 'Zi'
model(x) <- value
```

**Arguments**

x	<a href="#">Zi</a> -class object
value	model object

**Value**

model

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
model(Zi)
```

---

mtx

*Matrix Data*

---

**Description**

Zero-inflated matrix data

**Usage**

mtx

**Format**

**'mtx':**

A matrix with 100 rows and 10 columns

**Value**

a data matrix

---

otu\_table

*Access the otu table*

---

**Description**

access the [otu\\_table](#) of an [Zi](#)-class object if the inputdata slot is a phyloseq object

**Usage**

```
## S4 method for signature 'Zi'
otu_table(object)
```

**Arguments**

object            [Zi](#)-class object

**Value**

otu\_table

**Examples**

```

data(mtx)
OTU <- otu_table(mtx, taxa_are_rows = TRUE)
sample_data <- data.frame(SampleID = c('Sample1', 'Sample2', 'Sample3',
  'Sample4', 'Sample5', 'Sample6', 'Sample7', 'Sample8', 'Sample9',
  'Sample10'),
  Group = factor(x = c(1,1,1,1,1,2,2,2,2,2)))
SAM <- sample_data(sample_data)
tax_table <- data.frame(Kingdom = c(rep('Bacteria', times = 100)),
  Phylum = c(rep('Bacteroidetes', times = 50),
  rep('Firmicutes', times = 50)))
TAX <- tax_table(tax_table)
ps <- phyloseq::phyloseq(OTU, TAX, SAM)
Zi <- ziMain(ps)
otu_table(Zi)

```

---

plot

*Plotting*


---

**Description**

plot

**Usage**

```

## S4 method for signature 'Zi,ANY'
plot(x, y, ...)

```

**Arguments**

x	Zi-class object
y	the y coordinates of points in the plot, optional if x is an appropriate structure
...	Arguments to be passed to plot

**Value**

returns plot object

**Examples**

```

data(mtx)
Zi <- ziMain(mtx)
plot(Zi)

```

---

quantile	<i>Calculate the quantiles of zero-deinflated count data</i>
----------	--

---

**Description**

Calculate the quantiles of zero-deinflated data of a [Zi](#)-class object. To calculate the quantiles, the `deinflatedcounts` matrix will be extracted.

**Usage**

```
## S3 method for class 'Zi'
quantile(x, probs = seq(0, 1, 0.25), na.rm = TRUE, ...)
```

**Arguments**

<code>x</code>	A <a href="#">Zi</a> -class object
<code>probs</code>	A numeric <a href="#">vector</a> of J probabilities in [0,1]
<code>na.rm</code>	<a href="#">logical</a> If <a href="#">TRUE</a> NAs are excluded, otherwise not. default = <a href="#">TRUE</a>
<code>...</code>	<a href="#">quantile</a>

**Value**

quantile value

**See Also**

[quantile](#), [rowQuantiles](#), [colQuantiles](#)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
quantile(Zi)
```

---

<code>resample_deinflatedcounts</code>	<i>Resample a <a href="#">Zi</a>-class object</i>
--	---

---

**Description**

Resample the `deinflatedcounts` matrix of an [Zi](#)-class object. Resampling is done by drawing from a binomial distribution with a given probability that a count value (zero and non-zero) is a structural zero.

**Usage**

```
resample_deinflatedcounts(x)
```

**Arguments**

x [Zi-class object](#)

**Value**

a [Zi-class object](#) where the `deinflatedcounts` are resampled

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
resample_deinflatedcounts(Zi)
```

---

rowData	<i>Access the row data</i>
---------	----------------------------

---

**Description**

access the [rowData](#) of an [Zi-class object](#) if the inputdata is an object of the class `SummarizedExperiment`

**Usage**

```
## S4 method for signature 'Zi'
rowData(x, useNames = TRUE, ...)
```

**Arguments**

x [Zi-class object](#)  
 useNames returns a `rowData` dataframe with rownames  
 ... [rowData](#)

**Value**

DFrame

**Examples**

```
data(mtx)
colData <- data.frame(SampleID = c('Sample1', 'Sample2', 'Sample3', 'Sample4',
  'Sample5', 'Sample6', 'Sample7', 'Sample8', 'Sample9', 'Sample10'),
  Group = factor(x = c(1,1,1,1,1,2,2,2,2,2)))
rowData <- data.frame(Kingdom = c(rep('Bacteria', times = 100)),
  Phylum = c(rep('Bacteroidetes', times = 50),
  rep('Firmicutes', times = 50)))
se <- SummarizedExperiment::SummarizedExperiment(assays = list(counts = mtx),
  colData = colData, rowData = rowData)
Zi <- ziMain(se)
rowData(Zi)
```

---

rowQuantiles	<i>Calculate the row or column quantiles of zero-deinflated count data</i>
--------------	--

---

### Description

Calculate the row or column quantiles of zero-deinflated data of a [Zi](#)-class object. To calculate the quantiles, the deinflatedcounts matrix will be extracted

### Usage

```
## S4 method for signature 'Zi'
rowQuantiles(
  x,
  rows = NULL,
  cols = NULL,
  probs = seq(from = 0, to = 1, by = 0.25),
  na.rm = TRUE,
  type = 7L,
  ...,
  useNames = TRUE,
  drop = TRUE
)

## S4 method for signature 'Zi'
colQuantiles(
  x,
  rows = NULL,
  cols = NULL,
  probs = seq(from = 0, to = 1, by = 0.25),
  na.rm = TRUE,
  type = 7L,
  ...,
  useNames = TRUE,
  drop = TRUE
)
```

### Arguments

x	A <a href="#">Zi</a> -class object
rows, cols	A <a href="#">vector</a> indicating the subset of rows and/or columns to operate over. If <a href="#">NULL</a> (default), no subsetting is done.
probs	A numeric <a href="#">vector</a> of J probabilities in [0,1]
na.rm	<a href="#">logical</a> If <a href="#">TRUE</a> NAs are excluded, otherwise not. default = <a href="#">TRUE</a>
type	An integer specifying the type of estimator
...	Additional arguments passed to specific methods <a href="#">rowQuantiles</a>
useNames	<a href="#">logical</a> If <a href="#">TRUE</a> (default), names attributes of result are set. Else if <a href="#">FALSE</a> , no naming support is done.
drop	If <a href="#">TRUE</a> a <a href="#">vector</a> is returned if J == 1.

**Value**

a numeric [vector](#) of row/column length

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
rowQuantiles(Zi, useNames = TRUE)
colQuantiles(Zi, useNames = TRUE)
```

---

rowSds

*Row and Column Standard Deviations of zero inflated count data*


---

**Description**

Calculate row and column standard deviations of zero inflated count data taking weights for structural zeros into account

**Usage**

```
## S4 method for signature 'Zi'
rowSds(x, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)

## S4 method for signature 'Zi'
colSds(x, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)
```

**Arguments**

x	A <a href="#">Zi</a> -class object
rows, cols	A <a href="#">vector</a> indicating the subset of rows and/or columns to operate over. If <a href="#">NULL</a> (default), no subsetting is done
na.rm	<a href="#">logical</a> If <a href="#">TRUE</a> NAs are excluded, otherwise not. default = <a href="#">FALSE</a>
useNames	<a href="#">logical</a> If <a href="#">TRUE</a> (default), names attributes of result are set. Else if <a href="#">FALSE</a> , no naming support is done.

**Value**

a [vector](#) of row/column length

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
rowSds(Zi)
colSds(Zi)
```

---

rowVars	<i>Row and Column Variances of zero inflated count data</i>
---------	---

---

### Description

Calculate row and column variances of zero inflated count data taking weights for structural zeros into account.

### Usage

```
## S4 method for signature 'Zi'
rowVars(x, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)

## S4 method for signature 'Zi'
colVars(x, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)
```

### Arguments

x	A <a href="#">Zi</a> -class object
rows, cols	A <a href="#">vector</a> indicating the subset of rows and/or columns to operate over. If <a href="#">NULL</a> (default), no subsetting is done
na.rm	<a href="#">logical</a> If <a href="#">TRUE</a> NAs are excluded, otherwise not. default = <a href="#">FALSE</a>
useNames	<a href="#">logical</a> If <a href="#">TRUE</a> (default), names attributes of result are set. Else if <a href="#">FALSE</a> , no naming support is done.

### Value

a vector of row/col length

### Examples

```
data(mtx)
Zi <- ziMain(mtx)
rowVars(Zi)
colVars(Zi)
```

---

rowWeightedMeans	<i>Row and Column weighted means of zero inflated count data</i>
------------------	--

---

### Description

Calculate row and column weighted means of zero inflated count data, additionally taking weights for structural zeros into account.

**Usage**

```
## S4 method for signature 'Zi'
rowWeightedMeans(
  x,
  w,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  useNames = TRUE
)

## S4 method for signature 'Zi'
colWeightedMeans(
  x,
  w,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  useNames = TRUE
)
```

**Arguments**

x	A <a href="#">Zi</a> -class object
w	a numerical vector of weights either of length = rows or length = cols giving the weights to use for elements of x
rows, cols	A <a href="#">vector</a> indicating the subset of rows and/or columns to operate over. If <a href="#">NULL</a> (default), no subsetting is done
na.rm	<a href="#">logical</a> If <a href="#">TRUE</a> NAs are excluded, otherwise not. default = <a href="#">FALSE</a>
useNames	<a href="#">logical</a> If <a href="#">TRUE</a> (default), names attributes of result are set. Else if <a href="#">FALSE</a> , no naming support is done.

**Value**

a numeric vector of length N(K)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
rowWeightedMeans(Zi, w = runif(ncol(inputcounts(Zi)), 0.1,1))
colWeightedMeans(Zi, w = runif(nrow(inputcounts(Zi)), 0.1,1))
```

---

rowWeightedSds	<i>Row and column weighted standard deviations or variances of zero inflated count data</i>
----------------	---

---

**Description**

Calculate row and column standard deviations or variances of zero inflated count data, additionally taking weights for structural zeros into account.

**Usage**

```
## S4 method for signature 'Zi'
rowWeightedSds(x, w, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)

## S4 method for signature 'Zi'
colWeightedSds(x, w, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)

## S4 method for signature 'Zi'
rowWeightedVars(x, w, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)

## S4 method for signature 'Zi'
colWeightedVars(x, w, rows = NULL, cols = NULL, na.rm = FALSE, useNames = TRUE)
```

**Arguments**

x	A <a href="#">Zi</a> -class object
w	a numerical vector of weights either of length = rows or length = cols giving the weights to use for elements of x
rows, cols	A <a href="#">vector</a> indicating the subset of rows and/or columns to operate over. If <a href="#">NULL</a> (default), no subsetting is done
na.rm	<a href="#">logical</a> If <a href="#">TRUE</a> NAs are excluded, otherwise not. default = <a href="#">FALSE</a>
useNames	<a href="#">logical</a> If <a href="#">TRUE</a> (default), names attributes of result are set. Else if <a href="#">FALSE</a> , no naming support is done.

**Value**

a numeric vector of length N(K)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
rowWeightedSds(Zi, w = runif(ncol(inputcounts(Zi)), 0.1,1))
colWeightedSds(Zi, w = runif(nrow(inputcounts(Zi)), 0.1,1))
rowWeightedVars(Zi, w = runif(ncol(inputcounts(Zi)), 0.1,1))
colWeightedVars(Zi, w = runif(nrow(inputcounts(Zi)), 0.1,1))
```

---

sample\_data

*Access the sample data*

---

**Description**

access the [sample\\_data](#) of an [Zi](#)-class object if the inputdata slot is a phyloseq object

**Usage**

```
## S4 method for signature 'Zi'
sample_data(object)
```

**Arguments**

object            [Zi-class object](#)

**Value**

sample\_data

**Examples**

```
data(mtx)
OTU <- otu_table(mtx, taxa_are_rows = TRUE)
sample_data <- data.frame(SampleID = c('Sample1', 'Sample2', 'Sample3',
  'Sample4', 'Sample5', 'Sample6', 'Sample7', 'Sample8', 'Sample9',
  'Sample10'),
  Group = factor(x = c(1,1,1,1,1,1,2,2,2,2)))
SAM <- sample_data(sample_data)
tax_table <- data.frame(Kingdom = c(rep('Bacteria', times = 100)),
  Phylum = c(rep('Bacteroidetes', times = 50),
  rep('Firmicutes', times = 50)))
TAX <- tax_table(tax_table)
ps <- phyloseq::phyloseq(OTU, TAX, SAM)
Zi <- ziMain(ps)
sample_data(Zi)
```

---

sd

*Standard Deviation of zero inflated count data*

---

**Description**

Calculate the standard deviation of zero inflated count data taking weights for structural zeros into account.

**Usage**

```
## S4 method for signature 'Zi'
sd(x, na.rm = FALSE)
```

**Arguments**

x                    A [Zi-class object](#)

na.rm                [logical](#) If TRUE NAs are excluded, otherwise not. default = [FALSE](#)

**Value**

standard deviation value

**See Also**

[weightedSd](#), [rowSds](#), [colSds](#)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
sd(Zi)
```

---

show

*Show summary of Zi object*

---

**Description**

Message printed at command line

**Usage**

```
## S4 method for signature 'Zi'
show(object)
```

**Arguments**

object            [Zi-class object](#)

**Value**

returns a numeric vector of row/column length

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
Zi
show(Zi)
```

---

subset\_feature

*Subset a [Zi-class object](#) based on feature data*

---

**Description**

Subset a [Zi-class object](#) based on tax\_table of a phyloseq object or on rowData of a Summarized-Experiment object

**Usage**

```
subset_feature(Zi, ...)
```

**Arguments**

`Zi` [Zi-class object](#)  
 ... The subsetting expression that should be applied, see [subset](#) for more details

**Value**

a [Zi-class object](#) after subsetting is done

**Examples**

```
data(mtx)
OTU <- otu_table(mtx, taxa_are_rows = TRUE)
sample_data <- data.frame(SampleID = c('Sample1', 'Sample2', 'Sample3',
  'Sample4', 'Sample5', 'Sample6', 'Sample7', 'Sample8', 'Sample9',
  'Sample10'),
  Group = factor(x = c(1,1,1,1,1,1,2,2,2,2,2)))
SAM <- sample_data(sample_data)
tax_table <- data.frame(Kingdom = c(rep('Bacteria', times = 100)),
  Phylum = c(rep('Bacteroidetes', times = 50),
  rep('Firmicutes', times = 50)))
TAX <- tax_table(tax_table)
ps <- phyloseq::phyloseq(OTU, TAX, SAM)
Zi <- ziMain(ps)
subset_Zi_phylo <- subset_feature(Zi, ta2 == 'Bacteroidetes')
```

---

subset_sample	<i>Subset a <a href="#">Zi-class object</a> based on sample data</i>
---------------	--

---

**Description**

Subset a [Zi-class object](#) based on `sample_data` of an `phyloseq` object or on `colData` based on a `SummarizedExperiment` object

**Usage**

```
subset_sample(Zi, ...)
```

**Arguments**

`Zi` [Zi-class object](#)  
 ... The subsetting expression that should be applied, see [subset](#) for more details

**Value**

a [Zi-class object](#) after subsetting is done

**Examples**

```

data(mtx)
OTU <- otu_table(mtx, taxa_are_rows = TRUE)
sample_data <- data.frame(SampleID = c('Sample1', 'Sample2', 'Sample3',
  'Sample4', 'Sample5', 'Sample6', 'Sample7', 'Sample8', 'Sample9',
  'Sample10'),
  Group = factor(x = c(1,1,1,1,1,1,2,2,2,2,2)))
SAM <- sample_data(sample_data)
tax_table <- data.frame(Kingdom = c(rep('Bacteria', times = 100)),
  Phylum = c(rep('Bacteroidetes', times = 50),
  rep('Firmicutes', times = 50)))
TAX <- tax_table(tax_table)
ps <- phyloseq::phyloseq(OTU, TAX, SAM)
Zi <- ziMain(ps)
subset_Zi <- subset_sample(Zi, SampleID %in% c('Sample1','Sample2'))

```

t

*Transpose a Zi-class object***Description**

transpose all matrices of a [Zi-class object](#)

**Usage**

```

## S4 method for signature 'Zi'
t(x)

```

**Arguments**

x [Zi-class object](#)

**Value**

[Zi-class object](#)

**Examples**

```

data(mtx)
Zi <- ziMain(mtx)
t(Zi)

```

---

tax_table	<i>Access the taxonomy table</i>
-----------	----------------------------------

---

**Description**

access the taxonomy table ([tax\\_table](#)) of an [Zi](#)-class object if the inputdata slot is a phyloseq object

**Usage**

```
## S4 method for signature 'Zi'
tax_table(object)
```

**Arguments**

object            [Zi](#)-class object

**Value**

tax\_table

**Examples**

```
data(mtx)
OTU <- otu_table(mtx, taxa_are_rows = TRUE)
sample_data <- data.frame(SampleID = c('Sample1', 'Sample2', 'Sample3',
  'Sample4', 'Sample5', 'Sample6', 'Sample7', 'Sample8', 'Sample9',
  'Sample10'),
  Group = factor(x = c(1,1,1,1,1,1,2,2,2,2)))
SAM <- sample_data(sample_data)
tax_table <- data.frame(Kingdom = c(rep('Bacteria', times = 100)),
  Phylum = c(rep('Bacteroidetes', times = 50),
  rep('Firmicutes', times = 50)))
TAX <- tax_table(tax_table)
ps <- phyloseq::phyloseq(OTU, TAX, SAM)
Zi <- ziMain(ps)
tax_table(Zi)
```

---

var	<i>Variance of zero inflated count data</i>
-----	---

---

**Description**

Calculate the variance of zero inflated count data taking weights for structural zeros into account.

**Usage**

```
## S4 method for signature 'Zi,ANY'
var(x, na.rm = FALSE)
```

**Arguments**

x                    A [Zi](#)-class object  
na.rm                [logical](#) If [TRUE](#) NAs are excluded, otherwise not. default = [FALSE](#)

**Value**

variance value

**See Also**

[weightedVar](#), [rowVars](#), [colVars](#)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
var(Zi)
```

---

weighted.mean

*Weighted Arithmetic Mean of zero inflated count data*

---

**Description**

Calculate a weighted mean of zero inflated count data, additionally taking weights for structural zeros into account

**Usage**

```
## S4 method for signature 'Zi'
weighted.mean(x, w, ...)
```

**Arguments**

x                    A [Zi](#)-class object  
w                    a numerical [vector](#) of weight the same length as x giving the weights to use for elements of x  
...                  [weighted.mean](#)

**Value**

weighted mean value

**See Also**

[weighted.mean](#), [rowWeightedMeans](#), [colWeightedMeans](#)

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
weight <- runif(length(inputcounts(Zi)), 0.1, 1)
weighted.mean(Zi, w= weight)
```

---

`weightedSd`*Weighted Variance and weighted Standard Deviation*

---

### Description

Calculate a weighted variance and standard deviation of zero inflated count data, additionally taking weights for structural zeros into account

### Usage

```
weightedSd(x, w = NULL, idxs = NULL, na.rm = FALSE, center = NULL, ...)
```

```
weightedVar(x, w = NULL, idxs = NULL, na.rm = FALSE, center = NULL, ...)
```

### Arguments

<code>x</code>	A <a href="#">Zi</a> -class object
<code>w</code>	a numerical <a href="#">vector</a> of weight the same length as <code>x</code> giving the weights to use for elements of <code>x</code>
<code>idxs</code>	A <a href="#">vector</a> indicating subset of elements to operate over. If <code>NULL</code> , no subsetting is done.
<code>na.rm</code>	<a href="#">logical</a> If <code>TRUE</code> NAs are excluded, otherwise not. default = <code>FALSE</code>
<code>center</code>	<a href="#">numeric</a> scalar specifying the center location of the data. If <code>NULL</code> , it is estimated from data.
<code>...</code>	<a href="#">weightedVar</a>

### Value

a [numeric](#) scalar

### See Also

[weightedVar](#), [rowWeightedVars](#), [colWeightedVars](#)

### Examples

```
data(mtx)
Zi <- ziMain(mtx)
weight <- runif(length(inputcounts(Zi)), 0.1, 1)
weightedVar(Zi, w= weight)
weightedSd(Zi, w = weight)
```

---

weights	<i>Access the weights</i>
---------	---------------------------

---

**Description**

access the weights of an [Zi](#)-class object

**Usage**

```
weights(x)

## S4 method for signature 'Zi'
weights(x)

weights(x) <- value

## S4 replacement method for signature 'Zi'
weights(x) <- value
```

**Arguments**

x	<a href="#">Zi</a> -class object
value	weights object

**Value**

weights

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
weights(Zi)
```

---

Zi-class	<i>Class Zi</i>
----------	-----------------

---

**Description**

Objects of this class store all the results of the `ZiMain` function to continue zero inflated data analysis

**Value**

[Zi](#)-class object

**Slots**

`inputdata` a matrix, phyloseq or SummarizedExperiment object.  
`inputcounts` matrix. The count matrix, features as rows, samples as columns  
`model` list. The result of fitting a zero inflated model using [zeroinfl](#)  
`deinflatedcounts` matrix. The matrix where predicted structural zeros are omitted and stored as NA values  
`weights` matrix. A matrix containing weights for zero counts

---

 zi2deseq2

---

*Convert a Zi-class object to a DESeq2 dds object*


---

**Description**

A Zi-class object is converted to a DESeqDataSet object, which can be used for DESeq2 analysis. Both, weight and count matrices will be stored in assays of the DESeqDataSet.

**Usage**

```
zi2deseq2(ZiObject, design, colData, ...)
```

**Arguments**

<code>ZiObject</code>	Zi-class object
<code>design</code>	A formula which specifies the design of the experiment, taking the form <code>formula(~ x + y + z)</code> . That is, a formula with right-hand side only. By default, the functions in this package and DESeq2 will use the last variable in the formula (e.g. <code>z</code> ) for presenting results (fold changes, etc.) and plotting. When considering your specification of experimental design, you will want to re-order the levels so that the NULL set is first.
<code>colData</code>	if the <code>inputdata</code> of the Zi-class object is a matrix: a <code>DataFrame</code> or <code>data.frame</code> with at least a single column. Rows of <code>colData</code> correspond to columns of count-Data
<code>...</code>	<a href="#">phyloseq::phyloseq_to_deseq2</a> if the <code>inputdata</code> of the 'Zi'-object is a phyloseq object <a href="#">DESeq2::DESeqDataSet</a> if the <code>inputdata</code> the 'Zi'-object is a SummarizedExperiment object

**Value**

a dds class object

**Examples**

```
data(mtx)
Zi <- ziMain(mtx)
colData <- data.frame(group = factor(x = c(1,1,1,1,1,2,2,2,2,2)))
zi2deseq2(Zi, ~group, colData)
```

---

zi2phyloseq	<i>Replace the otu table of a phyloseq object</i>
-------------	---

---

**Description**

Replace the OTU table of a phyloseq object with the OTU table of zero de-inflated count data

**Usage**

```
zi2phyloseq(ZiObject)
```

**Arguments**

ZiObject      [Zi](#)-class object with a phyloseq object as input

**Value**

a 'phyloseq'-class object

**Examples**

```
data(mtx)
OTU <- otu_table(mtx, taxa_are_rows = TRUE)
sample_data <- data.frame(SampleID = c('Sample1', 'Sample2', 'Sample3',
  'Sample4', 'Sample5', 'Sample6', 'Sample7', 'Sample8', 'Sample9',
  'Sample10'),
  Group = factor(x = c(1,1,1,1,1,2,2,2,2,2)))
SAM <- sample_data(sample_data)
tax_table <- data.frame(Kingdom = c(rep('Bacteria', times = 100)),
  Phylum = c(rep('Bacteroidetes', times = 50),
  rep('Firmicutes', times = 50)))
TAX <- tax_table(tax_table)
ps <- phyloseq::phyloseq(OTU, TAX, SAM)
Zi <- ziMain(ps)
new_ps <- zi2phyloseq(Zi)
new_ps
```

---

ziMain	<i>ziMain - main function to fit a zero inflation model and calculate weights for structural zeros</i>
--------	--

---

**Description**

This function fits a zero-inflated mixture model (either Poisson or negative binomial distribution) to count data and calculates weights for all zeros indicating whether a zero is a real count (weight close to 1) or whether it is a structural zero (weight close to 0). The default model is a zero inflated negative binomial model.

The input inputdata of the ziMain function is either a phyloseq object, SummarizedExperiment object or count matrix.

In order to reduce calculation times, the count matrix is divided into blocks of around 5000 count values. Then, a zero inflation model (either Poisson or negative binomial distribution) is fitted to the data. The response variable count is estimated using the predictor variables sample(columns) and feature(rows). Using the fitted zero inflated model, probabilities given that a zero in the count matrix is a structural zero are predicted. Those probabilities are used in two ways: 1) A zero-deinflated count matrix is generated where a appropriate proportion of zeros are randomly replaced by NA. This count matrix can be used for analysis methods which cannot deal with weights. 2) Weights

$$w = \frac{(1 - \pi) f_{\text{NB}}(y; \mu, \theta)}{f_{\text{ZINB}}(y; \mu, \theta, \pi)}.$$

(see Van den Berge, K., Perraudeau, F., Sonesson, C. et al.) are calculated in order to down-weight structural zeros in analyses which can account for weighting of individual data points.

all zero counts are calculated given the following formula:

The result of the ziMain function can be used to analyze zero inflated count data.

## Usage

```
ziMain(
  inputdata,
  feature = "feature",
  formula = count ~ sample + feature,
  dist = "negbin",
  link = "logit",
  zeroRows.rm = FALSE,
  ...
)
```

## Arguments

inputdata	phyloseq object, SummarizedExperiment object, or matrix (rows =features, columns=samples)
feature	'feature', 'gene', 'OTU', 'phylum', etc. By default, rownames are labelled as feature1, feature2, ...
formula	formula to fit the zero inflated model $y \sim x_1 + x_2 + \dots$ , default = count ~ sample + feature. A different set of regressors can be specified using $y \sim x_1 + x_2 + \dots   z_1 + z_2$ <ul style="list-style-type: none"> <li>• ... where the first part describes the count data model and the second part describes the zero inflation model</li> </ul>
dist	= distribution, either poisson ('poisson'), negative binomial ('negbin')
link	= link function, either 'logit', 'probit', 'cloglog', 'cauchit'
zeroRows.rm	= logical, indicating whether rows that only contain zeros should be removed (TRUE) or not (FALSE) (they are removed to fit a zero inflated model and will be added afterwards count matrix per default = 0 and weights = 1)
...	additional parameters to describe the model, see <a href="#">zeroinfl</a>

## Value

Zi-class object

**Slots**

`inputdata` a matrix, phyloseq or SummarizedExperiment object.  
`inputcounts` matrix. The count matrix, features as rows, samples as columns  
`model` list. The result of fitting a zero inflated model using [zeroinfl](#)  
`deinflatedcounts` matrix. A matrix where zero counts are randomly replaced according to the estimated probability of being a structural zero  
`weights` matrix. A matrix containing weights for zero counts

**References**

Van den Berge, K., Perraudeau, F., Soneson, C. et al. Observation weights unlock bulk RNA-seq tools for zero inflation and single-cell applications. *Genome Biol* 19, 24 (2018). <https://doi.org/10.1186/s13059-018-1406-4>

**See Also**

[zeroinfl](#)

**Examples**

```
# zero-inflated count matrix
data(mtx)
# calling ziMain function:
Zi <- ziMain(mtx)
#Example Data Sets from other R packages
#data(enterotype)
#data(GlobalPatterns)
#data(esophagus)
#ziMain(esophagus)
#data(soilrep)
```

# Index

- \* **datasets**
    - mtx, [17](#)
  - \* **internal**
    - zitoools-package, [3](#)
  - [\\*](#), [3](#)
  - [\\*](#), Zi, ANY-method ([\\*](#)), [3](#)
  - [+](#), [4](#)
  - [+](#), Zi, ANY-method ([+](#)), [4](#)
  - [-](#), Zi, ANY-method ([+](#)), [4](#)
  - [/](#), [5](#)
  - [/](#), Zi, ANY-method ([/](#)), [5](#)
- assays, [5, 5](#)  
assays, Zi-method (assays), [5](#)
- boxplot, [6](#)  
boxplot.default, [6](#)
- colData, [7, 7](#)  
colData, Zi-method (colData), [7](#)  
colMeans2, [7, 14](#)  
colMeans2, Zi-method (colMeans2), [7](#)  
colMedians, [8, 8, 15](#)  
colMedians, Zi-method (colMedians), [8](#)  
colQuantiles, [19](#)  
colQuantiles (rowQuantiles), [21](#)  
colQuantiles, Zi-method (rowQuantiles), [21](#)  
colSds, [27](#)  
colSds (rowSds), [22](#)  
colSds, Zi-method (rowSds), [22](#)  
colVars, [31](#)  
colVars (rowVars), [23](#)  
colVars, Zi-method (rowVars), [23](#)  
colWeightedMeans, [31](#)  
colWeightedMeans (rowWeightedMeans), [23](#)  
colWeightedMeans, Zi-method (rowWeightedMeans), [23](#)  
colWeightedSds (rowWeightedSds), [24](#)  
colWeightedSds, Zi-method (rowWeightedSds), [24](#)  
colWeightedVars, [32](#)  
colWeightedVars (rowWeightedSds), [24](#)
- colWeightedVars, Zi-method (rowWeightedSds), [24](#)
- cor, [9, 9](#)  
cor, Zi, ANY-method (cor), [9](#)  
cov, [10](#)  
cov, Zi, ANY-method (cov), [10](#)
- deinflatedcounts, [10](#)  
deinflatedcounts, Zi-method (deinflatedcounts), [10](#)  
deinflatedcounts<- (deinflatedcounts), [10](#)  
deinflatedcounts<- , Zi-method (deinflatedcounts), [10](#)  
DESeq2::DESeqDataSet, [34](#)
- FALSE, [8, 21–26, 31, 32, 36](#)
- heatmap, [11, 11](#)
- inputcounts, [12](#)  
inputcounts, Zi-method (inputcounts), [12](#)  
inputcounts<- (inputcounts), [12](#)  
inputcounts<- , Zi-method (inputcounts), [12](#)  
inputdata, [12](#)  
inputdata, Zi-method (inputdata), [12](#)  
inputdata<- (inputdata), [12](#)  
inputdata<- , Zi-method (inputdata), [12](#)
- log1p, [13, 13](#)  
log1p, Zi-method (log1p), [13](#)  
log2p, [13, 14](#)  
log2p, Zi-method (log2p), [14](#)  
logical, [8, 15, 19, 21–26, 31, 32](#)
- mean, [14](#)  
mean.default, [14](#)  
median, [15, 15](#)  
median.default, [15](#)  
MissingValueHeatmap, [16](#)  
model, [16](#)  
model, Zi-method (model), [16](#)  
model<- (model), [16](#)  
model<- , Zi-method (model), [16](#)

- mtx, 17
- NA, 8, 21–26, 31, 32
- NULL, 8, 21–25, 32
- numeric, 32
- otu\_table, 17, 17
- otu\_table, Zi-method (otu\_table), 17
- phyloseq::phyloseq\_to\_deseq2, 34
- plot, 18
- plot, Zi, ANY-method (plot), 18
- quantile, 19, 19
- resample\_deinflatedcounts, 19
- rowData, 20, 20
- rowData, Zi-method (rowData), 20
- rowMeans2, 14
- rowMeans2 (colMeans2), 7
- rowMeans2, Zi-method (colMeans2), 7
- rowMedians, 15
- rowMedians (colMedians), 8
- rowMedians, Zi-method (colMedians), 8
- rowQuantiles, 19, 21, 21
- rowQuantiles, Zi-method (rowQuantiles), 21
- rowSds, 22, 27
- rowSds, Zi-method (rowSds), 22
- rowVars, 23, 31
- rowVars, Zi-method (rowVars), 23
- rowWeightedMeans, 23, 31
- rowWeightedMeans, Zi-method (rowWeightedMeans), 23
- rowWeightedSds, 24
- rowWeightedSds, Zi-method (rowWeightedSds), 24
- rowWeightedVars, 32
- rowWeightedVars (rowWeightedSds), 24
- rowWeightedVars, Zi-method (rowWeightedSds), 24
- sample\_data, 25, 25
- sample\_data, Zi-method (sample\_data), 25
- sd, 26
- sd, Zi-method (sd), 26
- show, 27
- show, Zi-method (show), 27
- subset, 28
- subset\_feature, 27
- subset\_sample, 28
- t, 29
- t, Zi-method (t), 29
- tax\_table, 30, 30
- tax\_table, Zi-method (tax\_table), 30
- TRUE, 8, 15, 19, 21–26, 31, 32, 36
- var, 30
- var, Zi, ANY-method (var), 30
- vector, 8, 19, 21–25, 31, 32
- weighted.mean, 14, 31, 31
- weighted.mean, Zi-method (weighted.mean), 31
- weightedSd, 27, 32
- weightedSd, Zi-method (weightedSd), 32
- weightedVar, 31, 32
- weightedVar (weightedSd), 32
- weightedVar, Zi-method (weightedSd), 32
- weights, 33
- weights, Zi-method (weights), 33
- weights<- (weights), 33
- weights<- , Zi-method (weights), 33
- zeroinfl, 34, 36, 37
- Zi, 3–5, 7, 8, 10–36
- Zi-class, 33
- zi2deseq2, 34
- zi2phyloseq, 35
- ziMain, 35
- ziMain, phyloseq-method (ziMain), 35
- ziMain, SummarizedExperiment-method (ziMain), 35
- zitoools (zitoools-package), 3
- zitoools-package, 3
- ~` (+), 4