

# Package ‘MDTS’

July 18, 2025

**Title** Detection of de novo deletion in targeted sequencing trios

**Version** 1.29.0

**Description** A package for the detection of de novo copy number deletions in targeted sequencing of trios with high sensitivity and positive predictive value.

**Depends** R (>= 3.5.0)

**Imports** GenomicAlignments, GenomicRanges, IRanges, Biostrings,  
DNAcopy, Rsamtools, parallel, stringr

**Suggests** testthat, knitr

**VignetteBuilder** knitr

**License** Artistic-2.0

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**biocViews** StatisticalMethod, Technology, Sequencing,  
TargetedResequencing, Coverage, DataImport

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

**git\_branch** devel

**git\_last\_commit** bf8ac95

**git\_last\_commit\_date** 2025-04-15

**Repository** Bioconductor 3.22

**Date/Publication** 2025-07-18

**Author** Jack M.. Fu [aut, cre]

**Maintainer** Jack M.. Fu <jmfu@jhsph.edu>

## Contents

|                             |   |
|-----------------------------|---|
| MDTS-package . . . . .      | 2 |
| calcBins . . . . .          | 2 |
| calcCounts . . . . .        | 3 |
| calcMD . . . . .            | 3 |
| denovoDeletions . . . . .   | 4 |
| getMetaData . . . . .       | 5 |
| normalizeCounts . . . . .   | 5 |
| segmentMD . . . . .         | 6 |
| visualizeDeletion . . . . . | 7 |

**Index****8**


---

|              |                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------|
| MDTS-package | <i>The MDTS package for Detection of Denovo Deletions from Targeted Sequencing Data Using Minimum-Distance</i> |
|--------------|----------------------------------------------------------------------------------------------------------------|

---

**Description**

The MDTS package for Detection of Denovo Deletions from Targeted Sequencing Data Using Minimum-Distance

---

|          |                                                       |
|----------|-------------------------------------------------------|
| calcBins | <i>Sample driven calculation of bins for counting</i> |
|----------|-------------------------------------------------------|

---

**Description**

This function will randomly select a sample of bam files to calculate dynamic MDTS bins for subsequent read-depth analysis.

**Usage**

```
calcBins(metaData, n, readLength, medianCoverage, minimumCoverage, genome,
         mappabilityFile, seed = 1337)
```

**Arguments**

|                 |                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------|
| metaData        | A table in the format of the output of getMetaData().                                      |
| n               | The number of subsamples to use.                                                           |
| readLength      | The read length of the experiment.                                                         |
| medianCoverage  | The median number of reads across sub-samples to reach before creating a new bin.          |
| minimumCoverage | The minimum number of coverage across all sub-samples required to create the proto-region. |
| genome          | The BSGenome object that assists in calculations of the GC content of the bins.            |
| mappabilityFile | A path to the bigwig file of 100mer mappability of the corresponding genome.               |
| seed            | Sets the seed so results are reproducible. Defaults to 1337.                               |

**Value**

Returns a GRanges object depicting the dynamic bins that MDTS calculates.

**Examples**

```
load(system.file("extdata", 'bins.RData', package = "MDTS"))
bins
```

---

|            |                                      |
|------------|--------------------------------------|
| calcCounts | <i>Creating the raw count matrix</i> |
|------------|--------------------------------------|

---

**Description**

This function will return a matrix of read counts where each column is a sample, and each row is a bin.

**Usage**

```
calcCounts(metaData, bins, rl, mc.cores = 1)
```

**Arguments**

|          |                                                                        |
|----------|------------------------------------------------------------------------|
| metaData | A table in the format of the output of getMetaData().                  |
| bins     | The set of bins determined by calcBins().                              |
| rl       | The read length of the experiment.                                     |
| mc.cores | The number of cores to use for multi-threaded analysis. Defaults to 1. |

**Value**

A data.frame that contains the counts for each sample in the metaData input that fall into each segment of bins.

**Examples**

```
## Not run:
pD <- getMetaData(
  'https://raw.githubusercontent.com/JMF47/MDTSDData/master/data/pD.ped')
genome = BSgenome.Hsapiens.UCSC.hg19
map_file <-
  "https://raw.githubusercontent.com/JMF47/MDTSDData/master/data/chr1.map.bw"
bins = calcBins(pD, n=5, rl=100, med=150, min=5, genome, map_file)

## End(Not run)
load(system.file("extdata", 'bins.RData', package = "MDTS"))
load(system.file("extdata", 'counts.RData', package = "MDTS"))
counts
```

---

|        |                                                |
|--------|------------------------------------------------|
| calcMD | <i>Calculating the Minimum Distance matrix</i> |
|--------|------------------------------------------------|

---

**Description**

This function will return a matrix of minimum distances where each column is a family, and each row is a bin.

**Usage**

```
calcMD(mCounts, metaData)
```

**Arguments**

|          |                                                                            |
|----------|----------------------------------------------------------------------------|
| mCounts  | A matrix of normalized coverage output by <code>normalizeCounts()</code> . |
| metaData | A table in the format of the output of <code>metaData()</code> .           |

**Value**

A `data.frame` of minimum distances. Each column is a trio, while each row is an entry in bins

**Examples**

```
load(system.file("extdata", 'bins.RData', package = "MDTS"))
load(system.file("extdata", 'counts.RData', package = "MDTS"))
load(system.file("extdata", 'pD.RData', package = "MDTS"))
mCounts <- normalizeCounts(counts, bins)
md <- calcMD(mCounts, pD)
```

---

|                 |                                |
|-----------------|--------------------------------|
| denovoDeletions | <i>Denovo Deletion Calling</i> |
|-----------------|--------------------------------|

---

**Description**

This function will return a single `GRanges` object containing all denovo deletions that passed filtering from a Circular Binary Segmentation object with supplementary information.

**Usage**

```
denovoDeletions(cbs, mCounts, bins)
```

**Arguments**

|         |                                                                         |
|---------|-------------------------------------------------------------------------|
| cbs     | The output from <code>segmentMD()</code> .                              |
| mCounts | The normalized counts matrix output by <code>normalizeCounts()</code> . |
| bins    | The set of bins determined by <code>calcBins()</code> .                 |

**Value**

A `GRanges` object that reports all detected denovo deletions passing requisite filters.

**Examples**

```
load(system.file("extdata", 'bins.RData', package = "MDTS"))
load(system.file("extdata", 'counts.RData', package = "MDTS"))
load(system.file("extdata", 'pD.RData', package = "MDTS"))
mCounts = normalizeCounts(counts, bins)
md = calcMD(mCounts, pD)
cbs = segmentMD(md, bins)
denovo = denovoDeletions(cbs, mCounts, bins)
```

---

|             |                                 |
|-------------|---------------------------------|
| getMetaData | <i>Constructor for metadata</i> |
|-------------|---------------------------------|

---

### Description

This function allows constructor of phenotype information necessary for downstream analysis. See format of required fields. Function will also rearrange the rows such that trios are grouped together - with proband first, mother second, and father third.

### Usage

```
getMetaData(path, id = "subj_id", familyId = "family_id",
  fatherId = "father_id", motherId = "mother_id", bamPath = "bam_path")
```

### Arguments

|          |                                                                                                      |
|----------|------------------------------------------------------------------------------------------------------|
| path     | The path pointing to the file that contains information on each subject in the dataset.              |
| id       | The column name that identifies each sample. Defaults to 'subj_id'.                                  |
| familyId | The column name that identifies which family the sample belongs to. Defaults to 'family_id'.         |
| fatherId | The column name that identifies the id of the father. Defaults to 'father_id'.                       |
| motherId | The column name that identifies the id of the mother. Defaults to 'mother_id'.                       |
| bamPath  | The column name that identifies where to find the bam file for each subject. Defaults to 'bam_path'. |

### Value

Returns a data.frame of required sample information for running MDTs.

### Examples

```
meta <- getMetaData(
  'https://raw.githubusercontent.com/JMF47/MDTSData/master/data/pD.ped')
```

---

|                 |                                            |
|-----------------|--------------------------------------------|
| normalizeCounts | <i>Calculating the normalized M scores</i> |
|-----------------|--------------------------------------------|

---

### Description

This function will return a matrix of normalized M scores where each column is a sample, and each row is a bin.

### Usage

```
normalizeCounts(counts, bins, GC = TRUE, map = TRUE, mc.cores = 1)
```

**Arguments**

|          |                                                                           |
|----------|---------------------------------------------------------------------------|
| counts   | A matrix of raw coverage output by calcCounts().                          |
| bins     | The set of bins determined by calcBins().                                 |
| GC       | Whether to loess adjust for GC. Defaults to TRUE.                         |
| map      | Whether to loess adjust for mappability. Defaults to TRUE. Defaults to 1. |
| mc.cores | The number of cores to use for multi-threaded analysis.                   |

**Value**

A data.frame of normalized counts. Each column is a sample, and each row is a entry of bins.

**Examples**

```
load(system.file("extdata", 'bins.RData', package = "MDTS"))
load(system.file("extdata", 'counts.RData', package = "MDTS"))
load(system.file("extdata", 'pD.RData', package = "MDTS"))
mCounts <- normalizeCounts(counts, bins)
```

segmentMD

*Circular Binary Segmentation on Minimum Distances***Description**

This function will return a GRanges object containing the copy number segments of all families in the input minimum distance matrix. It calls segment() from DNACopy (alpha=0.001, undo.splits="sdundo", undo.SD=4).

**Usage**

```
segmentMD(md, bins, alpha = 0.001, undo.splits = "sdundo", undo.SD = 4,
mc.cores = 1)
```

**Arguments**

|             |                                                                        |
|-------------|------------------------------------------------------------------------|
| md          | The minimum distance matrix produced by calcMD.                        |
| bins        | The set of bins determined by calcBins.                                |
| alpha       | Controls the alpha option in calling DNACopy::segment()                |
| undo.splits | Controls the undo.splits option in DNACopy::segment()                  |
| undo.SD     | Controls the undo.SD option in calling DNACopy::segment()              |
| mc.cores    | The number of cores to use for multi-threaded analysis. Defaults to 1. |

**Value**

A data.frame containing the segmented regions based to be parsed by denovoDeletions() minimum distance.

**Examples**

```
load(system.file("extdata", 'bins.RData', package = "MDTS"))
load(system.file("extdata", 'counts.RData', package = "MDTS"))
load(system.file("extdata", 'pD.RData', package = "MDTS"))
mCounts <- normalizeCounts(counts, bins)
md <- calcMD(mCounts, pD)
cbs <- segmentMD(md, bins)
```

---

|                   |                                    |
|-------------------|------------------------------------|
| visualizeDeletion | <i>Visualization for deletions</i> |
|-------------------|------------------------------------|

---

**Description**

This function plots the raw read information from the location of interest for a trio.

**Usage**

```
visualizeDeletion(deletion, bins, metaData, mCounts, md, save = FALSE)
```

**Arguments**

|          |                                                                                  |
|----------|----------------------------------------------------------------------------------|
| deletion | A GRanges object in the format of the output of <code>denovoDeletions()</code> . |
| bins     | The set of bins determined by <code>calcBins()</code> .                          |
| metaData | A table in the format of the output of <code>getMetaData()</code> .              |
| mCounts  | A matrix of normalized coverage output by <code>normalizedCounts()</code> .      |
| md       | The minimum distance matrix output by <code>calcMD()</code>                      |
| save     | If TRUE will save plot to current working directory instead of rendering.        |

**Value**

The file name if the plot was saved.

**Examples**

```
## Not run:
load(system.file("extdata", 'bins.RData', package = "MDTS"))
load(system.file("extdata", 'counts.RData', package = "MDTS"))
load(system.file("extdata", 'pD.RData', package = "MDTS"))
mCounts <- normalizeCounts(counts, bins)
md <- calcMD(mCounts, pD)
cbs <- segmentMD(md, bins)
denovo <- denovoDeletions(cbs, mCounts, bins)
visualizeDeletion(denovo[1], bins, pD, mCounts, md)

## End(Not run)
```

# Index

- \* **calcBins**
  - calcBins, [2](#)
- \* **calcCounts**
  - calcCounts, [3](#)
- \* **calcMD**
  - calcMD, [3](#)
- \* **denovoDeletions**
  - denovoDeletions, [4](#)
- \* **normalizeCounts**
  - normalizeCounts, [5](#)
- \* **segmentMD**
  - segmentMD, [6](#)
- \* **visualizeDeletion**
  - visualizeDeletion, [7](#)

calcBins, [2](#)  
calcCounts, [3](#)  
calcMD, [3](#)

denovoDeletions, [4](#)

getMetaData, [5](#)

MDTS (MDTS-package), [2](#)  
MDTS-package, [2](#)

normalizeCounts, [5](#)

segmentMD, [6](#)

visualizeDeletion, [7](#)