# Package 'mitoClone2'

December 2, 2025

**Type** Package

**Title** Clonal Population Identification in Single-Cell RNA-Seq Data using Mitochondrial and Somatic Mutations

**LinkingTo** Rhtslib (>= 1.13.1)

**Version** 1.16.0

**Description** This package primarily identifies variants in mitochondrial genomes from BAM alignment files. It filters these variants to remove RNA editing events then estimates their evolutionary relationship (i.e. their phylogenetic tree) and groups single cells into clones. It also visualizes the mutations and providing additional genomic context.

**License** GPL-3

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** false

**biocViews** Annotation, DataImport, Genetics, SNP, Software, SingleCell, Alignment

**Imports** reshape2, GenomicRanges, pheatmap, deepSNV, grDevices, Matrix, graphics, stats, utils, S4Vectors, Rhtslib, parallel, methods, ggplot2

**Suggests** knitr, rmarkdown, Biostrings, testthat

**RoxygenNote** 7.1.0

**SystemRequirements** GNU make, PhISCS (optional)

**Depends** R (>= 4.4.0)

**NeedsCompilation** yes

**URL** https://github.com/benstory/mitoClone2

**Bugreports** https://github.com/benstory/mitoClone2/issues

**git_url** https://git.bioconductor.org/packages/mitoClone2

**git_branch** RELEASE_3_22

**git_last_commit** 43b5799

**git_last_commit_date** 2025-10-29

**Repository** Bioconductor 3.22

**Date/Publication** 2025-12-01

**Author** Benjamin Story [aut, cre],
Lars Velten [aut],
Gregor Mönke [aut]

**Maintainer** Benjamin Story <story.benjamin@gmail.com>

# Contents



---

`bam2R_10x` *Read nucleotide counts from a 10x Genomics .bam file*

---

## Description

This function uses a C interface to read the nucleotide counts on each position of a .bam alignment. The counts are individually tabulated for each cell barcode as specified by the user. The counts of both strands are reported separately and nucleotides below a quality cutoff are masked.

## Usage

```
bam2R_10x(
  file,
  sites = "MT:1-16569",
  q = 25,
  mq = 0,
  s = 2,
  head.clip = 0,
```

```
  max.depth = 1e+06,
  verbose = FALSE,
  mask = 0,
  keepflag = 0,
  max.mismatches = NULL,
  ncores = 1,
  ignore_nonstandard = FALSE,
  min_reads_per_barcode = 50
)
```

## Arguments

| | |
|---|---|
| file | The file location of the BAM file as a string. |
| sites | The chromosome locations of interest in BED format as a string. Alternatively a single GRanges object will also work. |
| q | An optional cutoff for the nucleotide Phred quality. Default q = 25. Nucleotides with Q < q will be masked by 'N'. |
| mq | An optional cutoff for the read mapping quality. Default mq = 0 (no filter). reads with MQ < mq will be discarded. |
| s | Optional choice of the strand. Defaults to s = 2 (both). |
| head.clip | Should n nucleotides from the head of reads be clipped? Default 0. |
| max.depth | The maximal depth for the pileup command. Default 1,000,000. |
| verbose | Boolean. Set to TRUE if you want to get additional output. |
| mask | Integer indicating which flags to filter. Default 0 (no mask). Try 1796 (BAM_DEF_MASK). |
| keepflag | Integer indicating which flags to keep. Default 0 (no mask). Try 3 (PAIRED\|PROPERLY_PAIRED). |
| max.mismatches | Integer indicating maximum MN value to allow in a read. Default NULL (no filter). |
| ncores | Integer indicating the number of threads to use for the parallel function call that summarize the results for each bam file. Default 1. |
| ignore_nonstandard | Boolean indicating whether or not gapped alignments, insertions, or deletions should be included in the final output. Default FALSE. If you have an inflation of spliced mitochondrial reads it is recommended to set this to TRUE. |
| min_reads_per_barcode | Int defining how many reads a barcode must for it to be considered in the pileup tabulations. Default 50 |

## Details

This code is an adaption of code that was originally written by Moritz Gerstung for the deepSNV package

## Value

A named `list` of `matrix` with rows corresponding to genomic positions and columns for the nucleotide counts (A, T, C, G, -), masked nucleotides (N), (INS)ertions, (DEL)etions that count how often a read begins and ends at the given position, respectively. Each member of the list corresponds to an invididual cells or entity based on the cell barcode of interest. The names of the elements of the list correspond to the respective cell barcodes. For the intents and purposes of the mitoClone2

package this object is equivalent to the output from the baseCountsFromBamList function. The returned list has a variable length depending on the ignore_nonstandard parameter and each element contains a matrix has 8 columns and (stop - start + 1) rows. The two strands have their counts merged. If no counts are present in the provided sites parameter nothing will be returned. IMPORTANT: The names of the list will NOT reflect the source filename and will exclusively be named based on the respective the barcodes extracted from said file. If merging multiple datasets, it is important to change the list's names once imported to avoid naming collisions.

## Author(s)

Benjamin Story (adapted from original code with permission from Moritz Gerstung)

## Examples

```
bamCounts <- bam2R_10x(file = system.file("extdata",
"mm10_10x.bam", package="mitoClone2"), sites="chrM:1-15000")
```

---

# baseCountsFromBamList *Create a list object from a list of single-cell BAM files where each contains a matrix of the of AGCT nt counts at chosen sites*

---

## Description

Uses the deepSNV package to count nucleotide frequencies at every position in the mitochondrial genome for every cell.

## Usage

```
baseCountsFromBamList(
  bamfiles,
  sites = "chrM:1-16569",
  ncores = 1,
  ignore_nonstandard = FALSE
)
```

## Arguments

| | |
|---|---|
| bamfiles | A character vector specifying the bam file paths |
| sites | String specifying genomic regions, defaults to the entire mitochondrial genome |
| ncores | Number of threads to use for the computation. Default 1 |
| ignore_nonstandard | Ignore basecalls that are not AGCTN |

## Value

A list of base count matrices which can serve as an input to mutationCallsFromExclusionlist or mutationCallsFromCohort

## Examples

```
bamCounts <- baseCountsFromBamList(bamfiles =
list(system.file("extdata", "mm10_10x.bam",
package="mitoClone2")),sites="chrM:1-15000", ncores=1)
```

`clusterMetaclones` *Cluster mutations into clones - following the tree structure*

## Description

PhISCS orders all mutations into a hierarchical mutational tree; in many cases, the exact order of the acquisition of individual mutations in not unanimously determined from the data. This function computes the change in likelihood of the infered clonal assignment if two mutations are merged into a clone. Hierarchical clustering is then used to determine the clonal structure. The result is visualized and should be fine-tuned using the `min.lik` parameter.

## Usage

```
clusterMetaclones(mutcalls, min.lik = 1, plot = TRUE)
```

## Arguments

| | |
|---|---|
| `mutcalls` | mutcalls object of class `mutationCalls` for which `varCluster` has been run |
| `min.lik` | specifies the minimum difference in likelihood required. This parameter is set arbitrarily, see the vignette "Computation of clonal hierarchies and clustering of mutations" for more information. |
| `plot` | whether dendrograms should be plotted. |

## Value

Returns the provided `mutationCalls` class object with an additional 'mainClone' metadata which allows for further refinement of clonal population and association of cells with a cluster of mutations (in this case clones).

## Examples

```
P1 <- readRDS(system.file("extdata/sample_example1.RDS",package = "mitoClone2"))
P1 <- clusterMetaclones(P1)
## access via mainClone metadata
```

`data` *Mitochondrial exclusionlist*

## Description

List of variants that are likely not true somatic mutations and should thus be excluded

M: Mutant allele counts; N: Reference allele counts. P1: Patient 1; P2: Patient 2

## Usage

```
exclusionlists

M_P1

N_P1

M_P2

N_P2
```

## Format

A list with four entries: #'

- *three*: Regions of the mitochondrial genome that are within 1 nt of a 3-mer homopolymer (e.g. AAA)
- *mutaseq*: Mutations in the mitochondrial genome that were reoccuring across patients (present in more than one individual in the MutaSeq dataset)
- *masked*: Regions of the mitochondrial genome that are soft-masked in the UCSC or Ensembl annotations
- *rnaEDIT*: Regions of the mitochondrial genome that are thought to be subject to RNA-editing according to the REDIportal V2.0

a data frame of variable sites (columns) across single cells (rows)

An object of class `data.frame` with 1430 rows and 16 columns.

An object of class `data.frame` with 1430 rows and 16 columns.

An object of class `data.frame` with 1066 rows and 22 columns.

An object of class `data.frame` with 1066 rows and 22 columns.

---

# getAlleleCount *mutationCalls counts accessor*

---

## Description

Extracts the counts of allele for either the mutant or all the non-mutant alleles

## Usage

```
getAlleleCount(mutcall, type = c("mutant", "nonmutant"))
```

## Arguments

| | |
|---|---|
| `mutcall` | object of class `mutationCalls`. |
| `type` | character that is either ‘mutant‘ or ‘nonmutant‘ depending on which allele count the user wants to access |

## Value

Returns matrix of either mutant or non-mutant allele counts

## Examples

```
load(system.file("extdata/LudwigFig7.Rda",package = "mitoClone2"))
mutantAllele_count <- getAlleleCount(LudwigFig7,type='mutant')
```

---

`getCloneLikelihood` *mutationCalls accessors*

---

## Description

Retrieves the full matrix of likelihoods associating single cells with clones

## Usage

```
getCloneLikelihood(mutcall, mainClones = length(mutcall@mut2clone) > 0)

getMainClone(mutcall, mainClones = length(mutcall@mut2clone) > 0)

getConfidence(mutcall, mainClones = length(mutcall@mut2clone) > 0)

getMut2Clone(mutcall)
```

## Arguments

| | |
|---|---|
| `mutcall` | object of class `mutationCalls`. |
| `mainClones` | Retrieve likelihoods associated with the main Clones. Defaults to `TRUE` if `clusterMetaclones` has been run. |

## Value

Return `TRUE` if `clusterMetaclones` has been run otherwise returns the cell by clone matrix of likelihood associating each cell to a given clone.

## Functions

- `getMainClone()`: Retrieve the most likely clone associate with each cell.
- `getConfidence()`: Retrieve the likelihood of the most likely clone for each cell.
- `getMut2Clone()`: Retrieve the assignment of mutations to clones, once `clusterMetaclones` has been run.

## Examples

```
load(system.file("extdata/LudwigFig7.Rda",package =
"mitoClone2"))
likelihood_matrix <- getCloneLikelihood(LudwigFig7)
```

## getVarsCandidate *mutationCalls cluster accessor*

### Description

Extracts all the putative variants that we want to use for clustering

### Usage

```
getVarsCandidate(mutcall)
```

### Arguments

| | |
|---|---|
| mutcall | object of class mutationCalls. |

### Value

Returns a character vector including all the variants to be used for clustering

### Examples

```
load(system.file("extdata/LudwigFig7.Rda",package =
"mitoClone2"))
mutations_to_cluster <- getVarsCandidate(LudwigFig7)
```

## mitoPlot *Plot clone-specific variants in circular plots*

### Description

Plot clone-specific variants in circular plots

### Usage

```
mitoPlot(
  variants,
  patient = NULL,
  genome = "hg38",
  customGenome = NULL,
  showLegend = TRUE,
  showLabel = TRUE
)
```

### Arguments

| | |
|---|---|
| variants | Character vector of variants to plot in format 5643G>T or 5643 G>T. |
| patient | Characet vector identifying which variant belongs to what clone. The order should match that of the 'vars' parameter and shoul dbe of identical length. If none is provided, the function assumes all variants are from one single sample which will be named "Main Clone". Default: NULL. |
| genome | The mitochondrial genome of the sample being investigated. Please note that this is the UCSC standard chromosome sequence. Default: hg38. |
| customGenome | A GRanges object containing a custom annotation. If provided, this genome will be used instead of the predefined options specified by the 'genome' parameter. Default is NULL. |
| showLegend | Boolean for whether or not the gene legend should be present in the final output plot. Default: TRUE. |
| showLabel | Boolean for whether or not the name of the variant should be shown as a label in the final output plot. Default: TRUE. |

### Value

A ggplot object illustrating the clone specific mutations.

### Examples

```
known.variants <- c("9001 T>C","12345 G>A","1337 G>A")
mitoPlot(known.variants)
```

---

## mut2GR *Convert mutation string to GRanges*

---

### Description

Convert mutation string to GRanges

### Usage

```
mut2GR(mut)
```

### Arguments

| | |
|---|---|
| mut | The mutation to convert to a GRanges in the format of "position reference>alternate". |

### Value

Returns a GRanges object containg the site of the variant along with reference/alternate allele data in the metacolumns

### Examples

```
mutation.as.granges <- mut2GR('1434 G>A')
mutation.as.granges.no.space <- mut2GR('1434G>A')
```

## mutationCalls-class *mutationCalls class*

### Description

To create this class from a list of bam files (where each bam file corresponds to a single cell), use `mutationCallsFromCohort` or `mutationCallsFromExclusionlist`. To create this class if you already have the matrices of mutation counts, use its contstructor, i.e. `mutationCallsFromMatrix(M = data1, N = data2)`.

### Slots

`M` A matrix of read counts mapping to the *mutant* allele. Columns are genomic sites and rows and single cells.

`N` A matrix of read counts mapping to the *nonmutant* alleles. Columns are genomic sites and rows and single cells.

`ternary` Discretized version describing the mutational status of each gene in each cell, where 1 signfiies mutant, 0 signifies reference, and ? signifies dropout

`cluster` Boolean vector of length `ncol(M)` specifying if the given mutation should be included for clustering (`TRUE`) or only used for annotation.

`metadata` Metadata frame for annotation of single cells (used for plotting). Row names should be the same as in `M`

`tree` Inferred mutation tree

`cell2clone` Probability matrix of single cells and their assignment to clones.

`mut2clone` Maps mutations to main clones

`mainClone` Probability matrix of single cells and their assignment to main clones

`treeLikelihoods` Likelihood matrix underlying the inference of main clones, see `clusterMetaclones`

## mutationCallsFromCohort

*Create a mutationCalls objects from nucleotide base calls and defines a exclusionlist (cohort)*

### Description

Identifies relevant mitochondrial somatic variants from raw counts of nucleotide frequencies measured in single cells from several individuals. Applies two sets of filters: In the first step, filters on coverage to include potentially noisy variants; in the second step, compares allele frequencies between patients to remove variants that were observed in several individuals and that therefore are unlikely to represent true somatic variants (e.g. RNA editing events). The exclusionlist derived from the original Velten et al. 2021 dataset is available internal and can be used on single individuals using `mutationCallsFromExclusionlist`

## Usage

```
mutationCallsFromCohort(
  BaseCounts,
  sites,
  patient,
  MINREADS = 5,
  MINCELL = 20,
  MINFRAC = 0.1,
  MINCELLS.PATIENT = 10,
  MINFRAC.PATIENT = 0.01,
  MINFRAC.OTHER = 0.1,
  USE.REFERENCE = TRUE,
  genome = "hg38",
  customGenome = NULL
)
```

## Arguments

| | |
|---|---|
| BaseCounts | A list of base call matrices (one matrix per cell) as produced by baseCountsFromBamList or bam2R_10x. |
| sites | Vector specifying genomic regions, defaults to the entire mitochondrial genome. Excepts a string but may be included as a GRanges object. |
| patient | A character vector associating each cell / entry in the BaseCount list with a patient |
| MINREADS | Minimum number of reads on a site in a single cell to qualify the site as covered |
| MINCELL | Minimum number of cells across the whole data set to cover a site |
| MINFRAC | Fraction of reads on the mutant allele to provisionally classify a cell as mutant |
| MINCELLS.PATIENT | Minimum number of mutant cells per patient to classify the mutation as relevant in that patient, AND |
| MINFRAC.PATIENT | Minimum fraction of mutant cells per patient to classify the mutation as relevant in that patient |
| MINFRAC.OTHER | Minimum fraction of mutant cells identified in a second patient for the mutation to be excluded. Fraction relative to the fraction of of cells from the patient where a variant is enriched. |
| USE.REFERENCE | Boolean. The variant calls will be of the format REF>ALT where REF is decided based on the selected genome annotation. If set to FALSE, the reference allele will be the most abundant. |
| genome | The mitochondrial genome of the sample being investigated. Please note that this is the UCSC standard chromosome sequence. Default: hg38. |
| customGenome | A GRanges object containing a custom annotation. If provided, this genome will be used instead of the predefined options specified by the 'genome' parameter. Default is NULL. |

## Value

A list of mutationCalls objects (one for each patient) and an entry named exclusionlist containing a exclusionlist of sites with variants in several individuals

## Examples

```
sites.gr <- GenomicRanges::GRanges("chrM:1-15000")
BaseCounts <- bam2R_10x(file = system.file("extdata",
"mm10_10x.bam", package="mitoClone2"), sites=sites.gr)
mutCalls <- mutationCallsFromCohort(BaseCounts,
patient=c('sample2','sample1','sample2','sample2','sample1'),
MINCELL=1, MINFRAC=0, MINCELLS.PATIENT=1, genome='mm10',
sites=sites.gr)
```

---

`mutationCallsFromExclusionlist` *Create a mutationCalls object from nucleotide base calls using a exclusionlist (single individual)*

---

## Description

Identifies relevant mitochondrial somatic variants from raw counts of nucleotide frequencies. Applies two sets of filters: In the first step, filters on coverage and minimum allele frequency to exclude potentially noisy variants; in the second step, filters against a exclusionlist of variants that were observed in several individuals and that therefore are unlikely to represent true somatic variants (e.g. RNA editing events). These exclusionlists are created using `mutationCallsFromCohort`

## Usage

```
mutationCallsFromExclusionlist(
  BaseCounts,
  lim.cov = 20,
  min.af = 0.2,
  min.num.samples = 0.01 * length(BaseCounts),
  min.af.universal = min.af,
  universal.var.cells = 0.95 * length(BaseCounts),
  exclusionlists.use = exclusionlists,
  max.var.na = 0.5,
  max.cell.na = 0.95,
  genome = "hg38",
  customDNA = NULL,
  ncores = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| `BaseCounts` | A list of base call matrices (one matrix per cell) as produced by `baseCountsFromBamList` |
| `lim.cov` | Minimal coverage required per cell for a cell to be classified as covered |
| `min.af` | Minimal allele frequency for a cell to be classified as mutant |
| `min.num.samples` | Minimal number of cells required to be classified as covered and mutant according to the thresholds set in `lim.cov` and `min.af`. Usually specified as a fraction of the total number of cells. |

`min.af.universal`
: Minimal allele frequency for a cell to be classified as mutant, in the context of removing universal variants. Defaults to `min.af`, but can be set to lower values.

`universal.var.cells`
: Maximum number of cells required to be classified as mutant according to the threshold set in `min.af.universal`. Usually specified as a fraction of the total number of cells; serves to avoid e.g. germline variants.

`exclusionlists.use`
: List of sites to exclude for variants calling. The default exclusionlists object included with this package contains exclude or hardmask in GRanges format. The four exclusionlists included in this case are: "three" (hg38 sites that are part of homopolymer(e.g. AAA) of at least 3 bp in length), "mutaseq" (sites discovered to be overrepresented in AML SmartSeq2 data analysis from Velten et al 2021), "masked" (sites that are softmasked in either the UCSC or Refseq genome annotations), and "rnaEDIT" which are sites that are subjected to RNA-editing according to the REDIportal. These lists can also be input manually by a researcher and provided as either coordinates (as a string) or as a GRanges objects.

`max.var.na`
: Final filtering step: Remove all mutations with no coverage in more than this fraction of cells

`max.cell.na`
: Final filtering step: Remove all cells with no coverage in more than this fraction of mutations

`genome`
: The mitochondrial genome of the sample being investigated. Please note that this is the UCSC standard chromosome sequence. Default: hg38.

`customDNA`
: A character vector containing a custom DNA sequence. If provided, this sequence will be used instead of the predefined options specified by the 'genome' parameter. Default is NULL.

`ncores`
: number of cores to use for tabulating potential variants (defaults to 2)

`...`
: Parameters passed to `mutationCallsFromMatrix`

## Value

An object of class `mutationCalls`

## Examples

```
load(system.file("extdata/example_counts.Rda",package = "mitoClone2"))
Example <- mutationCallsFromExclusionlist(example.counts,
min.af=0.05, min.num.samples=5,
universal.var.cells = 0.5 * length(example.counts),
binarize = 0.1)
```

---

# mutationCallsFromMatrix

*mutationCalls constructor*

---

## Description

To be used when allele-specific count matrices are available.

### Usage

```
mutationCallsFromMatrix(
  M,
  N,
  cluster = NULL,
  metadata = data.frame(row.names = rownames(M)),
  binarize = 0.05
)
```

### Arguments

| | |
|---|---|
| M | A matrix of read counts mapping to the *mutant* allele. Columns are genomic sites and rows and single cells. |
| N | A matrix of read counts mapping to the *referece* allele. Columns are genomic sites and rows and single cells. |
| cluster | If NULL, only mutations with coverage in 20 percent of the cells or more will be used for the clustering, and all other mutations will be used for cluster annotation only. Alternatively, a boolean vector of length ncol(M) that specifies the desired behavior for each genomic site. |
| metadata | A data.frame of metadata that will be transfered to the final output where the row.names(metadata) correspond to the the row.names(M). |
| binarize | Allele frequency threshold to define a site as mutant (required for some clustering methods) |

### Value

An object of class mutationCalls.

### Examples

```
load(system.file("extdata/example_counts.Rda",package = "mitoClone2"))
## we have loaded the example.counts object
known.variants <- c("8 T>C","4 G>A","11 G>A","7 A>G","5 G>A","15 G>A","14 G>A")
known.subset <- pullcountsVars(example.counts, known.variants)
known.subset <- mutationCallsFromMatrix(t(known.subset$M), t(known.subset$N),
cluster = rep(TRUE, length(known.variants)))
```

---

## overwriteMetaclones *Manually overwrite clustering of mutations into clones*

---

### Description

The function clusterMetaclones provides an automated way to group mutations into clones for subsequent analyses (such as differential expression analyses). In practice, it may make sense to overwrite these results manually. See the vignette 'Computation of clonal hierarchies and clustering of mutations' for an example.

### Usage

```
overwriteMetaclones(mutcalls, mutation2clones)
```

### Arguments

| | |
|---|---|
| mutcalls | mutcalls object of class mutationCalls for which clusterMetaclones has been run |
| mutation2clones | Named integer vector that assigns mutations to clones. See the vignette 'Computation of clonal hierarchies and clustering of mutations' for an example. |

### Value

Returns the provided mutationCalls class object with the 'mainClone' metadata overwritten with the manual values provided by the user.

### Examples

```
P1 <- readRDS(system.file("extdata/sample_example1.RDS",package = "mitoClone2"))
new.n <- seq(17)
names(new.n) <- names(getMut2Clone(P1))
P1.newid <- overwriteMetaclones(P1,new.n)
```

---

## plotClones *Plot clonal assignment of single cells*

---

### Description

Creates a heatmap of single cell mutation calls, clustered using PhISCS.

### Usage

```
plotClones(mutcalls, what = c("alleleFreq", "ternary"), show = c(), ...)
```

### Arguments

| | |
|---|---|
| mutcalls | object of class mutationCalls. |
| what | One of the following: *alleleFreq*: The fraction of reads mapping to the mutant allele or *ternary*: Ternarized mutation status |
| show | boolean vector specifying for each mutation if it should be plotted on top of the heatmap as metadata; defaults to mutations not used for the clustering !mutcalls@cluster |
| ... | any arguments passed to pheatmap |

### Value

Returns TRUE only used for generating a PostScript tree image of the putative mutation tree

### Examples

```
P1 <-
readRDS(system.file("extdata/sample_example1.RDS",package =
"mitoClone2"))
plotClones(P1)
```

## predictCellAssignment *Predict cell assignments from fitted Vireo model*

### Description

Predict cell assignments from fitted Vireo model

### Usage

```
predictCellAssignment(model, threshold = 0.9)
```

### Arguments

| | |
|---|---|
| model | Fitted Vireo model |
| threshold | Minimum probability threshold for assignment |

### Value

Data frame with cell assignments and probabilities

### Examples

```
load(system.file("extdata/LudwigFig7.Rda",package = "mitoClone2"))
test.data <- list(N=as.matrix(t(LudwigFig7@N)),M=as.matrix(t(LudwigFig7@M)))
vireoModel <- vireoFit(test.data, n.donor = 9, filter.variants = FALSE, min_cells_per_sample = 5)
cellAssignments <- predictCellAssignment(vireoModel, threshold = 0.9)
```

## pullcountsVars *Pull variant counts*

### Description

Pull variant counts

### Usage

```
pullcountsVars(BaseCounts, vars, cells = NULL)
```

### Arguments

| | |
|---|---|
| BaseCounts | A list of base call matrices (one matrix per cell) as produced by baseCountsFromBamList |
| vars | Character vector of variants to pull, in format 5643G>T |
| cells | Character vector for cells to select, or NULL if all cells from the input are to be used |

### Value

A list with two entries, M (count table on the variant allele) and N (count table on the reference allele)

### Examples

```
load(system.file("extdata/example_counts.Rda",package = "mitoClone2"))
known.variants <- c("9 T>C","12 G>A","13 G>A")
counts.known.vars <- pullcountsVars(example.counts, vars=known.variants)
```

---

## quick_cluster *Quick clustering of mutations*

---

### Description

Performs a quick hierarchical clustering on a object of class mutationCalls. See varCluster for an alternative that infers mutational trees and uses sound models of dropout.

### Usage

```
quick_cluster(mutcalls, binarize = FALSE, drop_empty = TRUE, ...)
```

### Arguments

| | |
|---|---|
| mutcalls | object of class mutationCalls. |
| binarize | If FALSE, will use raw allele frequencies for the clustering. If TRUE, will use binarized mutation/reference/dropout calls. |
| drop_empty | Remove all rows in the provided mutcalls object where no cells exhibit a mutation. |
| ... | Parameters passed to pheatmap |

### Value

The result of running pheatmap

### Examples

```
load(system.file("extdata/LudwigFig7.Rda",package = "mitoClone2"))
quickCluster <- quick_cluster(LudwigFig7)
```

---

## removeWindow *Remove mutations that occuring at the same site*

---

### Description

Mutations co-occuring at the same genomic position may often be the result of sequencing artifacts or technical biases. In cases where the user which to drop these from a result this function may be used. ONLY WORKS FOR MITOCHONDRIAL MUTATIONS.

### Usage

```
removeWindow(x, window = 1)
```

### Arguments

| | |
|---|---|
| x | A list of strings that comprise sites that will be filtered |
| window | Integer of how close mutations must be to one another (in bp) to be removed |

### Value

Returns the same list of mutations excluding those, if any, that fall within the same window =

### Examples

```
P1.muts <- rep(TRUE,3)
names(P1.muts) <- c("X2537GA","X3351TC","X3350TC")
names(P1.muts) <- gsub("^X","",
gsub("(\\d+)([AGCT])([AGCT])","\\1 \\2>\\3",names(P1.muts)))
P1.muts <- P1.muts[removeWindow(names(P1.muts))]
```

---

## setVarsCandidate *mutationCalls cluster setter*

---

### Description

Sets the putative variants that we want to use for clustering

### Usage

```
setVarsCandidate(mutcall, varlist)
```

### Arguments

| | |
|---|---|
| mutcall | object of class mutationCalls. |
| varlist | vector of booleans with the names set to the variants to use for clustering |

### Value

Sets the cluster slot on a mutationCalls object

### Examples

```
load(system.file("extdata/LudwigFig7.Rda",package =
"mitoClone2"))
mutations_to_cluster <- getVarsCandidate(LudwigFig7)
mutations_to_cluster[] <- rep(c(TRUE,FALSE),each=19)
LudwigFig7 <- setVarsCandidate(LudwigFig7,mutations_to_cluster)
```

varCluster *Inference of mutational trees by of single cell mutational status*

## Description

From data on the observed mutational status of single cells at a number of genomic sites, computes a likely phylogenetic tree using PhISCS (https://github.com/sfu-compbio/PhISCS) and associates single cells with leaves of the tree. The function `clusterMetaclones` should be called on the output in order to group mutations into clones using a likelihood-based approach.

## Usage

```
varCluster(
  mutcalls,
  fn = 0.1,
  fp = 0.02,
  cores = 1,
  time = 10000,
  tempfolder = tempdir(),
  python_env = "",
  force_recalc = FALSE,
  method = "SCITE"
)
```

## Arguments

| | |
|---|---|
| `mutcalls` | object of class `mutationCalls`. |
| `fn` | false negative rate, i.e. the probability of only observing the reference allele if there is a mutation. #add gene-wise |
| `fp` | false positive, i.e. the probability of observing the mutant allele if there is no mutation. |
| `cores` | number of cores to use for PhISCS (defaults to 1) |
| `time` | maximum time to be used for PhISCS optimization, in seconds (defaults to 10000) |
| `tempfolder` | temporary folder to use for PhISCS output |
| `python_env` | Any shell commands to execute in order to make the gurobi python package available. The easiest solution is running R from an environment where the gurobi python package is avaiable. In some settings (e.g. RStudio Server), this parameter can be used instead. `muta_clone` executes PhISCS using a `system` call to python. The value of this parameter is prepended to the call. If you have a conda environment `myenv` that contains gurobipy, `source activate myenv` can work. Occassionally RStudio Server modifies your PATH so that that the conda and source commands are not available. In that case you can for example use `export PATH=/path/to/conda/:$PATH; source activate myenv`. easybuild users can `module load anaconda/v3; source activate myenv` |
| `force_recalc` | Rerun PhISCS even if the `tempfolder` contains valid PhISCS output |
| `method` | A string variable of either PhISCS or SCITE depending on the tree-inferring software the user wants to use. Default: PhISCS |

## Value

an object of class mutationCalls, with an inferred tree structure and cell to clone assignment added.

## Examples

```
load(system.file("extdata/LudwigFig7.Rda",package =
"mitoClone2"))
LudwigFig7 <- varCluster(LudwigFig7,
python_env = "",method='SCITE')
```

---

# vireo *Initialize Vireo model*

---

## Description

Initialize Vireo model

## Usage

```
vireo(
  n.cell,
  n.var,
  n.donor,
  n.gt = 3,
  learn.gt = TRUE,
  fix.beta.sum = FALSE,
  beta.mu.init = NULL,
  beta.sum.init = NULL,
  id.prob.init = NULL,
  gt.prob.init = NULL
)
```

## Arguments

| | |
|---|---|
| n.cell | Number of cells |
| n.var | Number of variants |
| n.donor | Number of donors |
| n.gt | Number of genotype states (default 3: 0,1,2) |
| learn.gt | Whether to learn genotype probabilities |
| fix.beta.sum | Whether to fix beta sum parameters |
| beta.mu.init | Initial beta mu values |
| beta.sum.init | Initial beta sum values |
| id.prob.init | Initial ID probabilities |
| gt.prob.init | Initial genotype probabilities |

vireoFit *Fit Vireo model with multiple initializations*

## Description

Fit Vireo model with multiple initializations

## Usage

```
vireoFit(
  data,
  n.donor,
  n.gt = 3,
  learn.gt = TRUE,
  n.init = 10,
  max.iter = 200,
  random.seed = NULL,
  verbose = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| data | A mitoClone2 data object containing M (ALT) and N (non-ALT) matrices |
| n.donor | Number of donors to identify |
| n.gt | Number of genotype states (default 3) |
| learn.gt | Whether to learn genotype probabilities |
| n.init | Number of random initializations |
| max.iter | Maximum iterations per initialization |
| random.seed | Random seed for reproducibility |
| verbose | Print progress messages |
| ... | Additional arguments passed to vireo.filter |

## Value

Best fitted Vireo model

## Examples

```
load(system.file("extdata/LudwigFig7.Rda",package = "mitoClone2"))
test.data <- list(N=as.matrix(t(LudwigFig7@N)),M=as.matrix(t(LudwigFig7@M)))
vireoModel <- vireoFit(test.data, n.donor = 9, filter.variants = FALSE, min_cells_per_sample = 5)
```

# Index