Package ‘CLONETv2’

October 12, 2022

Type Package

Title Clonality Estimates in Tumor

Version 2.2.1

Author Davide Prandi [aut],
Alessandro Romanel [ctb],
Tarcisio Fedrizzi [ctb],
Yari Ciani [cre]

Maintainer Yari Ciani <yari.ciani@unitn.it>

Description Analyze data from next-generation sequencing experiments on genomic samples. 'CLONETv2' offers a set of functions to compute allele specific copy number and clonality from segmented data and SNPs position pileup. The package has also calculated the clonality of single nucleotide variants given read counts at mutated positions. The package has been developed at the laboratory of Computational and Functional Oncology, Department of CIBIO, University of Trento (Italy), under the supervision of prof Francesca Demichelis. References: Prandi et al. (2014) <doi:10.1186/s13059-014-0439-6>; Carreira et al. (2014) <doi:10.1126/scitranslmed.3009448>; Romanel et al. (2015) <doi:10.1126/scitranslmed.aac9511>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 6.1.1

Depends R (>= 3.1)

Imports parallel, sets, ggplot2, ggrepel, arules, dbscan

NeedsCompilation no

Repository CRAN

Date/Publication 2021-10-13 20:40:14 UTC

R topics documented:

`CLONETv2-package` *CLONETv2*

Description

This package is designed to analyze data from next-generation sequencing experiments on genomic samples. It offers a set of functions to compute allele specific copy number and clonality from segmented data and SNPs position pileup. The library also calculated the clonality of single nucleotide variants given read counts at mutated positions.

The package has been developed at the laboratory of Computational and Functional Oncology, Department of CIBIO, University of Trento (Italy), under the supervision of prof. Francesca Demichelis.

Author(s)

Maintainer: Davide Prandi

Contributors: Tarcisio Fedrizzi, Alessandro Romanel

References

Prandi, D., Baca, S. C., Romanel, A., Barbieri, C. E., Mosquera, J. M., Fontugne, J., Beltran, H., Sboner, A., Garraway, L. A., Rubin, M. A., and Demichelis, F. (2014). Unraveling the clonal hierarchy of somatic genomic aberrations. Genome biology 15, 439.

Carreira, S., Romanel, A., Goodall, J., Grist, E., Ferraldeschi, R., Miranda, S., Prandi, D., Lorente, D., Frenel, J. S., Pezaro, C., et al. (2014). Tumor clone dynamics in lethal prostate cancer. Science translational medicine 6, 254ra125.

Beltran, H., Eng, K., Mosquera, J. M., Sigaras, A., Romanel, A., Rennert, H., Kossai, M., Pauli, C., Faltas, B., Fontugne, J., et al. (2015). Whole-Exome Sequencing of Metastatic Cancer and Biomarkers of Treatment Response. JAMA Oncol 1, 466-474.

Faltas, B. M., Prandi, D., Tagawa, S. T., Molina, A. M., Nanus, D. M., Sternberg, C., Rosenberg, J., Mosquera, J. M., Robinson, B., Elemento, O., et al. (2016). Clonal evolution of chemotherapyresistant urothelial carcinoma. Nature genetics 48, 1490-1499.

Examples

```
###############
###############
## Diploid tumor sample

## Load example data
seg_tb <- read.table(system.file("sample1.seg", package = "CLONETv2"),header = TRUE, as.is=TRUE)
pileup_tumor <- read.table(
  gzfile(system.file("sample1_tumor_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)
pileup_normal <- read.table(
  gzfile(system.file("sample1_normal_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)
snv_reads <- read.table(system.file("sample1_snv_read_count.tsv", package = "CLONETv2"),
  header = TRUE, as.is=TRUE, comment.char = "", check.names = FALSE, na.strings = "-")

## Compute beta table with default parameters
bt <- compute_beta_table(seg_tb, pileup_tumor, pileup_normal)

## Compute ploidy table with default parameters
pl_table <- compute_ploidy(bt)

## Compute admixture table with default parameters (admixture= 1-tumor_purity)
adm_table <- compute_dna_admixture(beta_table = bt, ploidy_table = pl_table)

## Check ploidy and admixture estimates
check_plot <-  check_ploidy_and_admixture(beta_table = bt, ploidy_table = pl_table,
  admixture_table = adm_table)
print(check_plot)

## Compute clonality table with default parameters
scna_clonality_table <- compute_scna_clonality_table(beta_table = bt, ploidy_table = pl_table,
  admixture_table = adm_table)

## Compute allele specific scna
allele_specific_cna_table <- compute_allele_specific_scna_table(beta_table = bt,
  ploidy_table = pl_table, admixture_table = adm_table)

## Compute snvs colonality
sample_id <- "sample1"
snv_clonality_table <- compute_snv_clonality(sample_id = sample_id, snv_read_count = snv_reads,
  beta_table = bt, ploidy_table = pl_table, admixture_table = adm_table)
```

```
###############
###############
## Aneuploid tumor sample

## Load example data
seg_tb <- read.table(system.file("sample2.seg", package = "CLONETv2"),header = TRUE, as.is=TRUE)
pileup_tumor <- read.table(
  gzfile(system.file("sample2_tumor_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)
pileup_normal <- read.table(
  gzfile(system.file("sample2_normal_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)
snv_reads <- read.table(system.file("sample2_snv_read_count.tsv", package = "CLONETv2"),
  header = TRUE, as.is=TRUE, comment.char = "", check.names = FALSE, na.strings = "-")

## Compute beta table with default parameters
bt <- compute_beta_table(seg_tb, pileup_tumor, pileup_normal)

## Compute ploidy table with default parameters
pl_table <- compute_ploidy(bt)

## Compute admixture table with default parameters (admixture= 1-tumor_purity)
adm_table <- compute_dna_admixture(beta_table = bt, ploidy_table = pl_table)

## Check ploidy and admixture estimates
check_plot <-  check_ploidy_and_admixture(beta_table = bt, ploidy_table = pl_table,
  admixture_table = adm_table)
print(check_plot)

## Compute clonality table with default parameters
scna_clonality_table <- compute_scna_clonality_table(beta_table = bt, ploidy_table = pl_table,
  admixture_table = adm_table)

## Compute allele specific scna
allele_specific_cna_table <- compute_allele_specific_scna_table(beta_table = bt,
  ploidy_table = pl_table, admixture_table = adm_table)

## Compute snvs colonality
sample_id <- "sample2"
snv_clonality_table <- compute_snv_clonality(sample_id = sample_id, snv_read_count = snv_reads,
  beta_table = bt, ploidy_table = pl_table, admixture_table = adm_table)


###############
###############
## Aneuploidy tumor sample with problematic ploidy estimate

## Load example data
seg_tb <- read.table(system.file("sample3.seg", package = "CLONETv2"),header = TRUE, as.is=TRUE)
pileup_tumor <- read.table(
  gzfile(system.file("sample3_tumor_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)
```

```
pileup_normal <- read.table(
  gzfile(system.file("sample3_normal_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)

## Compute beta table with default parameters
bt <- compute_beta_table(seg_tb, pileup_tumor, pileup_normal)

## Compute ploidy table with default parameters
pl_table <- compute_ploidy(bt)

## Compute admixture table with default parameters (admixture= 1-tumor_purity)
adm_table <- compute_dna_admixture(beta_table = bt, ploidy_table = pl_table)

## Check ploidy and admixture estimates
check_plot <-  check_ploidy_and_admixture(beta_table = bt, ploidy_table = pl_table,
  admixture_table = adm_table)
print(check_plot)
## Observed data (gray points) does not fit with expcted positions (Red circles)


###############
###############
## Tumor sample with problem in the segmented input data

## Load example data
seg_tb <- read.table(system.file("sample4.seg", package = "CLONETv2"),header = TRUE, as.is=TRUE)
pileup_tumor <- read.table(
  gzfile(system.file("sample4_tumor_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)
pileup_normal <- read.table(
  gzfile(system.file("sample4_normal_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)

## Compute beta table with default parameters
bt <- compute_beta_table(seg_tb, pileup_tumor, pileup_normal)

## Compute ploidy table with default parameters
pl_table <- compute_ploidy(bt)

## Compute admixture table with default parameters (admixture= 1-tumor_purity)
adm_table <- compute_dna_admixture(beta_table = bt, ploidy_table = pl_table)

## Check ploidy and admixture estimates
check_plot <-  check_ploidy_and_admixture(beta_table = bt, ploidy_table = pl_table,
  admixture_table = adm_table)
print(check_plot)
## CLONETv2 does not provide an estimate of the DNA admixture because
## (LogR, beta) data does not fit any CLONETv2 model


###############
###############
## Diploid tumor sample with subclonal hemizygous and homozygous deletions
## Load example data
seg_tb <- read.table(system.file("sample5.seg", package = "CLONETv2"),header = TRUE, as.is=TRUE)
pileup_tumor <- read.table(
  gzfile(system.file("sample5_tumor_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)
pileup_normal <- read.table(
  gzfile(system.file("sample5_normal_pileup.tsv.gz", package = "CLONETv2")),
  header = TRUE, as.is=TRUE)
snv_reads <- read.table(system.file("sample5_snv_read_count.tsv", package = "CLONETv2"),
  header = TRUE, as.is=TRUE, comment.char = "", check.names = FALSE, na.strings = "-")

## Compute beta table with default parameters
bt <- compute_beta_table(seg_tb, pileup_tumor, pileup_normal)

## Compute ploidy table with default parameters
pl_table <- compute_ploidy(bt)

## Compute admixture table with default parameters (admixture= 1-tumor_purity)
adm_table <- compute_dna_admixture(beta_table = bt, ploidy_table = pl_table)

## Check ploidy and admixture estimates
check_plot <-  check_ploidy_and_admixture(beta_table = bt, ploidy_table = pl_table,
  admixture_table = adm_table)
print(check_plot)

## Compute clonality table with default parameters
scna_clonality_table <- compute_scna_clonality_table(beta_table = bt, ploidy_table = pl_table,
  admixture_table = adm_table)

## Compute allele specific scna
allele_specific_cna_table <- compute_allele_specific_scna_table(beta_table = bt,
  ploidy_table = pl_table, admixture_table = adm_table)

## Compute snvs colonality
sample_id <- "sample5"
snv_clonality_table <- compute_snv_clonality(sample_id = sample_id, snv_read_count = snv_reads,
  beta_table = bt, ploidy_table = pl_table, admixture_table = adm_table)
```

adm_table_toy *Toy example of admixture table.*

Description

Toy example of admixture table.

Usage

```
adm_table_toy
```

Format

An object of class `data.frame` with 1 rows and 4 columns.

`allele_specific_cna_table_toy` *Toy example of allele specific table of somatic copy number.*

Description

Toy example of allele specific table of somatic copy number.

Usage

```
allele_specific_cna_table_toy
```

Format

An object of class `data.frame` with 4 rows and 15 columns.

`bt_toy` *Toy example of beta table.*

Description

Toy example of beta table.

Usage

```
bt_toy
```

Format

An object of class `data.frame` with 4 rows and 10 columns.

check_ploidy_and_admixture

Function to compute ploidy from a beta table.

Description

This function takes the beta table of a tumor sample and returns its ploidy.

Usage

```
check_ploidy_and_admixture(beta_table, ploidy_table, admixture_table)
```

Arguments

`beta_table`	data.frame formatted as the output of function `compute_beta_table`
`ploidy_table`	data.frame formatted as the output of function `compute_ploidy`
`admixture_table`	data.frame formatted as the output of function `compute_dna_admixture`

Value

A ggplot2 plot reporting log2 on the x axis and beta and the y axis. Each dot represents a segment of the input beta_table. Red transparent circles corresponds to expected log2 vs beta position for different allele specific copy number combinations given ploidy and admixture reported in tables ploidy_table and admixture_table, respectively. Labels in the form (cnA, cnB) indicate repsectively the major and minor allele copy number value. Labels above the plot comprises sample name and ploddy/admixture estimates.

Author(s)

Davide Prandi

Examples

```
## check ploidy and admixture estimates
check_plot_toy <- check_ploidy_and_admixture(beta_table = bt_toy, ploidy_table = pl_table_toy,
  admixture_table = adm_table_toy)
```

`compute_allele_specific_scna_table` *Function to compute allele specific somatic copy number*

Description

This function takes the beta table of a tumor sample together with the associated ploidy and admixtures tables and computes the allele specific copy number of each segment in the beta table.

Usage

```
compute_allele_specific_scna_table(beta_table, ploidy_table,
  admixture_table, error_tb = error_table, allelic_imbalance_th = 0.5,
  n_digits = 3, n_cores = 1, debug = F)
```

Arguments

`beta_table` data.frame formatted as the output of function `compute_beta_table`

`ploidy_table` data.frame formatted as the output of function `compute_ploidy`

`admixture_table` data.frame formatted as the output of function `compute_dna_admixture`

`error_tb` data.frame that reports for each combination of coverage and number informative SNPs the expected estimation error around beta. The data.frame error_tb must contains 3 columns:

- **mean.cov** mean coverage
- **n.info.snps** number of informative SNPs
- **adm.estimation.error** estimated error on computed beta on a segment with coverage mean.cov and n.info.snps informative SNPs

Package CLONETv2 have built in error_tb named error_table (default=error_table)

`allelic_imbalance_th` maximum distance from allele spefici copy number of a segment to define integer alelle specific copy number value. Value 0.5 corresponds to round cnA and cnB (default=0.5)

`n_digits` number of digits in the output table (default=3)

`n_cores` number of cores (default=1)

`debug` return extra columns for debugging (default=F)

Value

A data.frame that extends input beta_table with columns

- **log2.corr** log2 ratio adjusted by ploidy and admixture
- **cnA** copy number of the major allele
- **cnB** copy number of the minor allele
- **cnA.int** integet copy number of the major allele
- **cnB.int** integet copy number of the minor allele

Author(s)

Davide Prandi

Examples

```
## Compute clonality table with default parameters
allele_specific_cna_table_toy <- compute_allele_specific_scna_table(
  beta_table = bt_toy, ploidy_table = pl_table_toy,
  admixture_table = adm_table_toy)
```

---

compute_beta_table *Function to compute beta table*

Description

This function takes segmented data and per base pileup of tumor and matched normal of a sample as input and associates a beta value to each genomic segment.

Usage

```
compute_beta_table(seg_tb, pileup_tumor, pileup_normal,
  min_coverage = 20, min_required_snps = 10, min_af_het_snps = 0.2,
  max_af_het_snps = 0.8, n_digits = 3, n_cores = 1, plot_stats = F,
  debug = F)
```

Arguments

`seg_tb`	data.frame in SEG format. Rows report per segment log2 ratio numeric value. CLONETv2 inteprets first column as sample name, columns two to four as genomic coordinates (chromosome, start location, and end location), column five is not used, and column six is the log2 ratio returned by segmentation algorithm.
`pileup_tumor, pileup_normal`	data.frame reporting pileup of SNPs in tumor and normal samples respectively. First row contains column names and subsequent rows report the pileup of a specific genomic positions. Required information for each genomic position includes chromosome, position, allelic fraction, and coverage. Required column names are chr, pos, af, and cov
`min_coverage`	minimum number of reads for considering a pileup position valid (default=20)
`min_required_snps`	minimum number of snps to call beta for a segment (default=10)
`min_af_het_snps`	minimum allowed allelic fraction of a SNP genomic position (default=0.2)
`max_af_het_snps`	maximum allowed allelic fraction of a SNP genomic position (default=0.8)

`n_digits`	number of digits in the output table (default=3)
`n_cores`	number of available cores for computation (default=1)
`plot_stats`	plot summary statistics of the computed beta table (default=F)
`debug`	return extra columns for debugging (default=F)

Value

A data.frame that extends input seg_tb with columns beta, nsnp, cov, n_beta. Moreover, CLONETv2 renames colums of seg_tb as sample, chr, start, end, XYZ, log2, with XYZ being the original name of column five As for seg_tb, each raw of the output table represents a genomic segments. For each raw, the value of beta is the proportion of neutral reads in the segment, while nsnp and cov represents respectively the number of informative SNPs and the mean coverage of the given segment. The value n_beta is the proportion of neutral reads in the normal sample. The value of n_beta should be 1 as in normal samples parental chromosomes are equally represented. Values lower than 1 of n_beta could indicate the presence of germline CNVs or sequencing errors.

Author(s)

Davide Prandi, Alessandro Romanel

Examples

```
## Compue beta table with default parameters
bt_toy <- compute_beta_table(seg_tb_toy, pileup_tumor_toy, pileup_normal_toy)
```

`compute_dna_admixture` *Function to compute DNA admixture of a tumor sample from the associatd beta table and ploidy table*

Description

This function takes a beta table and the associated ploidy table and computes DNA admixture.

Usage

```
compute_dna_admixture(beta_table, ploidy_table, min_required_snps = 10,
  min_coverage = 20, error_tb = error_table, library_type = "WES",
  n_digits = 3, n_cores = 1, debug = F)
```

Arguments

`beta_table`	data.frame formatted as the output of function `compute_beta_table`
`ploidy_table`	data.frame formatted as the output of function `compute_ploidy`
`min_required_snps`	minimum number of informative snps in a segment valid for computing ploidy (default=10)

`min_coverage`	minimum coverage of a segment valid for computing ploidy (default=20)
`error_tb`	data.frame that reports for each combination of coverage and number informative SNPs the expected estimation error around beta. The data.frame error_tb must contains 3 columns: **mean.cov** mean coverage **n.info.snps** number of informative SNPs **adm.estimation.error** estimated error on computed beta on a segment with coverage mean.cov and n.info.snps informative SNPs Package CLONETv2 have built in error_tb named error_table (default=error_table)
`library_type`	WES, WGS (default=WES)
`n_digits`	number of digits in the output table (default=3)
`n_cores`	number of available cores for computation (default=1)
`debug`	return extra columns for debugging (default=F)

Value

A data.frame with two columns: sample that corresponds to column sample of the input beta_table, and amd that represent the fraction of estimated DNA admixture

Author(s)

Davide Prandi

Examples

```
## Compute admixture table with default parameters
adm_table_toy <- compute_dna_admixture(beta_table = bt_toy, ploidy_table = pl_table_toy)
```

`compute_ploidy` *Function to compute ploidy from a beta table.*

Description

This function takes the beta table of a tumor sample and returns its ploidy.

Usage

```
compute_ploidy(beta_table, max_homo_dels_fraction = 0.01,
  beta_limit_for_neutral_reads = 0.9, min_coverage = 20,
  min_required_snps = 10, library_type = "WES", n_digits = 3,
  n_cores = 1)
```

Arguments

`beta_table`	data.frame formatted as the output of function `compute_beta_table`
`max_homo_dels_fraction`	estimated maximum proportion of genomic segments corresponding to an homozygous deletion (default=0.01)
`beta_limit_for_neutral_reads`	minimum beta value of a segment valid for computing ploidy (default=0.90)
`min_coverage`	minimum coverage of a segment valid for computing ploidy (default=20)
`min_required_snps`	minimum number of informative snps in a segment valid for computing ploidy (default=10)
`library_type`	WES, WGS (default=WES)
`n_digits`	number of digits in the output table (default=3)
`n_cores`	number of available cores for computation (default=1)

Value

A data.frame with two columns: sample that corresponds to column sample of the input beta_table, and ploidy computed

Author(s)

Davide Prandi

Examples

```
## Compute ploidy table with default parameters
pl_table_toy <- compute_ploidy(bt_toy)
```

compute_scna_clonality_table

Function to compute clonality of somatic copy number data

Description

This function takes the beta table of a tumor sample together with the associated ploidy and admixtures tables and computes the clonality of each segment in the beta table.

Usage

```
compute_scna_clonality_table(beta_table, ploidy_table, admixture_table,
  error_tb = error_table, clonality_threshold = 0.85,
  beta_threshold = 0.9, n_digits = 3, n_cores = 1, debug = F)
```

Arguments

`beta_table`	data.frame formatted as the output of function `compute_beta_table`
`ploidy_table`	data.frame formatted as the output of function `compute_ploidy`
`admixture_table`	data.frame formatted as the output of function `compute_dna_admixture`
`error_tb`	data.frame that reports for each combination of coverage and number informative SNPs the expected estimation error around beta. The data.frame error_tb must contains 3 columns: **mean.cov** mean coverage; **n.info.snps** number of informative SNPs; **adm.estimation.error** estimated error on computed beta on a segment with coverage mean.cov and n.info.snps informative SNPs. Package CLONETv2 have built in error_tb named error_table (default=error_table)
`clonality_threshold`	threshold to discretize continuous clonality value (default=0.85)
`beta_threshold`	threshold on beta value to determine clonality direction (default=0.90)
`n_digits`	number of digits in the output table (default=3)
`n_cores`	number of cores (default=1)
`debug`	return extra columns for debugging (default=F)

Value

A data.frame that extends input beta_table with columns

clonality estimated fraction of tumor cell with log2 copy number

clonality.min minum estimated fraction of tumor cell with log2 copy number

clonality.max minum estimated fraction of tumor cell with log2 copy number

clonality.status discretized clonality status into five values: *clonal*, large majority of the tumor cells has the same copy number; *subclonal*, not all the tumor cells has the same copy number; *not.analysed*, is is not possible to determine clonality; *uncertain.clonal* and *uncertain.subclonal* correspond respectively to *clonal* and *subclonal* populations but with less reliable clonality estimate

Author(s)

Davide Prandi

Examples

```
## Compute clonality table with default parameters
scna_clonality_table_toy <- compute_scna_clonality_table(beta_table = bt_toy,
  ploidy_table = pl_table_toy, admixture_table = adm_table_toy)
```

`compute_snv_clonality` *Function to compute clonality of SNVs*

Description

This function takes as input the genomic position of a SNVs and computes the percentage of genomic homogeneus cells harboring the mutation.

Usage

```
compute_snv_clonality(sample_id, snv_read_count, beta_table, ploidy_table,
  admixture_table, error_tb = error_table, error_rate = 0.05,
  n_digits = 3, n_cores = 1, annotation_style = "VEP", debug = F)
```

Arguments

`sample_id`	the id of the analyzed sample. It must be the same value reported in column sample of tables beta_table, ploidy_table, and admixture_table
`snv_read_count`	data.frame reporting in each row the genomic coordinates of an SNV together with number of reference and alternative reads covering the position in columns rc_ref_tumor and rc_alt_tumor, respectively. See parameter annotation_style for details about column names
`beta_table`	data.frame formatted as the output of function `compute_beta_table`
`ploidy_table`	data.frame formatted as the output of function `compute_ploidy`
`admixture_table`	data.frame formatted as the output of function `compute_dna_admixture`
`error_tb`	data.frame that reports for each combination of coverage and number informative SNPs the expected estimation error around beta. The data.frame error_tb must contains 3 columns: **mean.cov** mean coverage; **n.info.snps** number of informative SNPs; **adm.estimation.error** estimated error on computed beta on a segment with coverage mean.cov and n.info.snps informative SNPs. Package CLONETv2 have built in error_tb named error_table (default=error_table)
`error_rate`	expected fraction of SNV positions with outlier variant allelic fraction (default=0.05)
`n_digits`	number of digits in the output table (default=3)
`n_cores`	number of cores (default=1)
`annotation_style`	a string that corresponds to the format of the columns that describe the genomic coordinates of a SNV. Accepted values are VEP and MAF. VEP annotation describes genomic coordinates with a single column named Location. MAF format has columns Chromosome, Start_position, and End_position for each aberrant position
`debug`	return extra columns for debugging (default=F)

Value

A data.frame that extends input table snv_read_count with columns sample, cnA, cnB, t_af, t_af_corr, SNV.clonality, and SNV.clonality.status. Columns cnA and cnB report the allele specific copy number of the genomic segment containing the SNV position. Columns t_af and t_af_corr are respectively raw and ploidy/purity adjusted tumor varian allelic fractions. SNV.clonality reports the percentage of tumor cells harboring the SNV and with allele specific copy number cnA and cnB. SNV.clonality.status column lists dicretized SNV.clonality values. Discrete states are clonal, uncertain.clonal, uncertain.subclonal, and subclonal based in threshold automatically computed on the SNV.clonality values. Empty SNV.clonality.status of an SNV indicates that clonality cannot be assessed.

Author(s)

Davide Prandi, Tarcisio Fedrizzi

Examples

```
## Compute SNVs clonality
snv_clonality_table_toy <- compute_snv_clonality("toy_sample",
  snv_reads_toy, bt_toy, pl_table_toy, adm_table_toy)
```

---

error_table *Beta estimation error.*

Description

A precomputed table reporting for different combinations of coverage and number of informative SNPs the expected error of the beta value computed by function `compute_beta_table`.

Usage

```
error_table
```

Format

A data frame column names mean.cov, n.info.snps, and adm.estimation.error

mean.cov genomic segment coverage

n.info.snps number of informative SNPs

adm.estimation.error expected error on beta estimate

`pileup_normal_toy` *Toy example of normal pileup data.*

Description

Toy example of normal pileup data.

Usage

```
pileup_normal_toy
```

Format

An object of class `data.frame` with 816 rows and 11 columns.

`pileup_tumor_toy` *Toy example of tumor pileup data.*

Description

Toy example of tumor pileup data.

Usage

```
pileup_tumor_toy
```

Format

An object of class `data.frame` with 816 rows and 11 columns.

`pl_table_toy` *Toy example of ploidy table.*

Description

Toy example of ploidy table.

Usage

```
pl_table_toy
```

Format

An object of class `data.frame` with 1 rows and 2 columns.

`scna_clonality_table_toy` *Toy example of clonality table of somatic copy number.*

Description

Toy example of clonality table of somatic copy number.

Usage

```
scna_clonality_table_toy
```

Format

An object of class `data.frame` with 4 rows and 25 columns.

`seg_tb_toy` *Toy example of segmetd data.*

Description

Toy example of segmetd data.

Usage

```
seg_tb_toy
```

Format

An object of class `data.frame` with 4 rows and 6 columns.

`snv_clonality_table_toy` *Toy example of snv clonality table.*

Description

Toy example of snv clonality table.

Usage

```
snv_clonality_table_toy
```

Format

An object of class `data.frame` with 2 rows and 78 columns.

`snv_reads_toy` *Toy example of snv data.*

Description

Toy example of snv data.

Usage

```
snv_reads_toy
```

Format

An object of class `data.frame` with 2 rows and 71 columns.

Index