Package 'SCOPE'

July 10, 2025

Type Package

Title A normalization and copy number estimation method for single-cell DNA sequencing

Version 1.20.0

Author Rujin Wang, Danyu Lin, Yuchao Jiang

Maintainer Rujin Wang <rujin@email.unc.edu>

Description Whole genome single-cell DNA sequencing (scDNA-seq) enables characterization of copy number profiles at the cellular level. This circumvents the averaging effects associated with bulk-tissue sequencing and has increased resolution yet decreased ambiguity in deconvolving cancer subclones and elucidating cancer evolutionary history. ScDNA-seq data is, however, sparse, noisy, and highly variable even within a homogeneous cell population, due to the biases and artifacts that are introduced during the library preparation and sequencing procedure. Here, we propose SCOPE, a normalization and copy number estimation method for scDNA-seq data. The distinguishing features of SCOPE include: (i) utilization of cell-specific Gini coefficients for quality controls and for identification of normal/diploid cells, which are further used as negative control samples in a Poisson latent factor model for normalization; (ii) modeling of GC content bias using an expectation-maximization algorithm embedded in the Poisson generalized linear models, which accounts for the different copy number states along the genome; (iii) a cross-sample iterative segmentation procedure to identify breakpoints that are shared across cells from the same genetic background.

Depends R (>= 3.6.0), GenomicRanges, IRanges, Rsamtools, GenomeInfoDb, BSgenome.Hsapiens.UCSC.hg19

Imports stats, grDevices, graphics, utils, DescTools, RColorBrewer, gplots, foreach, parallel, doParallel, DNAcopy, BSgenome, Biostrings, BiocGenerics, S4Vectors

Suggests knitr, rmarkdown, WGSmapp, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Mmusculus.UCSC.mm10, testthat (>= 2.1.0)

VignetteBuilder knitr

biocViews SingleCell, Normalization, CopyNumberVariation, Sequencing, WholeGenome, Coverage, Alignment, QualityControl, DataImport, DNASeq

License GPL-2

LazyData true

RoxygenNote 6.1.1

Encoding UTF-8

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

git_branch RELEASE_3_21

git_last_commit bda8fef

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-07-09

Contents

`coverageObj.scopeDemo` *Pre-stored coverageObj.scope data for demonstration purposes*

Description

Pre-stored coverageObj.scope data for demonstration purposes

Usage

```
coverageObj.scopeDemo
```

Format

Pre-computed using whole genome sequencing data of three single cells from 10X Genomics Single-Cell CNV solution

---

`get_bam_bed` *Get bam file directories, sample names, and whole genomic bins*

Description

Get bam file directories, sample names, and whole genomic bins from .bed file

Usage

```
get_bam_bed(bamdir, sampname, hgref = "hg19", resolution = 500,
            sex = FALSE)
```

Arguments

`bamdir`	vector of the directory of a bam file. Should be in the same order as sample names in `sampname`.
`sampname`	vector of sample names. Should be in the same order as bam directories in `bamdir`.
`hgref`	reference genome. This should be 'hg19', 'hg38' or 'mm10'. Default is human genome `hg19`.
`resolution`	numeric value of fixed bin-length. Default is `500`. Unit is "kb".
`sex`	logical, whether to include sex chromosomes. Default is `FALSE`.

Value

A list with components

`bamdir`	A vector of bam directories
`sampname`	A vector of sample names
`ref`	A GRanges object specifying whole genomic bin positions

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
library(WGSmapp)
library(BSgenome.Hsapiens.UCSC.hg38)
bamfolder <- system.file('extdata', package = 'WGSmapp')
bamFile <- list.files(bamfolder, pattern = '*.dedup.bam$')
bamdir <- file.path(bamfolder, bamFile)
sampname_raw <- sapply(strsplit(bamFile, '.', fixed = TRUE), '[', 1)
bambedObj <- get_bam_bed(bamdir = bamdir, sampname = sampname_raw,
                         hgref = "hg38")
bamdir <- bambedObj$bamdir
sampname_raw <- bambedObj$sampname
ref_raw <- bambedObj$ref
```

---

get_coverage_scDNA *Get read coverage from single-cell DNA sequencing*

Description

Get read coverage for each genomic bin across all single cells from scDNA-seq. Blacklist regions, such as segmental duplication regions and gaps near telomeres/centromeres will be masked prior to getting coverage.

Usage

```
get_coverage_scDNA(bambedObj, mapqthres, seq, hgref = "hg19")
```

Arguments

bambedObj	object returned from get_bam_bed
mapqthres	mapping quality threshold of reads
seq	the sequencing method to be used. This should be either 'paired-end' or 'single-end'
hgref	reference genome. This should be 'hg19', 'hg38' or 'mm10'. Default is human genome hg19.

Value

Y	Read depth matrix

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
library(WGSmapp)
library(BSgenome.Hsapiens.UCSC.hg38)
bamfolder <- system.file('extdata', package = 'WGSmapp')
bamFile <- list.files(bamfolder, pattern = '*.dedup.bam$')
bamdir <- file.path(bamfolder, bamFile)
sampname_raw <- sapply(strsplit(bamFile, '.', fixed = TRUE), '[', 1)
bambedObj <- get_bam_bed(bamdir = bamdir,
                         sampname = sampname_raw,
                         hgref = "hg38")

# Getting raw read depth
coverageObj <- get_coverage_scDNA(bambedObj,
                                  mapqthres = 40,
                                  seq = 'paired-end',
                                  hgref = "hg38")
Y_raw <- coverageObj$Y
```

---

get_gc *Compute GC content*

Description

Compute GC content for each bin

Usage

```
get_gc(ref, hgref = "hg19")
```

Arguments

ref	GRanges object returned from get_bam_bed
hgref	reference genome. This should be 'hg19', 'hg38' or 'mm10'. Default is human genome hg19.

Value

gc	Vector of GC content for each bin/target

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
## Not run:
library(WGSmapp)
library(BSgenome.Hsapiens.UCSC.hg38)
bamfolder <- system.file('extdata', package = 'WGSmapp')
bamFile <- list.files(bamfolder, pattern = '*.dedup.bam$')
bamdir <- file.path(bamfolder, bamFile)
sampname_raw <- sapply(strsplit(bamFile, '.', fixed = TRUE), '[', 1)
bambedObj <- get_bam_bed(bamdir = bamdir,
                         sampname = sampname_raw,
                         hgref = "hg38")
bamdir <- bambedObj$bamdir
sampname_raw <- bambedObj$sampname
ref_raw <- bambedObj$ref

gc <- get_gc(ref_raw, hgref = "hg38")

## End(Not run)
```

---

get_gini *Compute Gini coefficients for single cells*

Description

Gini index is defined as two times the area between the Lorenz curve and the diagonal.

Usage

```
get_gini(Y)
```

Arguments

Y	raw read depth matrix after quality control procedure

Value

Gini	Vector of Gini coefficients for single cells from scDNA-seq

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Gini <- get_gini(Y_sim)
```

get_mapp *Compute mappability*

Description

Compute mappability for each bin. Note that scDNA sequencing is whole-genome amplification and the mappability score is essential to determine variable binning method. Mappability track for 100-mers on the GRCh37/hg19 human reference genome from ENCODE is pre-saved. Compute the mean of mappability scores that overlapped reads map to bins, weighted by the width of mappability tracks on the genome reference. Use liftOver utility to calculate mappability for hg38, which is pre-saved as well. For mm10, there are two workarounds: 1) set all mappability to 1 to avoid extensive computation; 2) adopt QC procedures based on annotation results, e.g., filter out bins within black list regions, which generally have low mappability.

Usage

```
get_mapp(ref, hgref = "hg19")
```

Arguments

ref	GRanges object returned from get_bam_bed
hgref	reference genome. This should be 'hg19', 'hg38' or 'mm10'. Default is human genome hg19.

Value

mapp	Vector of mappability for each bin/target

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
## Not run:
library(WGSmapp)
library(BSgenome.Hsapiens.UCSC.hg38)
bamfolder <- system.file('extdata', package = 'WGSmapp')
bamFile <- list.files(bamfolder, pattern = '*.dedup.bam$')
bamdir <- file.path(bamfolder, bamFile)
sampname_raw <- sapply(strsplit(bamFile, '.', fixed = TRUE), '[', 1)
bambedObj <- get_bam_bed(bamdir = bamdir,
                         sampname = sampname_raw,
                         hgref = "hg38")
bamdir <- bambedObj$bamdir
sampname_raw <- bambedObj$sampname
ref_raw <- bambedObj$ref
```

```
mapp <- get_mapp(ref_raw, hgref = "hg38")

## End(Not run)
```

`get_samp_QC` *Get QC metrics for single cells*

Description

Perform QC step on single cells.

Usage

```
get_samp_QC(bambedObj)
```

Arguments

`bambedObj`	object returned from `get_bam_bed`

Value

`QCmetric`	A matrix containing total number/proportion of reads, total number/proportion of mapped reads, total number/proportion of mapped non-duplicate reads, and number/proportion of reads with mapping quality greater than 20

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
library(WGSmapp)
library(BSgenome.Hsapiens.UCSC.hg38)
bamfolder <- system.file('extdata', package = 'WGSmapp')
bamFile <- list.files(bamfolder, pattern = '*.dedup.bam$')
bamdir <- file.path(bamfolder, bamFile)
sampname_raw <- sapply(strsplit(bamFile, '.', fixed = TRUE), '[', 1)
bambedObj <- get_bam_bed(bamdir = bamdir,
                         sampname = sampname_raw,
                         hgref = "hg38")
QCmetric_raw = get_samp_QC(bambedObj)
```

iCN_sim — *A post cross-sample segmentation integer copy number matrix returned by SCOPE in the demo*

Description

A post cross-sample segmentation integer copy number matrix returned by SCOPE in the demo

Usage

```
iCN_sim
```

Format

A post cross-sample segmentation integer copy number matrix of five toy cells returned by SCOPE

initialize_ploidy — *Ploidy pre-initialization*

Description

Pre-estimate ploidies across all cells

Usage

```
initialize_ploidy(Y, Yhat, ref, maxPloidy = 6, minPloidy = 1.5,
                  minBinWidth = 5, SoS.plot = FALSE)
```

Arguments

Y	raw read depth matrix after quality control procedure
Yhat	normalized read depth matrix
ref	GRanges object after quality control procedure
maxPloidy	maximum ploidy candidate. Defalut is 6
minPloidy	minimum ploidy candidate. Defalut is 1.5
minBinWidth	the minimum number of bins for a changed segment. Defalut is 5
SoS.plot	logical, whether to generate ploidy pre-estimation plots. Default is FALSE.

Value

ploidy.SoS	Vector of pre-estimated ploidies for each cell

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Gini <- get_gini(Y_sim)

# first-pass CODEX2 run with no latent factors
normObj.sim <- normalize_codex2_ns_noK(Y_qc = Y_sim,
                                       gc_qc = ref_sim$gc,
                                       norm_index = which(Gini<=0.12))
Yhat.noK.sim <- normObj.sim$Yhat
beta.hat.noK.sim <- normObj.sim$beta.hat
fGC.hat.noK.sim <- normObj.sim$fGC.hat
N.sim <- normObj.sim$N

# Ploidy initialization
ploidy.sim <- initialize_ploidy(Y = Y_sim,
                                Yhat = Yhat.noK.sim,
                                ref = ref_sim)
ploidy.sim
```

---

initialize_ploidy_group

Group-wise ploidy pre-initialization

Description

Pre-estimate ploidies across cells with shared clonal memberships

Usage

```
initialize_ploidy_group(Y, Yhat, ref, groups,
                        maxPloidy = 6, minPloidy = 1.5,
                        minBinWidth = 5, SoS.plot = FALSE)
```

Arguments

Y	raw read depth matrix after quality control procedure
Yhat	normalized read depth matrix
ref	GRanges object after quality control procedure
groups	clonal membership labels for each cell
maxPloidy	maximum ploidy candidate. Defalut is 6
minPloidy	minimum ploidy candidate. Defalut is 1.5
minBinWidth	the minimum number of bins for a changed segment. Defalut is 5
SoS.plot	logical, whether to generate ploidy pre-estimation plots. Default is FALSE.

Value

ploidy.SoS	Vector of group-wise pre-estimated ploidies for each cell

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Gini <- get_gini(Y_sim)

# first-pass CODEX2 run with no latent factors
normObj.sim <- normalize_codex2_ns_noK(Y_qc = Y_sim,
                                       gc_qc = ref_sim$gc,
                                       norm_index = which(Gini<=0.12))
Yhat.noK.sim <- normObj.sim$Yhat
beta.hat.noK.sim <- normObj.sim$beta.hat
fGC.hat.noK.sim <- normObj.sim$fGC.hat
N.sim <- normObj.sim$N

# Group-wise ploidy initialization
clones <- c("normal", "tumor1", "normal", "tumor1", "tumor1")
ploidy.sim.group <- initialize_ploidy_group(Y = Y_sim, Yhat = Yhat.noK.sim,
                                ref = ref_sim, groups = clones)
ploidy.sim.group
```

---

normalize_codex2_ns_noK

Normalization of read depth without latent factors under the case-control setting

---

Description

Assuming that all reads are from diploid regions, fit a Poisson generalized linear model to normalize the raw read depth data from single-cell DNA sequencing, without latent factors under the case-control setting.

Usage

```
normalize_codex2_ns_noK(Y_qc, gc_qc, norm_index)
```

Arguments

Y_qc	read depth matrix after quality control
gc_qc	vector of GC content for each bin after quality control
norm_index	indices of normal/diploid cells

Value

A list with components

Yhat	A list of normalized read depth matrix
fGC.hat	A list of estimated GC content bias matrix
beta.hat	A list of estimated bin-specific bias vector
N	A vector of cell-specific library size factor, which is computed from the genome-wide read depth data

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Gini <- get_gini(Y_sim)
# first-pass CODEX2 run with no latent factors
normObj.sim <- normalize_codex2_ns_noK(Y_qc = Y_sim,
                                       gc_qc = ref_sim$gc,
                                       norm_index = which(Gini<=0.12))
```

---

normalize_scope *Normalization of read depth with latent factors using Expectation-Maximization algorithm under the case-control setting*

Description

Fit a Poisson generalized linear model to normalize the raw read depth data from single-cell DNA sequencing, with latent factors under the case-control setting. Model GC content bias using an expectation-maximization algorithm, which accounts for the different copy number states.

Usage

```
normalize_scope(Y_qc, gc_qc, K, norm_index, T, ploidyInt,
                beta0, minCountQC = 20)
```

Arguments

Y_qc	read depth matrix after quality control
gc_qc	vector of GC content for each bin after quality control
K	Number of latent Poisson factors
norm_index	indices of normal/diploid cells

T	a vector of integers indicating number of CNV groups. Use BIC to select optimal number of CNV groups. If T = 1, assume all reads are from normal regions so that EM algorithm is not implemented. Otherwise, we assume there is always a CNV group of heterozygous deletion and a group of null region. The rest groups are representative of different duplication states.
ploidyInt	a vector of initialized ploidy return from initialize_ploidy. Users are also allowed to provide prior-knowledge ploidies as the input and to manually tune a few cells that have poor fitting
beta0	a vector of initialized bin-specific biases returned from CODEX2 without latent factors
minCountQC	the minimum read coverage required for normalization and EM fitting. Defalut is 20

Value

A list with components

Yhat	A list of normalized read depth matrix with EM
alpha.hat	A list of absolute copy number matrix
fGC.hat	A list of EM estimated GC content bias matrix
beta.hat	A list of EM estimated bin-specific bias vector
g.hat	A list of estimated Poisson latent factor
h.hat	A list of estimated Poisson latent factor
AIC	AIC for model selection
BIC	BIC for model selection
RSS	RSS for model selection
K	Number of latent Poisson factors

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Gini <- get_gini(Y_sim)

# first-pass CODEX2 run with no latent factors
normObj.sim <- normalize_codex2_ns_noK(Y_qc = Y_sim,
                                       gc_qc = ref_sim$gc,
                                       norm_index = which(Gini<=0.12))
Yhat.noK.sim <- normObj.sim$Yhat
beta.hat.noK.sim <- normObj.sim$beta.hat
fGC.hat.noK.sim <- normObj.sim$fGC.hat
N.sim <- normObj.sim$N

# Ploidy initialization
ploidy.sim <- initialize_ploidy(Y = Y_sim,
```

```
                                    Yhat = Yhat.noK.sim,
                                    ref = ref_sim)
ploidy.sim

normObj.scope.sim <- normalize_scope(Y_qc = Y_sim, gc_qc = ref_sim$gc,
                                     K = 1, ploidyInt = ploidy.sim,
                                     norm_index = which(Gini<=0.12), T = 1:5,
                                     beta0 = beta.hat.noK.sim)
Yhat.sim <- normObj.scope.sim$Yhat[[which.max(normObj.scope.sim$BIC)]]
fGC.hat.sim <- normObj.scope.sim$fGC.hat[[which.max(normObj.scope.sim$BIC)]]
```

`normalize_scope_foreach` *Normalization of read depth with latent factors using Expectation-Maximization algorithm under the case-control setting in parallel*

Description

Fit a Poisson generalized linear model to normalize the raw read depth data from single-cell DNA sequencing, with latent factors under the case-control setting. Model GC content bias using an expectation-maximization algorithm, which accounts for the different copy number states.

Usage

```
normalize_scope_foreach(Y_qc, gc_qc, K, norm_index, T,
    ploidyInt, beta0, minCountQC = 20, nCores = NULL)
```

Arguments

`Y_qc`	read depth matrix after quality control
`gc_qc`	vector of GC content for each bin after quality control
`K`	Number of latent Poisson factors
`norm_index`	indices of normal/diploid cells
`T`	a vector of integers indicating number of CNV groups. Use BIC to select optimal number of CNV groups. If `T = 1`, assume all reads are from normal regions so that EM algorithm is not implemented. Otherwise, we assume there is always a CNV group of heterozygous deletion and a group of null region. The rest groups are representative of different duplication states.
`ploidyInt`	a vector of initialized ploidy return from `initialize_ploidy`. Users are also allowed to provide prior-knowledge ploidies as the input and to manually tune a few cells that have poor fitting
`beta0`	a vector of initialized bin-specific biases returned from CODEX2 without latent factors
`minCountQC`	the minimum read coverage required for normalization and EM fitting. Defalut is `20`
`nCores`	number of cores to use. If `NULL`, number of cores is detected. Default is `NULL`.

Value

A list with components

Yhat	A list of normalized read depth matrix with EM
alpha.hat	A list of absolute copy number matrix
fGC.hat	A list of EM estimated GC content bias matrix
beta.hat	A list of EM estimated bin-specific bias vector
g.hat	A list of estimated Poisson latent factor
h.hat	A list of estimated Poisson latent factor
AIC	AIC for model selection
BIC	BIC for model selection
RSS	RSS for model selection
K	Number of latent Poisson factors

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Gini <- get_gini(Y_sim)

# first-pass CODEX2 run with no latent factors
normObj.sim <- normalize_codex2_ns_noK(Y_qc = Y_sim,
                                       gc_qc = ref_sim$gc,
                                       norm_index = which(Gini<=0.12))
Yhat.noK.sim <- normObj.sim$Yhat
beta.hat.noK.sim <- normObj.sim$beta.hat
fGC.hat.noK.sim <- normObj.sim$fGC.hat
N.sim <- normObj.sim$N

# Ploidy initialization
ploidy.sim <- initialize_ploidy(Y = Y_sim,
                            Yhat = Yhat.noK.sim,
                            ref = ref_sim)
ploidy.sim

# Specify nCores = 2 only for checking examples
normObj.scope.sim <- normalize_scope_foreach(Y_qc = Y_sim,
                        gc_qc = ref_sim$gc,
                        K = 1, ploidyInt = ploidy.sim,
                        norm_index = which(Gini<=0.12), T = 1:5,
                        beta0 = beta.hat.noK.sim, nCores = 2)
Yhat.sim <- normObj.scope.sim$Yhat[[which.max(normObj.scope.sim$BIC)]]
fGC.hat.sim <- normObj.scope.sim$fGC.hat[[which.max(normObj.scope.sim$BIC)]]
```

`normalize_scope_group` *Group-wise normalization of read depth with latent factors using Expectation-Maximization algorithm and shared clonal memberships*

Description

Fit a Poisson generalized linear model to normalize the raw read depth data from single-cell DNA sequencing, with latent factors and shared clonal memberships. Model GC content bias using an expectation-maximization algorithm, which accounts for clonal specific copy number states.

Usage

```
normalize_scope_group(Y_qc, gc_qc, K, norm_index, groups, T,
                      ploidyInt, beta0, minCountQC = 20)
```

Arguments

`Y_qc`	read depth matrix after quality control
`gc_qc`	vector of GC content for each bin after quality control
`K`	Number of latent Poisson factors
`norm_index`	indices of normal/diploid cells using group/clone labels
`groups`	clonal membership labels for each cell
`T`	a vector of integers indicating number of CNV groups. Use BIC to select optimal number of CNV groups. If `T = 1`, assume all reads are from normal regions so that EM algorithm is not implemented. Otherwise, we assume there is always a CNV group of heterozygous deletion and a group of null region. The rest groups are representative of different duplication states.
`ploidyInt`	a vector of group-wise initialized ploidy return from `initialize_ploidy_group`. Users are also allowed to provide prior-knowledge ploidies as the input and to manually tune a few cells/clones that have poor fitting
`beta0`	a vector of initialized bin-specific biases returned from CODEX2 without latent factors
`minCountQC`	the minimum read coverage required for normalization and EM fitting. Defalut is `20`

Value

A list with components

`Yhat`	A list of normalized read depth matrix with EM
`alpha.hat`	A list of absolute copy number matrix
`fGC.hat`	A list of EM estimated GC content bias matrix
`beta.hat`	A list of EM estimated bin-specific bias vector
`g.hat`	A list of estimated Poisson latent factor

h.hat	A list of estimated Poisson latent factor
AIC	AIC for model selection
BIC	BIC for model selection
RSS	RSS for model selection
K	Number of latent Poisson factors

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Gini <- get_gini(Y_sim)

# first-pass CODEX2 run with no latent factors
normObj.sim <- normalize_codex2_ns_noK(Y_qc = Y_sim,
                                       gc_qc = ref_sim$gc,
                                       norm_index = which(Gini<=0.12))
Yhat.noK.sim <- normObj.sim$Yhat
beta.hat.noK.sim <- normObj.sim$beta.hat
fGC.hat.noK.sim <- normObj.sim$fGC.hat
N.sim <- normObj.sim$N

# Group-wise ploidy initialization
clones <- c("normal", "tumor1", "normal", "tumor1", "tumor1")
ploidy.sim.group <- initialize_ploidy_group(Y = Y_sim, Yhat = Yhat.noK.sim,
                                ref = ref_sim, groups = clones)
ploidy.sim.group

normObj.scope.sim.group <- normalize_scope_group(Y_qc = Y_sim,
                                    gc_qc = ref_sim$gc,
                                    K = 1, ploidyInt = ploidy.sim.group,
                                    norm_index = which(clones=="normal"),
                                    groups = clones,
                                    T = 1:5,
                                    beta0 = beta.hat.noK.sim)
Yhat.sim.group <- normObj.scope.sim.group$Yhat[[which.max(
                                    normObj.scope.sim.group$BIC)]]
fGC.hat.sim.group <- normObj.scope.sim.group$fGC.hat[[which.max(
                                    normObj.scope.sim.group$BIC)]]
```

---

normObj.scopeDemo *Pre-stored normObj.scope data for demonstration purposes*

Description

Pre-stored normObj.scope data for demonstration purposes

Usage

```
normObj.scopeDemo
```

Format

Pre-computed by SCOPE using pre-stored data `Y_sim`

`perform_qc` *Quality control for cells and bins*

Description

Perform QC step on single cells and bins.

Usage

```
perform_qc(Y_raw, sampname_raw, ref_raw, QCmetric_raw,
        cov_thresh = 0, minCountQC = 20,
        mapq20_thresh = 0.3, mapp_thresh = 0.9,
        gc_thresh = c(20, 80), nMAD = 3)
```

Arguments

`Y_raw`	raw read count matrix returned from `get_coverage_scDNA`
`sampname_raw`	sample names for quality control returned from `get_bam_bed`
`ref_raw`	raw GRanges object with corresponding GC content and mappability for quality control returned from `get_bam_bed`
`QCmetric_raw`	a QC metric for single cells returned from `get_samp_QC`
`cov_thresh`	scalar variable specifying the lower bound of read count summation of each cell. Default is `0`
`minCountQC`	the minimum read coverage required for normalization and EM fitting. Defalut is `20`
`mapq20_thresh`	scalar variable specifying the lower threshold of proportion of reads with mapping quality greater than 20. Default is `0.3`
`mapp_thresh`	scalar variable specifying mappability of each genomic bin. Default is `0.9`
`gc_thresh`	vector specifying the lower and upper bound of GC content threshold for quality control. Default is `20-80`
`nMAD`	scalar variable specifying the number of MAD from the median of total read counts adjusted by library size for each cell. Default is `3`

Value

A list with components

Y	read depth matrix after quality control
sampname	sample names after quality control
ref	A GRanges object specifying whole genomic bin positions after quality control
QCmetric	A data frame of QC metric for single cells after quality control

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Y_raw <- coverageObj.scopeDemo$Y
sampname_raw <- rownames(QCmetric.scopeDemo)
ref_raw <- ref.scopeDemo
QCmetric_raw <- QCmetric.scopeDemo
qcObj <- perform_qc(Y_raw = Y_raw, sampname_raw = sampname_raw,
                ref_raw = ref_raw, QCmetric_raw = QCmetric_raw)
```

---

plot_EM_fit *Visualize EM fitting for each cell.*

Description

A pdf file containing EM fitting results and plots is generated.

Usage

```
plot_EM_fit(Y_qc, gc_qc, norm_index, T, ploidyInt, beta0,
                minCountQC = 20, filename)
```

Arguments

Y_qc	read depth matrix across all cells after quality control
gc_qc	vector of GC content for each bin after quality control
norm_index	indices of normal/diploid cells
T	a vector of integers indicating number of CNV groups. Use BIC to select optimal number of CNV groups. If T = 1, assume all reads are from normal regions so that EM algorithm is not implemented. Otherwise, we assume there is always a CNV group of heterozygous deletion and a group of null region. The rest groups are representative of different duplication states.
ploidyInt	a vector of initialized ploidy return from initialize_ploidy

beta0	a vector of initialized bin-specific biases returned from CODEX2 without latent factors
minCountQC	the minimum read coverage required for EM fitting. Defalut is 20
filename	the name of output pdf file

Value

pdf file with EM fitting results and two plots: log likelihood, and BIC versus the number of CNV groups.

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Gini <- get_gini(Y_sim)
# first-pass CODEX2 run with no latent factors
normObj.sim <- normalize_codex2_ns_noK(Y_qc = Y_sim,
                                       gc_qc = ref_sim$gc,
                                       norm_index = which(Gini<=0.12))
Yhat.noK.sim <- normObj.sim$Yhat
beta.hat.noK.sim <- normObj.sim$beta.hat
fGC.hat.noK.sim <- normObj.sim$fGC.hat
N.sim <- normObj.sim$N

# Ploidy initialization
ploidy.sim <- initialize_ploidy(Y = Y_sim,
                                Yhat = Yhat.noK.sim,
                                ref = ref_sim)
ploidy.sim

plot_EM_fit(Y_qc = Y_sim, gc_qc = ref_sim$gc,
        norm_index = which(Gini<=0.12), T = 1:7,
        ploidyInt = ploidy.sim,
        beta0 = beta.hat.noK.sim,
        filename = 'plot_EM_fit_demo.pdf')
```

---

plot_iCN *Plot post-segmentation copy number profiles of integer values*

Description

Show heatmap of inferred integer copy-number profiles by SCOPE with cells clustered by hierarchical clustering

Usage

```
plot_iCN(iCNmat, ref, Gini, annotation = NULL,
        plot.dendrogram = TRUE, show.names = FALSE, filename)
```

Arguments

iCNmat	inferred integer copy-number matrix by SCOPE, with each column being a cell and each row being a genomic bin
ref	GRanges object after quality control procedure
Gini	vector of Gini coefficients for each cell, with the same order as that of cells in columns of iCNmat
annotation	vector of annotation for each cell, with the same order as that of cells in columns of iCNmat. Default is NULL.
plot.dendrogram	logical, whether to plot the dendrogram. Default is TRUE.
show.names	logical, whether to show cell names by y axis. Default is FALSE.
filename	name of the output png file

Value

png file with integer copy-number profiles across single cells with specified annotations

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Gini <- get_gini(Y_sim)
plot_iCN(iCNmat = iCN_sim,
        ref = ref_sim,
        Gini = Gini,
        filename = 'plot_iCN_demo')
```

QCmetric.scopeDemo *Pre-stored QCmetric data for demonstration purposes*

Description

Pre-stored QCmetric data for demonstration purposes

Usage

```
QCmetric.scopeDemo
```

Format

Pre-computed using whole genome sequencing data of three single cells from 10X Genomics Single-Cell CNV solution

---

`ref.scopeDemo` *Pre-stored 500kb-size reference genome for demonstration purposes*

Description

Pre-stored 500kb-size reference genome for demonstration purposes

Usage

```
ref.scopeDemo
```

Format

Pre-computed using whole genome sequencing data with GC content and mappability scores

---

`ref_sim` *A reference genome in the toy dataset*

Description

A reference genome in the toy dataset

Usage

```
ref_sim
```

Format

A GRanges object with 1544 bins and 1 metadata column of GC content

segment_CBScs *Cross-sample segmentation*

Description

SCOPE offers a cross-sample Poisson likelihood-based recursive segmentation, enabling shared breakpoints across cells from the same genetic background.

Usage

```
segment_CBScs(Y, Yhat, sampname, ref, chr,
                    mode = "integer", max.ns)
```

Arguments

Y	raw read depth matrix after quality control procedure
Yhat	normalized read depth matrix
sampname	vector of sample names
ref	GRanges object after quality control procedure
chr	chromosome name. Make sure it is consistent with the reference genome.
mode	format of returned copy numbers. Only integer mode is supported for scDNA-seq data.
max.ns	a number specifying how many rounds of nested structure searching would be performed. Defalut is 0.

Value

A list with components

poolcall	Cross-sample CNV callings indicating shared breakpoints
finalcall	Final cross-sample segmented callset of CNVs with genotyping results
image.orig	A matrix giving logarithm of normalized z-scores
image.seg	A matrix of logarithm of estimated copy number over 2
iCN	A matrix of inferred integer copy number profiles

Author(s)

Rujin Wang <rujin@email.unc.edu>

Examples

```
Yhat.sim <- normObj.scopeDemo$Yhat[[which.max(normObj.scopeDemo$BIC)]]
segment_cs_chr1 <- segment_CBScs(Y = Y_sim, Yhat = Yhat.sim,
                        sampname = colnames(Y_sim),
                        ref = ref_sim, chr = 'chr1', max.ns = 1)
```

Y_sim *A read count matrix in the toy dataset*

Description

A read count matrix in the toy dataset

Usage

```
Y_sim
```

Format

A read count matrix with 1544 bins and 39 cells

Index