

# Package ‘cfdnakit’

May 17, 2024

**Title** Fragmen-length analysis package from high-throughput sequencing of cell-free DNA (cfDNA)

**Version** 1.2.0

**Description** This package provides basic functions for analyzing shallow whole-genome sequencing (~0.3X or more) of cell-free DNA (cfDNA). The package basically extracts the length of cfDNA fragments and aids the visualization of fragment-length information. The package also extract fragment-length information per non-overlapping fixed-sized bins and used it for calculating ctDNA estimation score (CES).

**License** GPL-3

**Encoding** UTF-8

**LazyData** FALSE

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**biocViews** CopyNumberVariation, Sequencing, WholeGenome

**BugReports** <https://github.com/Pitithat-pu/cfdnakit/issues>

**Imports** Biobase, dplyr, GenomicRanges, GenomeInfoDb, ggplot2, IRanges, magrittr, PSCBS, QDNAseq, Rsamtools, utils, S4Vectors, stats, rlang

**Depends** R (>= 4.3)

**Suggests** rmarkdown, knitr, roxygen2, BiocStyle

**VignetteBuilder** knitr

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

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** c461945

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-17

**Author** Pitithat Puranachot [aut, cre]  
(<https://orcid.org/0000-0002-6786-9240>)

**Maintainer** Pitithat Puranachot <[pitithat@gmail.com](mailto:pitithat@gmail.com)>

## Contents

|                              |           |
|------------------------------|-----------|
| cfdnakit-package             | 2         |
| calculate_CES_score          | 4         |
| call_cnv                     | 5         |
| create_blacklist_gr          | 6         |
| create_PoN                   | 6         |
| extract_insert_size          | 7         |
| filter_read_on_blacklist     | 8         |
| fragment_dist                | 8         |
| get_fragment_profile         | 9         |
| get_segment_byresolution     | 10        |
| get_solution_table           | 10        |
| get_zscore_profile           | 11        |
| GRCh2UCSCGRanges             | 12        |
| if_exist_baifile             | 12        |
| if_ucsc_chrformat            | 13        |
| make_density_table           | 13        |
| overlap_bin_with_segment     | 14        |
| plot_cnv_solution            | 14        |
| plot_distance_matrix         | 15        |
| plot_fragment_dist           | 16        |
| plot_sl_ratio                | 17        |
| plot_transformed_sl          | 17        |
| read_bamfile                 | 18        |
| read_PoN_files               | 19        |
| segmentByPSCB                | 20        |
| test_ysize_KolmogorovSmirnov | 21        |
| UCSC2GRChSampleBam           | 21        |
| util.bias_correct            | 22        |
| zscore_transform             | 22        |
| %>%                          | 23        |
| <b>Index</b>                 | <b>24</b> |

---

|                  |   |
|------------------|---|
| cfdnakit-package | <i>Fragmen-length analysis package from high-throughput sequencing of cell-free DNA (cfDNA)</i> |
|------------------|---|

---

## Description

This package provides basic functions for analyzing shallow whole-genome sequencing (~0.3X or more) of cell-free DNA (cfDNA). The package basically extracts the length of cfDNA fragments and aids the visualization of fragment-length information. The package also extract fragment-length information per non-overlapping fixed-sized bins and used it for calculating ctDNA estimation score (CES).

## Details

This package provides functions for analyzing using shallow whole-genome sequencing data (~0.3X or more) of circulating cell-free DNA (cfDNA). The aim is to estimate circulating tumor DNA using its characteristic short-fragmented cfDNA. The package extracts length of each cfDNA and assists the visualization of fragment-length distribution. A short-fragment ratio is calculated per non-overlapping fixed-sized bins. Genome-wide copy-number alteration is estimated by the short-fragmented cfDNA. The ctDNA estimation score (CES) comprehensively estimates the circulating tumor DNA based on the short-fragment analysis.

## Author(s)

Dr. rer. nat. Pitithat Puranachot

## Examples

```
library(cfdnakit)
## Reading in a bamfile
sample_bamfile = system.file("extdata",
                             "ex.plasma.bam",
                             package = "cfdnakit")
plasma_SampleBam = read_bamfile(sample_bamfile,
                               apply_blacklist = FALSE)

## Plot a fragment-length distribution of a sample
plot_fragment_dist(list("Plasma.Sample"=plasma_SampleBam))

## Plot a fragment-length distribution of two samples
control_RDS_file =
  system.file("extdata", "BH01_CHR15.SampleBam.rds",
             package = "cfdnakit")
### Load example SampleBam of Healthy cfDNA
control_bins =
  readRDS(control_RDS_file)

comparing_list = list("Healthy.cfDNA"=control_bins,
                    "Patient.1"=plasma_SampleBam)
plot_fragment_dist(comparing_list)

## Derived and plot genome-wide short-fragment cfDNA
patient.SampleFragment =
  get_fragment_profile(plasma_SampleBam,
                    sample_id = "Patient.1")
plot_sl_ratio(patient.SampleFragment)

## Derived and plot normalized short-fragment cfDNA
PoN_rdsfile = system.file(
  "extdata",
  "ex.PoN.rds",
  package = "cfdnakit")
```

```

        ## Loading example PoN data
PoN.profiles = readRDS(PoN_rdsfile)

sample_zscore =
  get_zscore_profile(patient.SampleFragment,
                    PoN.profiles)
sample_zscore_segment = segmentByPSCB(sample_zscore)
plot_transformed_sl(sample_zscore, sample_zscore_segment)

## Estimate circulating tumor DNA
calculate_CES_score(sample_zscore_segment)

```

---

calculate\_CES\_score    *Calculate CES Score from Segmentation*

---

### Description

Calculate CES Score from Segmentation

### Usage

```
calculate_CES_score(sample_segmentation)
```

### Arguments

```
sample_segmentation
  Segmentation Dataframe
```

### Value

Numeric; CES score

### Examples

```

### Loading example SampleBam file
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")
sample_bambin <- readRDS(example_file)
### Example PoN
PoN_rdsfile <- system.file("extdata", "ex.PoN.rds", package = "cfdnakit")
pon_profiles <- readRDS(PoN_rdsfile)
sample_profile <- get_fragment_profile(sample_bambin, sample_id = "Patient1")

sample_zscore <- get_zscore_profile(sample_profile, pon_profiles)

sample_zscore_segment <- segmentByPSCB(sample_zscore)

calculate_CES_score(sample_zscore_segment)

```

---

|          |   |
|----------|---|
| call_cnv | <i>Call Copy-number Variation from SLRatio and segmentation</i> |
|----------|---|

---

## Description

Call Copy-number Variation from SLRatio and segmentation

## Usage

```
call_cnv(
  sample_segmentation,
  sample_zscore,
  callChr = seq_len(22),
  tfs = c(0, 0.7),
  ploidy = c(1.5, 3),
  MaxCN = 4
)
```

## Arguments

|                     |  |
|---------------------|--|
| sample_segmentation | segmentation dataframe from segmentByPSCBS             |
| sample_zscore       | zscore dataframe                                       |
| callChr             | chromosome to analysis : Default c(1:22)               |
| tfs                 | range of fitting tumor fraction : Default c(0,0.8)     |
| ploidy              | range of fitting chromosomal ploidy : Default c(1.5,4) |
| MaxCN               | maximum copy-number : Default 4                        |

## Value

List of cnvcalling solutions

## Examples

```
### Loading example SampleBam file
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")
sample_bambin <- readRDS(example_file)
### Example PoN
PoN_rdsfile <- system.file("extdata", "ex.PoN.rds", package = "cfdnakit")
pon_profiles <- readRDS(PoN_rdsfile)
sample_profile <- get_fragment_profile(sample_bambin, sample_id = "Patient1")

sample_zscore <- get_zscore_profile(sample_profile, pon_profiles)

sample_zscore_segment <- segmentByPSCB(sample_zscore)

sample_cnv <- call_cnv(sample_zscore_segment, sample_zscore, tfs=c(0.1,0.3), ploidy=c(1.5,2), MaxCN=3)
```

```
plot_cnv_solution(sample_cnv, selected_solution = 1)
```

---

create\_blacklist\_gr     *Create Blacklist regions GRanges object*

---

**Description**

Create Blacklist regions GRanges object

**Usage**

```
create_blacklist_gr(blacklist_files)
```

**Arguments**

blacklist\_files  
Character; Filepath to file containing blacklist regions

**Value**

GRanges object of blacklist regions

---

create\_PoN             *Create Panel-of-Normal (PoN) object*

---

**Description**

Create Panel-of-Normal (PoN) object

**Usage**

```
create_PoN(list_rdsfiles)
```

**Arguments**

list\_rdsfiles     Character; a file contains paths to Profile.Rdata per line

**Value**

Null

**Examples**

```
healthy.1 <- system.file("extdata","ex.healthy1.rds",package = "cfdnakit")
healthy.2 <- system.file("extdata","ex.healthy2.rds",package = "cfdnakit")

path_to_PoN_txt <- paste0(system.file("extdata",package = "cfdnakit"),"/temp.reference_healthy.listfile")
fileConn<-file(path_to_PoN_txt)
writeLines(c(healthy.1,healthy.2), fileConn)
close(fileConn)

PoN.profiles <- create_PoN(path_to_PoN_txt)
file.remove(path_to_PoN_txt)
```

---

extract\_insert\_size     *Extract Insert size from SampleBam*

---

**Description**

Extract Insert size from SampleBam

**Usage**

```
extract_insert_size(readbam_bin, maximum_length = 600, minimum_length = 20)
```

**Arguments**

readbam\_bin     SampleBam Object

maximum\_length     Int; Maximum length of fragment. cfDNA fragment longer than this value will not be considered; Default 600

minimum\_length     Int; Minimum length of fragment. cfDNA fragment shorter than this value will not be considered; Default 20

**Value**

Numeric Vector; Insert size of given sample

**Examples**

```
### Loading example SampleBam file
example_file <- system.file("extdata","example_patientcfDNA_SampleBam.RDS",package = "cfdnakit")
sample_bambin <- readRDS(example_file)
extract_insert_size(sample_bambin)
### Extract only insert size of fragment having specific size
extract_insert_size(sample_bambin,maximum_length=500, minimum_length = 50)
```

---

filter\_read\_on\_blacklist

*Filter out reads on blacklist regions*

---

### Description

Filter out reads on blacklist regions

### Usage

```
filter_read_on_blacklist(sample_bin, blacklist_files = NULL, genome = "hg19")
```

### Arguments

|                 |  |
|-----------------|--|
| sample_bin      | SampleBam; Object from function read_bamfile                                   |
| blacklist_files | Character; Filepath to file containing blacklist regions                       |
| genome          | Character; Abbreviation of reference genome; Either hg19 or mm10. default:hg19 |

### Value

SampleBam after filtering out read on balck list regions

---

fragment\_dist

*Get insert-size distribution table*

---

### Description

Get insert-size distribution table

### Usage

```
fragment_dist(readbam_bin, maximum_length = 600, minimum_length = 20)
```

### Arguments

|                |  |
|----------------|--|
| readbam_bin    | SampleBam Object from function read_bamfile  |
| maximum_length | Int; Maximum length of fragment. cfDNA fragment longer than this value will not be considered; Default 600 |
| minimum_length | Int; Minimum length of fragment. cfDNA fragment shorter than this value will not be considered; Default 20 |

### Value

Distribution table of fragment length

---

get\_fragment\_profile *Getting fragment-length information*

---

## Description

Getting fragment-length information

## Usage

```
get_fragment_profile(  
  readbam_bin,  
  sample_id,  
  genome = "hg19",  
  short_range = c(100, 150),  
  long_range = c(151, 250),  
  maximum_length = 600,  
  minimum_length = 20  
)
```

## Arguments

|                |  |
|----------------|--|
| readbam_bin    | SampleBam Object   |
| sample_id      | Character; Given sample ID   |
| genome         | abbreviation of reference genome; namely hg19, mm10. default:hg19  |
| short_range    | Vector of 2 Int; Range of fragment length to be defined as short fragment; Default c(100,150)              |
| long_range     | Vector of 2 Int; Range of fragment length to be defined as long fragment; Default c(151,250)               |
| maximum_length | Int; Maximum length of fragment. cfDNA fragment longer than this value will not be considered; Default 600 |
| minimum_length | Int; Minimum length of fragment. cfDNA fragment shorter than this value will not be considered; Default 20 |

## Value

SampleFragment Object; Fragment length information for quality check and downstream analysis per bin and summary of sample

## Examples

```
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfDNAkit")  
sample_bam_bin <- readRDS(example_file)  
sample_profile <- get_fragment_profile(sample_bam_bin, sample_id = "Patient1")
```



## Examples

```
##'  
### Loading example SampleBam file  
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")  
sample_bambin <- readRDS(example_file)  
### Example PoN  
PoN_rdsfile <- system.file("extdata", "ex.PoN.rds", package = "cfdnakit")  
pon_profiles <- readRDS(PoN_rdsfile)  
sample_profile <- get_fragment_profile(sample_bambin, sample_id = "Patient1")  
  
sample_zscore <- get_zscore_profile(sample_profile, pon_profiles)  
  
sample_zscore_segment <- segmentByPSCB(sample_zscore)  
  
sample_cnv <- call_cnv(sample_zscore_segment, sample_zscore, tfs=c(0.1,0.3), ploidy=c(1.5,2), MaxCN=3)  
get_solution_table(sample_cnv)
```

---

get\_zscore\_profile      *Transform SLRatio with PoN Fragment profile*

---

## Description

Transform SLRatio with PoN Fragment profile

## Usage

```
get_zscore_profile(fragment_profile, pon_profile)
```

## Arguments

```
fragment_profile      Sample Profile  
pon_profile          PoN Profiles
```

## Value

Dataframe of robust transformed SLratio

## Examples

```
### Loading example SampleBam file  
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")  
sample_bambin <- readRDS(example_file)  
  
### Example PoN  
PoN_rdsfile <- system.file("extdata", "ex.PoN.rds", package = "cfdnakit")  
pon_profiles <- readRDS(PoN_rdsfile)  
sample_profile <- get_fragment_profile(sample_bambin, sample_id = "Patient1")
```

```
sample_zscore <- get_zscore_profile(sample_profile,pon_profiles)
sample_zscore_segment <- segmentByPSCB(sample_zscore)
```

---

GRCh2UCSCGRanges      *Convert GRCh chromosome format to UCSC style*

---

**Description**

Convert GRCh chromosome format to UCSC style

**Usage**

```
GRCh2UCSCGRanges(which)
```

**Arguments**

which                  GRanges object;

**Value**

GRanges; GRanges after chromosome format conversion

---

if\_exist\_baifile      *Check if bai file exist from given bam*

---

**Description**

Check if bai file exist from given bam

**Usage**

```
if_exist_baifile(bamfile)
```

**Arguments**

bamfile                Character; Path to sample bamfile

**Value**

Boolean if the bai file exist

---

|                   |   |
|-------------------|---|
| if_ucsc_chrformat | <i>Check UCSC chromosomes format for input bam file</i> |
|-------------------|---|

---

**Description**

Check UCSC chromosomes format for input bam file

**Usage**

```
if_ucsc_chrformat(bamfile_path)
```

**Arguments**

bamfile\_path    Character; Path to sample bamfile

**Value**

Boolean; if the input bam file is UCSC format, chr prefix

---

|                    |   |
|--------------------|---|
| make_density_table | <i>Make Fragment-length density table</i> |
|--------------------|---|

---

**Description**

Make Fragment-length density table

**Usage**

```
make_density_table(readbam_bin, minimum_length, maximum_length)
```

**Arguments**

readbam\_bin    List; A list containing SampleBam object/objects from the read\_bamfile function

minimum\_length    numeric;

maximum\_length    numeric

**Value**

data.frame

---

overlap\_bin\_with\_segment  
*Overlap and merge bin data frame with segmentation dataframe*

---

**Description**

Overlap and merge bin data frame with segmentation dataframe

**Usage**

```
overlap_bin_with_segment(per_bin_profile, sample_segmentation)
```

**Arguments**

```
per_bin_profile      bin dataframe
sample_segmentation  segmentation dataframe
```

**Value**

dataframe of overlapping bin and segmentation

---

plot\_cnv\_solution      *Plot Fragment-length profile with CNV calling result*

---

**Description**

Plot Fragment-length profile with CNV calling result

**Usage**

```
plot_cnv_solution(
  cnvcall,
  selected_solution = 1,
  genome = "hg19",
  ylim = c(-30, 30)
)
```

**Arguments**

```
cnvcall      solution results from call_cnv function
selected_solution  solution rank to plot
genome       Character; version of reference genome (default hg19)
ylim        Vector of 2 Int; ylim of plot (default c(-20,20))
```

**Value**

ggplot object plot Genomics CNV profile of selected solution

**Examples**

```
### Loading example SampleBam file
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")
sample_bambin <- readRDS(example_file)
### Example PoN
PoN_rdsfile <- system.file("extdata", "ex.PoN.rds", package = "cfdnakit")
pon_profiles <- readRDS(PoN_rdsfile)
sample_profile <- get_fragment_profile(sample_bambin, sample_id = "Patient1")

sample_zscore <- get_zscore_profile(sample_profile, pon_profiles)
sample_zscore_segment <- segmentByPSCB(sample_zscore)

sample_cnv <- call_cnv(sample_zscore_segment, sample_zscore, tfs=c(0.1,0.3), ploidies=c(1.5,2), MaxCN=3)
plot_cnv_solution(sample_cnv, selected_solution = 1)
```

---

plot\_distance\_matrix *Plot Distance Matrix from CNVCalling*

---

**Description**

Plot Distance Matrix from CNVCalling

**Usage**

```
plot_distance_matrix(cnvcall)
```

**Arguments**

cnvcall           cnvcalling result from function call\_cnv.R

**Value**

ggplot object ; distance matrix per cnvcalling solution

**Examples**

```
### Loading example SampleBam file
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")
sample_bambin <- readRDS(example_file)
### Example PoN
PoN_rdsfile <- system.file("extdata", "ex.PoN.rds", package = "cfdnakit")
pon_profiles <- readRDS(PoN_rdsfile)
sample_profile <- get_fragment_profile(sample_bambin, sample_id = "Patient1")
```

```

sample_zscore <- get_zscore_profile(sample_profile,pon_profiles)
sample_zscore_segment <- segmentByPSCB(sample_zscore)

sample_cnv <- call_cnv(sample_zscore_segment,sample_zscore, tfs=c(0.1,0.3),ploidies=c(1.5,2), MaxCN=3)
plot_distance_matrix(sample_cnv)

```

---

plot\_fragment\_dist      *Plot Fragment-length Distribution*

---

### Description

Plot Fragment-length Distribution

### Usage

```
plot_fragment_dist(readbam_list, maximum_length = 550, minimum_length = 20)
```

### Arguments

|                |  |
|----------------|--|
| readbam_list   | List; A list containing SampleBam object/objects from the read_bamfile function                            |
| maximum_length | Int; Maximum length of fragment. cfDNA fragment longer than this value will not be considered; Default 550 |
| minimum_length | Int; Minimum length of fragment. cfDNA fragment shorter than this value will not be considered; Default 20 |

### Value

distribution plot

### Examples

```

example_file <- system.file("extdata","example_patientcfDNA_SampleBam.RDS",package = "cfdnakit")
sample_bambin <- readRDS(example_file)

### adding more samples to the plot
example_file2 <- system.file("extdata","BH01_CHR15.SampleBam.rds",package = "cfdnakit")
control_bambin <- readRDS(example_file2)
readbam_list <- list(plasma1 = sample_bambin, Healthy.blood.plasma=control_bambin)
plot_fragment_dist(readbam_list)

```

---

plot\_sl\_ratio      *Plot Short/Long-fragment Ratio*

---

**Description**

Plot Short/Long-fragment Ratio

**Usage**

```
plot_sl_ratio(fragment_profile, ylim = c(0, 0.4), genome = "hg19")
```

**Arguments**

|                  |   |
|------------------|---|
| fragment_profile | list  |
| ylim             | plot y-axis limit                                     |
| genome           | Character; version of reference genome (default hg19) |

**Value**

plot

**Examples**

```
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")
sample_bambin <- readRDS(example_file)
sample_profile <- get_fragment_profile(sample_bambin, sample_id = "Patient1")
plot_sl_ratio(fragment_profile = sample_profile)

### change plot y-axis
plot_sl_ratio(fragment_profile = sample_profile, ylim=c(0.1,0.5))

### change reference genome
plot_sl_ratio(fragment_profile = sample_profile, genome="hg38")
```

---

plot\_transformed\_sl      *Plot z-transformed Short/Long-fragment Ratio*

---

**Description**

Plot z-transformed Short/Long-fragment Ratio

**Usage**

```
plot_transformed_sl(
  sample_transformed_sl,
  sample_segment_df = NULL,
  ylim = c(-30, 30),
  genome = "hg19"
)
```

**Arguments**

```
sample_transformed_sl      Dataframe z-transformed SLRatio from get_zscore_profile
sample_segment_df         Dataframe segmenation from segmentByPSCB
ylim                       plot y-axis limit
genome                     Character; version of reference genome (default hg19)
```

**Value**

Genome-wide plot of z-transformed SLRatio

**Examples**

```
### Loading example SampleBam file
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")
sample_bambin <- readRDS(example_file)
### Example PoN
PoN_rdsfile <- system.file("extdata", "ex.PoN.rds", package = "cfdnakit")
pon_profiles <- readRDS(PoN_rdsfile)
sample_profile <- get_fragment_profile(sample_bambin, sample_id = "Patient1")

sample_zscore <- get_zscore_profile(sample_profile, pon_profiles)
sample_zscore_segment <- segmentByPSCB(sample_zscore)
plot_transformed_sl(sample_zscore, sample_zscore_segment)
## Change reference genome
plot_transformed_sl(sample_zscore, sample_zscore_segment, genome="hg38")
```

---

|              |   |
|--------------|---|
| read_bamfile | <i>Read a bam file Read a bam file from give path. Alignment and sequencing read information will be binned into non-overlapping size</i> |
|--------------|---|

---

**Description**

Read a bam file Read a bam file from give path. Alignment and sequencing read information will be binned into non-overlapping size

**Usage**

```
read_bamfile(
  bamfile_path,
  binsize = 1000,
  blacklist_files = NULL,
  genome = "hg19",
  target_bedfile = NULL,
  min_mapq = 20,
  apply_blacklist = TRUE
)
```

**Arguments**

|                 |  |
|-----------------|--|
| bamfile_path    | Character; Path to sample bamfile  |
| binsize         | Int; Size of non-overlapping windows in KB. Only 100,500 and 1000 is available; Default 1000 |
| blacklist_files | Character; Filepath to file containing blacklist regions                                     |
| genome          | Character; abbreviation of reference genome; available genome: hg19,hg38, mm10. default:hg19 |
| target_bedfile  | Character; Path to exon/target bedfile; Default NULL   |
| min_mapq        | Int; minimum read mapping quality; Default 20  |
| apply_blacklist | Logical; To exclude read on the blacklist regions Default TRUE                               |

**Value**

SampleBam Object; A list object containing read information from the BAM file.

**Examples**

```
f1 <- system.file("extdata","ex.plasma.bam",package = "cfdnakit")
### read bam file with default params (hg19, 1000K binsize)
sample.bam <-read_bamfile(f1, apply_blacklist=FALSE)
```

---

|                |  |
|----------------|--|
| read_PoN_files | <i>Read Fragment Profile from a list of rds file</i> |
|----------------|--|

---

**Description**

Read Fragment Profile from a list of rds file

**Usage**

```
read_PoN_files(list_rdsfiles)
```

**Arguments**

list\_rdsfiles path to file containing list of rds file

**Value**

list containing content of rds file

---

|               |                                     |
|---------------|-------------------------------------|
| segmentByPSCB | <i>Segmentation data with PSCBS</i> |
|---------------|-------------------------------------|

---

**Description**

Segmentation data with PSCBS

**Usage**

```
segmentByPSCB(sample_transformed_sl)
```

**Arguments**

sample\_transformed\_sl  
dataframe of z-transformed SLRatio

**Value**

Dataframe of segmentation result

**Examples**

```
### Loading example SampleBam file
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")
sample_bambin <- readRDS(example_file)
### Example PoN
PoN_rdsfile <- system.file("extdata", "ex.PoN.rds", package = "cfdnakit")
pon_profiles <- readRDS(PoN_rdsfile)
sample_profile <- get_fragment_profile(sample_bambin, sample_id = "Patient1")

sample_zscore <- get_zscore_profile(sample_profile, pon_profiles)
sample_zscore_segment <- segmentByPSCB(sample_zscore)
```

---

```
test_ysize_KolmogorovSmirnov
      KolmogorovSmirnov test for insert size
```

---

**Description**

KolmogorovSmirnov test for insert size

**Usage**

```
test_ysize_KolmogorovSmirnov(control_insert_size, sample_insert_size)
```

**Arguments**

```
control_insert_size
      Vector of insert size of a control sample
sample_insert_size
      Vector of insert size of a testing sample
```

**Value**

KS.Test result

**Examples**

```
### Loading example SampleBam file
example_file <- system.file("extdata", "example_patientcfDNA_SampleBam.RDS", package = "cfdnakit")
sample_bambin <- readRDS(example_file)
control_rds <- "BH01_CHR15.SampleBam.rds"
control_RDS_file <- system.file("extdata", control_rds, package = "cfdnakit")
control_fragment_profile <- readRDS(control_RDS_file)
sample.isize <- extract_insert_size(sample_bambin)
healthy.isize <- extract_insert_size(control_fragment_profile)
test_ysize_KolmogorovSmirnov(sample.isize, healthy.isize)
```

---

```
UCSC2GRChSampleBam      Convert UCSC chromosome format to GRCh style from a list of alignment information
```

---

**Description**

Convert UCSC chromosome format to GRCh style from a list of alignment information

**Usage**

```
UCSC2GRChSampleBam(sample.bam)
```

**Arguments**

sample.bam      list of alignment information from function read\_bamfile

**Value**

List; list of alignment information after conversion

---

util.bias\_correct      *Correct GC Bias readcount*

---

**Description**

Correct GC Bias readcount

**Usage**

```
util.bias_correct(readcount, bias)
```

**Arguments**

readcount      numeric  
bias            numeric

**Value**

numeric

---

zscore\_transform      *zscore\_transform transforms SLRatio profile into z-score*

---

**Description**

zscore\_transform transforms SLRatio profile into z-score

**Usage**

```
zscore_transform(per_bin_profile)
```

**Arguments**

per\_bin\_profile  
SampleFragment from function get\_fragment\_profile

**Value**

dataframe of z-score per bin

---

`%>%`*Pipe operator*

---

**Description**

See `magrittr::%>%` for details.

**Arguments**

|                  |  |
|------------------|--|
| <code>lhs</code> | A value or the <code>magrittr</code> placeholder.          |
| <code>rhs</code> | A function call using the <code>magrittr</code> semantics. |

**Value**

The result of calling `rhs(lhs)`.

# Index

- \* **internal**
  - %>%, [23](#)
- \* **package cf-DNA**
  - cfdnakit-package, [2](#)
  - %>%, [23](#), [23](#)
- calculate\_CES\_score, [4](#)
- call\_cnv, [5](#)
- cfdnakit (cfdnakit-package), [2](#)
- cfdnakit-package, [2](#)
- create\_blacklist\_gr, [6](#)
- create\_PoN, [6](#)
- extract\_insert\_size, [7](#)
- filter\_read\_on\_blacklist, [8](#)
- fragment\_dist, [8](#)
- get\_fragment\_profile, [9](#)
- get\_segment\_byresolution, [10](#)
- get\_solution\_table, [10](#)
- get\_zscore\_profile, [11](#)
- GRCh2UCSCGRanges, [12](#)
- if\_exist\_baifile, [12](#)
- if\_ucsc\_chrformat, [13](#)
- make\_density\_table, [13](#)
- overlap\_bin\_with\_segment, [14](#)
- plot\_cnv\_solution, [14](#)
- plot\_distance\_matrix, [15](#)
- plot\_fragment\_dist, [16](#)
- plot\_sl\_ratio, [17](#)
- plot\_transformed\_sl, [17](#)
- read\_bamfile, [18](#)
- read\_PoN\_files, [19](#)
- segmentByPSCB, [20](#)
- test\_ysize\_KolmogorovSmirnov, [21](#)
- UCSC2GRChSampleBam, [21](#)
- util.bias\_correct, [22](#)
- zscore\_transform, [22](#)