

# Package ‘RNAprobR’

April 15, 2020

**Title** An R package for analysis of massive parallel sequencing based RNA structure probing data

**Version** 1.18.0

**Date** 2017-10-27

**Description** This package facilitates analysis of Next Generation Sequencing data for which positional information with a single nucleotide resolution is a key. It allows for applying different types of relevant normalizations, data visualization and export in a table or UCSC compatible bedgraph file.

**Depends** R (>= 3.1.1), GenomicFeatures(>= 1.16.3), plyr(>= 1.8.1), BiocGenerics(>= 0.10.0)

**Imports** Biostrings(>= 2.32.1), GenomicRanges(>= 1.16.4), IRanges(>= 2.10.5), Rsamtools(>= 1.16.1), rtracklayer(>= 1.24.2), GenomicAlignments(>= 1.5.12), S4Vectors(>= 0.14.7), graphics, stats, utils

**Suggests** BiocStyle

**License** GPL (>=2)

**LazyData** true

**biocViews** Coverage, Normalization, Sequencing, GenomeAnnotation

**RoxygenNote** 6.0.1

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

**git\_branch** RELEASE\_3\_10

**git\_last\_commit** 35a155a

**git\_last\_commit\_date** 2019-10-29

**Date/Publication** 2020-04-14

**Author** Lukasz Jan Kielpinski [aut],  
Nikos Sidiropoulos [cre, aut],  
Jeppe Vinther [aut]

**Maintainer** Nikos Sidiropoulos <[nikos.sidiro@gmail.com](mailto:nikos.sidiro@gmail.com)>

## R topics documented:

bam2bedgraph . . . . .	2
BED2txDb . . . . .	3

bedgraph2norm . . . . .	4
comp . . . . .	5
compdata . . . . .	6
correct_oversaturation . . . . .	7
dtr . . . . .	8
GR2norm_df . . . . .	9
k2n_calc . . . . .	10
norm2bedgraph . . . . .	11
norm_df2GR . . . . .	12
plotReads . . . . .	13
plotRNA . . . . .	14
readsamples . . . . .	15
slograt . . . . .	16
swinsor . . . . .	17
swinsor_vector . . . . .	18
winsor . . . . .	19

<b>Index</b>	<b>21</b>
--------------	-----------

---

bam2bedgraph	<i>Function converts bam file to bedgraph by counting number of reads starting at each position (termination counts). It creates two-track bedgraph file (one track for each strand).</i>
--------------	---

---

## Description

Function converts bam file to bedgraph by counting number of reads starting at each position (termination counts). It creates two-track bedgraph file (one track for each strand).

## Usage

```
bam2bedgraph(bam_path, allowed_flags = 0:4095, maxMemory = 8000,
             genome_build, bedgraph_out_file = "out_file", track_name = "Track_name",
             track_description = "Track_description")
```

## Arguments

bam_path	path to a bam file to be converted
allowed_flags	integer vector with SAM flags should be kept, see <a href="https://broadinstitute.github.io/picard/explain-flags.html">https://broadinstitute.github.io/picard/explain-flags.html</a> for explanation
maxMemory	maxMemory of scanBam function used internally
genome_build	character specifying which UCSC genome build should data be displayed in, e.g. "mm9"
bedgraph_out_file	character specifying prefix of output file. Generated file name is: prefix.bedgraph; if file with such a name already exists new tracks will be appended.
track_name	character specifying track name
track_description	character specifying track description

**Value**

NULL. Creates a two-track bedgraph file (one track for each strand).

**Author(s)**

Lukasz Jan Kielpinski

---

BED2txDb

*Bedgraph to TranscriptDb object*

---

**Description**

Function to transform BED format file to Bioconductor TranscriptDb object

**Usage**

```
BED2txDb(input_bed_path)
```

**Arguments**

`input_bed_path` Path to BED file. If 12 column BED provided, function is splice aware. If 6 column BED provided, function assumes no splicing.

**Value**

TranscriptDb object

**Author(s)**

Lukasz Jan Kielpinski, Nikos Sidiropoulos

**Examples**

```
write(paste(c("chr1", 134212702, 134229870, "ENSMUST0000072177", 0, "+",
             134212806, 134228958, 0, 8, "347,121,24,152,66,120,133,1973,",
             "0,8827,10080,11571,12005,13832,14433,15195,"), collapse = "\t"),
      file="dummy.bed")
BED2txDb("dummy.bed")
```



```

                                width=round(runif(100)*100+1)),
                                strand="+", EUC=round(runif(100)*100))
dummy_comp_GR_control <- comp(dummy_euc_GR_control)
dummy_comp_GR_treated <- comp(dummy_euc_GR_treated)
dummy_norm <- dtcr(control_GR=dummy_comp_GR_control,
                  treated_GR=dummy_comp_GR_treated)

write(paste(c("chr1", 134212702, 134229870, "DummyRNA", 0, "+", 134212806,
             134228958, 0, 8, "347,121,24,152,66,120,133,1973,",
             "0,8827,10080,11571,12005,13832,14433,15195,"), collapse = "\t"),
      file="dummy.bed")
norm2bedgraph(norm_GR = dummy_norm, bed_file = "dummy.bed")

write(c(">DummyRNA", paste(sample(c("A","C","G","T"), 100, replace=TRUE),
                              collapse="")), file="dummy.fa")
bedgraph2norm(bedgraph_file = "out_file.bedgraph", fasta_file = "dummy.fa",
              bed_file = "dummy.bed")

```

---

comp	<i>Arranging information from GRanges produced by readsamples() on per position (nucleotide) basis.</i>
------	---

---

## Description

comp() takes as input euc\_GR GRanges object produced by readsamples() and produces Comp\_GR GRanges.

## Usage

```
comp(euc_GR, cutoff = 1, fasta_file)
```

## Arguments

euc_GR	GRanges generated by readsamples() function
cutoff	specifies cutoff length, only inserts of this length or longer will be used for processing (default: 1)
fasta_file	path to fasta file to which reads were mapped. Used to report nucleotide at each position (not required)

## Value

GRanges object with: 1) seqnames (RNAid), 2) start (position within RNA), and metadata: 3) TCR (termination coverage ratio), 4) TC (termination count), 5) Cover (coverage) and 6) PC (priming count) for each position within each RNA.

## Author(s)

Lukasz Jan Kielbinski, Nikos Sidiropoulos

## References

Kielbinski, L.J., and Vinther, J. (2014). Massive parallel-sequencing-based hydroxyl radical probing of RNA accessibility. *Nucleic Acids Res.*

**See Also**

[readsamples](#), [dtcr](#), [slograt](#), [swinsor](#), [compdata](#), [comp](#)

**Examples**

```
dummy_euc_GR <- GRanges(seqnames="DummyRNA",
                        IRanges(start=round(runif(100)*100),
                                width=round(runif(100)*100+1)), strand="+",
                        EUC=round(runif(100)*100))
comp(dummy_euc_GR)
```

---

compdata

*Create or extend norm\_GR GRanges using Comp\_GR GRanges*

---

**Description**

Add metadata present in GRanges made by `comp()` function (termination count (TC), termination-coverage ratio (TCR), coverage (Cover) and priming count (PC)) to GRanges made by normalizing functions (`dtcr()`, `slograt()`, `swinsor()`, `compdata()`).

**Usage**

```
compdata(Comp_GR, nt_offset = 1, add_to)
```

**Arguments**

Comp_GR	GRanges object made by <code>comp()</code> function.
nt_offset	how many nucleotides before modification the reverse transcription terminates (default: 1)
add_to	normalized data frame with already performed normalization of another kind. Results will be merged

**Value**

norm\_GR norm\_GR GRanges extended by metadata from Comp\_GR

**Author(s)**

Lukasz Jan Kielpinski, Nikos Sidiropoulos

**See Also**

[comp](#), [dtcr](#), [slograt](#), [swinsor](#), [GR2norm\\_df](#), [plotRNA](#), [norm2bedgraph](#)

**Examples**

```
dummy_euc_GR_treated <- GRanges(seqnames="DummyRNA",
                                IRanges(start=round(runif(100)*100),
                                          width=round(runif(100)*100+1)), strand="+",
                                EUC=round(runif(100)*100))
dummy_comp_GR_treated <- comp(dummy_euc_GR_treated)
dummy_swinsor <- swinsor(dummy_comp_GR_treated)
dummy_swinsor <- compdata(Comp_GR=dummy_comp_GR_treated,
                          add_to=dummy_swinsor)
dummy_swinsor
```

---

correct\_oversaturation

*Correcting EUC of oversaturated fragments.*

---

**Description**

If for a given fragment the number of observed unique barcodes is equal to the total barcode complexity (all combinations of barcodes are associated with a given fragment), then the readsamples function assigns infinite EUC. This can be corrected by the function `correct_oversaturation()`. By comparing observed read counts with EUCs for other fragments it calculates the correction factor. Then, for the oversaturated fragments it multiplies the observed read counts by the correction factor to estimate EUC. The assumption behind this correction is that fragments have similar rate of PCR duplicates production.

**Usage**

```
correct_oversaturation(euc_GR, read_counts_file)
```

**Arguments**

`euc_GR` GRanges produced by readsamples() function  
`read_counts_file` path to a file with observed read counts.

**Value**

`euc_GR` GRanges analogous to the readsamples() function output, but with finite EUCs where infinity was present.

**Examples**

```
write(c("DummyRNA\t1\t2\t1000", "DummyRNA\t3\t4\t1024"),
      file="dummy_unique_barcode")
write(c("DummyRNA\t1\t2\t5000", "DummyRNA\t3\t4\t10000"),
      file="dummy_read_counts")
my_EUCs <- readsamples(samples = "dummy_unique_barcode", euc = "Fu", m=1024)
correct_oversaturation(euc_GR = my_EUCs,
                      read_counts_file = "dummy_read_counts")
```

---

dtr	<i>Calculate deltaTCR.</i>
-----	----------------------------

---

### Description

Performs deltaTCR (dtr) normalization given control and treated GRanges generated by comp() function.

### Usage

```
dtr(control_GR, treated_GR, window_size = 3, nt_offset = 1,  
    bring_to_zero = TRUE, add_to)
```

### Arguments

control_GR	GRanges object made by comp() function from the control sample.
treated_GR	GRanges object made by comp() function from the treated sample.
window_size	if smoothing is to be performed, what should be the window size? (use only odd numbers to ensure that windows are centred on a nucleotide of interest) (default: 3)
nt_offset	how many nucleotides before a modification the reverse transcription terminates. E.g. for HRF-Seq nt_offset=1 (default: 1)
bring_to_zero	should in deltaTCR calculations negative deltaTCR's be brought to 0 as was done in HRF-Seq paper (default: T)
add_to	GRanges object made by other normalization function (dtr(), slograt(), swinsor(), compdata()) to which normalized values should be added.

### Value

GRanges object with "dtr" (deltaTCR) and "dtr.p" (p.value of comparing control and treated calculated with pooled two-proportion Z-test) metadata.

### Author(s)

Lukasz Jan Kielpinski, Nikos Sidiropoulos

### References

Kielpinski, L.J., and Vinther, J. (2014). Massive parallel-sequencing-based hydroxyl radical probing of RNA accessibility. *Nucleic Acids Res.*

### See Also

[comp](#), [slograt](#), [swinsor](#), [compdata](#), [GR2norm\\_df](#), [plotRNA](#), [norm2bedgraph](#)



**Examples**

```

dummy_euc_GR_control <- GRanges(seqnames="DummyRNA",
                                IRanges(start=round(runif(100)*100),
                                         width=round(runif(100)*100+1)), strand="+",
                                EUC=round(runif(100)*100))
dummy_euc_GR_treated <- GRanges(seqnames="DummyRNA",
                                IRanges(start=round(runif(100)*100),
                                         width=round(runif(100)*100+1)), strand="+",
                                EUC=round(runif(100)*100))
dummy_comp_GR_control <- comp(dummy_euc_GR_control)
dummy_comp_GR_treated <- comp(dummy_euc_GR_treated)
dtkr(control_GR=dummy_comp_GR_control, treated_GR=dummy_comp_GR_treated)

```

GR2norm\_df

*Export normalized GRanges object to data frame***Description**

Function to make data frame out of GRanges output of normalizing functions (dtkr(), slograt(), swinsor(), compdata()) for all or a set of chosen transcripts in the file.

**Usage**

```
GR2norm_df(norm_GR, RNAid = "all", norm_methods = "all")
```

**Arguments**

norm_GR	GRanges object made by other normalization function (dtkr(), slograt(), swinsor(), compdata()) from which data is to be extracted
RNAid	Transcript identifiers of transcripts that are to be extracted
norm_methods	Names of the columns to be extracted.

**Value**

Data frame object with columns: RNAid, Pos and desired metadata columns (e.g. nt, dtkr)

**Author(s)**

Lukasz Jan Kielpinski, Nikos Sidiropoulos

**See Also**

[norm\\_df2GR](#), [dtkr](#), [swinsor](#), [slograt](#), [compdata](#)

**Examples**

```
dummy_euc_GR_treated <- GRanges(seqnames="DummyRNA",
                                IRanges(start=round(runif(100)*100),
                                          width=round(runif(100)*100+1)), strand="+",
                                EUC=round(runif(100)*100))
dummy_comp_GR_treated <- comp(dummy_euc_GR_treated)
dummy_swinsor <- swinsor(dummy_comp_GR_treated)
GR2norm_df(dummy_swinsor)
```

k2n\_calc

*Calculate number of Estimated Unique Counts (EUC's) corresponding to given number of observed unique barcodes.*

**Description**

Function calculates EUC's for each number of observed barcodes accounting for differential ligation probability of different barcodes. Function `k2n_calc()` writes file with a vector in which an *i*-th element is an estimated unique count given observing *i* unique barcodes.

**Usage**

```
k2n_calc(merged_file, unique_barcode_file, output_file)
```

**Arguments**

<code>merged_file</code>	path to merged_temp file containing 4 column: 1) RNAid, 2) Start, 3) End, 4) Barcode sequence (required)
<code>unique_barcode_file</code>	character with path to unique_barcode file (required)
<code>output_file</code>	name of a file to be generated (if specified [recommended] function will write a file, if not - function will return a vector)

**Value**

If `output_file` specified function writes a file, if not - returns a vector.

**Author(s)**

Lukasz Jan Kielinski, Nikos Sidiropoulos

**References**

Kielinski, L.J., and Vinther, J. (2014). Massive parallel-sequencing-based hydroxyl radical probing of RNA accessibility. *Nucleic Acids Res.*

**See Also**

[readsamples](#)

**Examples**

```
write(c("DummyRNA\t1\t1\tA", "DummyRNA\t1\t1\tC", "DummyRNA\t2\t2\tG",
       "DummyRNA\t2\t2\tT"), file="dummy_merged_file")
write(c("DummyRNA\t1\t1\t2", "DummyRNA\t2\t2\t2"),
      file="dummy_unique_barcode")
k2n_calc(merged_file = "dummy_merged_file",
        unique_barcode_file = "dummy_unique_barcode")
```

---

norm2bedgraph	<i>Exporting data in norm_df data frame (product of dtcr, slograt and swinsor) to bedgraph format compatible with UCSC Genome Browser</i>
---------------	---

---

**Description**

Function converts annotation from transcript to genomic coordinates and creates two-track bedgraph file (one track for each strand)

**Usage**

```
norm2bedgraph(norm_GR, txDb, bed_file, norm_method, genome_build,
              bedgraph_out_file = "out_file", track_name = "Track_name",
              track_description = "Track_description")
```

**Arguments**

norm_GR	norm_GR GRanges with data to be exported, required
txDb	TranscriptDb object with transcript definitions. Names must match those in norm_df
bed_file	character containing file path to BED file with transcript definitions. Supply txDb XOR bedfile
norm_method	character specifying which normalized column should be processed into bed-graph. If not provided, the first column matching dtcr, slograt or swinsor is transformed.
genome_build	character specifying which UCSC genome build should data be displayed in, e.g. "mm9"
bedgraph_out_file	character specifying prefix of output file. Generated file name is: prefix.bedgraph; if file with such a name already exists new tracks will be appended.
track_name	character specifying track name
track_description	character specifying track description

**Value**

Function writes bedgraph file.

**Author(s)**

Lukasz Jan Kielpinski, Nikos Sidiropoulos

**See Also**

[bedgraph2norm](#), [norm\\_df2GR](#), [docr](#), [slograt](#), [swinsor](#), [compdata](#)

**Examples**

```
dummy_euc_GR_control <- GRanges(seqnames="DummyRNA",
                                IRanges(start=round(runif(100)*100),
                                         width=round(runif(100)*100+1)), strand="+",
                                EUC=round(runif(100)*100))
dummy_euc_GR_treated <- GRanges(seqnames="DummyRNA",
                                 IRanges(start=round(runif(100)*100),
                                         width=round(runif(100)*100+1)), strand="+",
                                 EUC=round(runif(100)*100))
dummy_comp_GR_control <- comp(dummy_euc_GR_control)
dummy_comp_GR_treated <- comp(dummy_euc_GR_treated)
dummy_norm <- dtcr(control_GR=dummy_comp_GR_control,
                  treated_GR=dummy_comp_GR_treated)
write(paste(c("chr1", 134212702, 134229870, "DummyRNA", 0, "+", 134212806,
             134228958, 0, 8, "347,121,24,152,66,120,133,1973,",
             "0,8827,10080,11571,12005,13832,14433,15195,"), collapse = "\t"),
      file="dummy.bed")
norm2bedgraph(norm_GR = dummy_norm, bed_file = "dummy.bed")
```

---

norm\_df2GR

*Function to convert norm\_df data frame (product of GR2norm\_df()) to GRanges.*

---

**Description**

Function to convert norm\_df data frame (product of GR2norm\_df()) to GRanges.

**Usage**

```
norm_df2GR(norm_df)
```

**Arguments**

norm\_df            norm\_df data frame needs to have columns: RNAid (equivalent to seqnames in GRanges) and Pos (equivalent to start in GRanges) and metadata

**Value**

GRanges compatible with objects created by normalizing functions (docr(), slograt(), swinsor(), compdata())

**Author(s)**

Lukasz Jan Kielpinski

**See Also**

[docr](#), [slograt](#), [swinsor](#), [compdata](#), [GR2norm\\_df](#), [norm2bedgraph](#)

**Examples**

```
dummy_norm_df <- data.frame(RNAid="dummyRNA", Pos=1:100,
                           my_data1=runif(1:100))
norm_df2GR(dummy_norm_df)
```

plotReads

*Plotting ranges from GRanges***Description**

Function plots cDNA inserts from GRanges created by readsamples() function. Similar to Figure 4A in HRF-Seq paper (see References).

**Usage**

```
plotReads(euc_GR, RNAid, cutoff = 1, order_by = 1, ylab, xlab, main, ylim,
          xlim, ...)
```

**Arguments**

euc_GR	GRanges generated by readsamples() function
RNAid	Transcript identifier, for which transcript plot should be generated.
cutoff	specifies cutoff length, only inserts of this length or longer will be used for processing (default: 1)
order_by	how displayed reads in plotReads function should be sorted. 1 - for sorting by termination location, 2 for sorting by reverse transcription start site
ylab	a title for the y axis: see <a href="#">title</a> .
xlab	a title for the x axis: see <a href="#">title</a> .
main	an overall title for the plot: see <a href="#">title</a> .
xlim, ylim	numeric vectors of length 2, giving the x and y coordinates ranges.
...	Arguments to be passed to methods, such as <a href="#">graphical parameters</a> (see <a href="#">par</a> ).

**Value**

Plotting function.

**Author(s)**

Lukasz Jan Kielpinski

**References**

Kielpinski, L.J., and Vinther, J. (2014). Massive parallel-sequencing-based hydroxyl radical probing of RNA accessibility. *Nucleic Acids Res.*

**See Also**

[plot](#), [plot.default](#), [readsamples](#)

**Examples**

```
dummy_euc_GR <- GRanges(seqnames="DummyRNA",
                        IRanges(start=round(runif(100)*100),
                                width=round(runif(100)*100+1)), strand="+",
                        EUC=round(runif(100)*100))
plotReads(dummy_euc_GR, RNAid="DummyRNA")
```

---

plotRNA

*Plot normalized values over transcript positions*


---

**Description**

Function plotting normalized values over transcript positions.

**Usage**

```
plotRNA(norm_GR, RNAid, norm_method, stat_method, stat_cutoff, main, type, ylab,
        xlab, ...)
```

**Arguments**

norm_GR	norm_GR GRanges with data to be exported, required
RNAid	Transcript identifier, for which transcript plot should be generated.
norm_method	Which normalization method should be to be used for plotting (column name).
stat_method	Name of a column to be used for adding significance asterisks. If stat_method not provided, function tries to match with "norm_method", if no guess - empty vector.
stat_cutoff	below what value of statistics (from stat_method, p-value or standard deviation) report significance. If not provided - minimal value from stat_method used. To suppress reporting significant sites provide negative value
main	an overall title for the plot: see <a href="#">title</a> .
type	what type of plot should be drawn. See <a href="#">plot</a> for possible types.
ylab	a title for the y axis: see <a href="#">title</a> .
xlab	a title for the x axis: see <a href="#">title</a> .
...	Arguments to be passed to methods, such as <a href="#">graphical parameters</a> (see <a href="#">par</a> ).

**Value**

Plotting function.

**Author(s)**

Lukasz Jan Kielpinski

**See Also**

[plot](#), [plot.default](#), [docr](#), [slograt](#), [swinsor](#), [compdata](#)

## Examples

```
dummy_euc_GR_treated <- GRanges(seqnames="DummyRNA",
                                IRanges(start=round(runif(100)*100),
                                         width=round(runif(100)*100+1)), strand="+",
                                EUC=round(runif(100)*100))
dummy_comp_GR_treated <- comp(dummy_euc_GR_treated)
dummy_swinsor <- swinsor(dummy_comp_GR_treated)
plotRNA(dummy_swinsor, RNAid="DummyRNA")
```

---

readsamples

---

*Import of tables prepared by Galaxy workflow to R environment*


---

## Description

Function readsamples() reads the output of read processing and mapping workflow which has to consist of 4 columns 1) RNAid, 2)Insert start, 3)Insert end, 4)Unique barcode count. It combines separate files coming from the same treatment (e.g. controls) and calculates estimated unique counts (EUCs) by either (a) keeping unique counts (euc="counts"), (b) using formula from Fu GK et al. PNAS 2011 (binomial distribution calculation) (euc="Fu") or (c) using method described in Kielpinski and Vinther, NAR 2014 (euc="HRF-Seq") If euc="Fu" then the count of all possible barcodes is required (m), e.g. if you use 7 nucleotide, fully degenerate random barcodes (NNNNNNN) then m=16384 (m=4\*\*7) If euc="HRF-Seq" then the path to a precomputed k2n file is required (generate using k2n\_calc() function)(default: "counts")

## Usage

```
readsamples(samples, euc = "counts", m = "", k2n_files = "")
```

## Arguments

samples	vector with paths to unique_barcodes files to be combined
euc	method of calculating estimated unique counts (default: "counts")
m	random barcode complexity (required if and only if euc="Fu")
k2n_files	vector with paths to k2n files corresponding to files given in samples (required if and only if euc="HRF-Seq"; order important!). Recycled if necessary

## Value

euc\_GR GRanges containing information: 1) seqnames (sequence name; RNAid) 2) Start, 3) End, 4) EUC value of a given fragment

## Author(s)

Lukasz Jan Kielpinski, Nikos Sidiropoulos

## References

Fu, G.K., Hu, J., Wang, P.H., and Fodor, S.P. (2011). Counting individual DNA molecules by the stochastic attachment of diverse labels. Proc Natl Acad Sci U S A 108, 9026-9031. Kielpinski, L.J., and Vinther, J. (2014). Massive parallel-sequencing-based hydroxyl radical probing of RNA accessibility. Nucleic Acids Res.

**See Also**

[comp](#), [plotReads](#), [k2n\\_calc](#)

**Examples**

```
write("DummyRNA\t1\t2\t3",file="dummy_unique_barcode")
readsamples(samples = "dummy_unique_barcode", euc = "counts")
```

---

slograt

*Smooth Log2-ratio*


---

**Description**

Performs smooth-log2-ratio calculation given control and treated GRanges generated by `comp()` function.

**Usage**

```
slograt(control_GR, treated_GR, window_size = 5, nt_offset = 1,
        depth_correction = "all", pseudocount = 5, add_to)
```

**Arguments**

<code>control_GR</code>	GRanges object made by <code>comp()</code> function from the control sample.
<code>treated_GR</code>	GRanges object made by <code>comp()</code> function from the treated sample.
<code>window_size</code>	if smoothing is to be performed, then what should be the window size? (use only odd numbers to ensure that windows are centred on a nucleotide of interest) (default: 5)
<code>nt_offset</code>	How many position in the 5' direction should the signal be offset to account for the fact that reverse transcription termination occurs before site of modification.
<code>depth_correction</code>	One of three values: "no" - counts are used as given, "all" - counts from sample with higher total sum of EUCs are multiplied by sum of EUCs from sample with lower total sum of EUCs and divided by sum of EUCs from sample with higher EUC count (default), "RNA" as in "all" but on per RNA basis
<code>pseudocount</code>	What pseudocount should be added to each nucleotide prior to calculating log2 ratio (default: 5)
<code>add_to</code>	GRanges object made by other normalization function ( <code>dscr()</code> , <code>slograt()</code> , <code>swin-sor()</code> , <code>compdata()</code> ) to which normalized values should be added.

**Value**

GRanges object with "slograt" (smooth log2 ratio) and "slograt.p" (p.value of comparing control and treated) metadata.

**Author(s)**

Lukasz Jan Kielpinski, Nikos Sidiropoulos



## References

Wan, Y., Qu, K., Zhang, Q.C., Flynn, R.A., Manor, O., Ouyang, Z., Zhang, J., Spitale, R.C., Snyder, M.P., Segal, E., et al. (2014). Landscape and variation of RNA secondary structure across the human transcriptome. *Nature* 505, 706-709.

## See Also

[comp](#), [dctcr](#), [compdata](#), [swinsor](#), [GR2norm\\_df](#), [plotRNA](#), [norm2bedgraph](#)

## Examples

```
dummy_euc_GR_control <- GRanges(seqnames="DummyRNA",
                                IRanges(start=round(runif(100)*100),
                                          width=round(runif(100)*100+1)), strand="+",
                                EUC=round(runif(100)*100))
dummy_euc_GR_treated <- GRanges(seqnames="DummyRNA",
                                IRanges(start=round(runif(100)*100),
                                          width=round(runif(100)*100+1)), strand="+",
                                EUC=round(runif(100)*100))
dummy_comp_GR_control <- comp(dummy_euc_GR_control)
dummy_comp_GR_treated <- comp(dummy_euc_GR_treated)
slograt(control_GR=dummy_comp_GR_control, treated_GR=dummy_comp_GR_treated)
```

---

swinsor

*Smooth Winsorization*

---

## Description

Performs sliding window Winsorization given treated GRanges generated by `comp()` function. It winsorizes values in windows (of a size specified by `window_size`) sliding by 1 nt over whole transcript length and reports mean winsorized value for each nucleotide (as well as standard deviation).

## Usage

```
swinsor(Comp_GR, winsor_level = 0.9, window_size = 71, only_top = FALSE,
        nt_offset = 1, add_to)
```

## Arguments

<code>Comp_GR</code>	GRanges object made by <code>comp()</code> function.
<code>winsor_level</code>	Winsorization level. Bottom outliers will be set to $(1-winsor\_level)/2$ quantile and top outliers to $(1+winsor\_level)/2$ quantile.
<code>window_size</code>	Size of a sliding window.
<code>only_top</code>	If TRUE then bottom values are not Winsorized and are set to 0.
<code>nt_offset</code>	How many position in the 5' direction should the signal be offset to account for the fact that reverse transcription termination occurs before site of modification.
<code>add_to</code>	GRanges object made by other normalization function ( <code>dctcr()</code> , <code>slograt()</code> , <code>swinsor()</code> , <code>compdata()</code> ) to which normalized values should be added.

**Value**

GRanges object with "swinsor" (mean smooth-Winsor values) and "swinsor.sd" (standard deviation of smooth-Winsor values) metadata.

**Author(s)**

Lukasz Jan Kielpinski, Jeppe Vinther, Nikos Sidiropoulos

**References**

"Analysis of sequencing based RNA structure probing data" Kielpinski, Sidiropoulos, Vinther. Chapter in "Methods in Enzymology" (in preparation)

**See Also**

[comp](#), [dtcr](#), [slograt](#), [compdata](#), [GR2norm\\_df](#), [plotRNA](#), [norm2bedgraph](#), [winsor](#), [swinsor\\_vector](#)

**Examples**

```
dummy_euc_GR <- GRanges(seqnames="DummyRNA",
                        IRanges(start=round(runif(100)*100),
                                width=round(runif(100)*100+1)), strand="+",
                        EUC=round(runif(100)*100))
dummy_comp_GR <- comp(dummy_euc_GR)
swinsor(dummy_comp_GR)
```

---

swinsor\_vector

*Smooth Winsor Normalization*

---

**Description**

Function performs Winsor normalization (see `winsor()` function) of each window of specified `window_size`, sliding in a given vector by 1 position, and reports a list of (1) mean Winsorized values for each vector position (mean of Winsorized value for a given position as calculated within each overlapping window) and (2) standard deviation of those Winsorized values.

**Usage**

```
swinsor_vector(input_vector, window_size, winsor_level = 0.9,
              only_top = FALSE)
```

**Arguments**

<code>input_vector</code>	Vector with values to be smooth-Winsorized
<code>window_size</code>	Size of a sliding window.
<code>winsor_level</code>	Winsorization level. Bottom outliers will be set to $(1-\text{winsor\_level})/2$ quantile and top outliers to $(1+\text{winsor\_level})/2$ quantile.
<code>only_top</code>	If TRUE then bottom values are not Winsorized and are set to 0.

**Value**

comp1            Vector with mean Winsorized values for each input\_vector position  
 comp2            Vector with standard deviation of Winsorized values for each input\_vector position

**Author(s)**

Lukasz Jan Kielpinski

**References**

"Analysis of sequencing based RNA structure probing data" Kielpinski, Sidiropoulos, Vinther. Chapter in "Methods in Enzymology" (in preparation)

**Examples**

```
data_set <- runif(1:100)*100
plot(swinsor_vector(data_set, window_size=71,
                    winsor_level=0.8)[[1]] ~ data_set)
```

---

winsor

*Winsor normalization with fitting to <0,1> range.*

---

**Description**

Function performs Winsor normalization of a supplied vector. Steps: 1. Calculate top winsor value  $[(1+winsor\_level)/2]$  quantile, and bottom winsor value  $[(1-winsor\_level)/2]$  quantile 2. Each value below bottom winsor value set to bottom winsor value; each value above top winsor value set to top winsor value 3. Transform linearly all the values to  $[0,1]$  range

**Usage**

```
winsor(input_vector, winsor_level = 0.9, only_top = FALSE)
```

**Arguments**

input\_vector    Vector with values to be Winsorized  
 winsor\_level    Winsorization level. Bottom outliers will be set to  $(1-winsor\_level)/2$  quantile and top outliers to  $(1+winsor\_level)/2$  quantile.  
 only\_top        If TRUE then bottom values are not Winsorized and the lowest is set to 0.

**Value**

Vector of numerics within  $<0,1>$ .

**Author(s)**

Lukasz Jan Kielpinski

**References**

Hastings, Cecil; Mosteller, Frederick; Tukey, John W.; Winsor, Charles P. Low Moments for Small Samples: A Comparative Study of Order Statistics. *The Annals of Mathematical Statistics* 18 (1947), no. 3, 413–426.

**Examples**

```
data_set <- runif(1:100)*100  
plot(winsor(data_set, winsor_level=0.8) ~ data_set)
```

# Index

## \*Topic ~winsorising

winsor, 19

bam2bedgraph, 2

BED2txDb, 3, 4

bedgraph2norm, 4, 12

comp, 5, 6, 8, 16–18

compdata, 4, 6, 6, 8, 9, 12, 14, 17, 18

correct\_oversaturation, 7

dtcr, 4, 6, 8, 9, 12, 14, 17, 18

GR2norm\_df, 4, 6, 8, 9, 12, 17, 18

graphical parameters, 13, 14

k2n\_calc, 10, 16

norm2bedgraph, 4, 6, 8, 11, 12, 17, 18

norm\_df2GR, 9, 12, 12

par, 13, 14

plot, 13, 14

plot.default, 13, 14

plotReads, 13, 16

plotRNA, 4, 6, 8, 14, 17, 18

readsamples, 6, 10, 13, 15

slograt, 4, 6, 8, 9, 12, 14, 16, 18

swinsor, 4, 6, 8, 9, 12, 14, 17, 17

swinsor\_vector, 18, 18

title, 13, 14

winsor, 18, 19