

# Package ‘HiCDCPlus’

December 23, 2024

**Type** Package

**Title** Hi-C Direct Caller Plus

**Version** 1.15.0

**Description** Systematic 3D interaction calls and differential analysis for Hi-C and HiChIP. The HiC-DC+ (Hi-C/HiChIP direct caller plus) package enables principled statistical analysis of Hi-C and HiChIP data sets – including calling significant interactions within a single experiment and performing differential analysis between conditions given replicate experiments – to facilitate global integrative studies. HiC-DC+ estimates significant interactions in a Hi-C or HiChIP experiment directly from the raw contact matrix for each chromosome up to a specified genomic distance, binned by uniform genomic intervals or restriction enzyme fragments, by training a background model to account for random polymer ligation and systematic sources of read count variation.

**License** GPL-3

**Encoding** UTF-8

**biocViews** HiC, DNA3DStructure, Software, Normalization

**RoxygenNote** 7.1.1

**SystemRequirements** JRE 8+

**LinkingTo** Rcpp

**Imports**

Rcpp,InteractionSet,GenomicInteractions,bbmle,pscl,BSgenome,data.table,dplyr,tidyr,GenomeInfoDb,rlang,splines,MASS

**Suggests** BSgenome.Mmusculus.UCSC.mm9, BSgenome.Mmusculus.UCSC.mm10, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Hsapiens.UCSC.hg38, RUnit, BiocGenerics, knitr, rmarkdown, HiTC, DESeq2, Matrix, BiocFileCache, rappdirs

**Enhances** parallel

**VignetteBuilder** knitr

**NeedsCompilation** yes

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

**git\_branch** devel

**git\_last\_commit** cd71b78

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-12-23

**Author** Merve Sahin [cre, aut] (ORCID: <<https://orcid.org/0000-0003-3858-8332>>)

**Maintainer** Merve Sahin <merve.sahn@gmail.com>

## Contents

add_1D_features . . . . .	3
add_2D_features . . . . .	4
add_hicpro_allvalidpairs_counts . . . . .	5
add_hicpro_matrix_counts . . . . .	6
add_hic_counts . . . . .	6
construct_features . . . . .	7
construct_features_chr . . . . .	8
construct_features_parallel . . . . .	10
expand_1D_features . . . . .	11
extract_hic_eigenvectors . . . . .	12
generate_binned_gi_list . . . . .	13
generate_bintolen_gi_list . . . . .	14
generate_df_gi_list . . . . .	15
get_chrs . . . . .	16
get_chr_sizes . . . . .	16
get_enzyme_cutsites . . . . .	17
gi_list2HTClist . . . . .	17
gi_list_binsize_detect . . . . .	18
gi_list_Dthreshold.detect . . . . .	19
gi_list_read . . . . .	19
gi_list_topdom . . . . .	20
gi_list_validate . . . . .	21
gi_list_write . . . . .	22
hic2icenorm_gi_list . . . . .	23
hicdc2hic . . . . .	24
hicdcdiff . . . . .	25
HiCDCPlus . . . . .	27
HiCDCPlus_chr . . . . .	28
HiCDCPlus_parallel . . . . .	30
HTClist2gi_list . . . . .	31
straw . . . . .	32
straw_dump . . . . .	33

**Index**

**34**

---

add_1D_features	<i>add_1D_features</i>
-----------------	------------------------

---

### Description

Adds 1D features to the `gi_list` instance. If any bin on `gi_list` overlaps with multiple feature records, feature values are aggregated for the bin according to the vector valued function `agg` (e.g., `sum`, `mean`)

### Usage

```
add_1D_features(gi_list, df, chrs = NULL, features = NULL, agg = mean)
```

### Arguments

<code>gi_list</code>	List of <code>GenomicInteractions</code> objects where each object named with chromosome contains intrachromosomal interaction information (see <code>?gi_list_validate</code> for a detailed explanation of valid <code>gi_list</code> instances).
<code>df</code>	<code>DataFrame</code> with columns named <code>'chr'</code> , and <code>'start'</code> and features to be added with their respective names.
<code>chrs</code>	a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to all chromosomes specified in the data frame <code>df</code> .
<code>features</code>	features to be added. Needs to be a subset of <code>colnames(df)</code> . Defaults to all columns in <code>df</code> other than <code>'chr'</code> , <code>'start'</code> , and <code>'end'</code> .
<code>agg</code>	any vector valued function with one data argument: defaults to <code>mean</code> .

### Value

a `gi_list` instance with 1D features stored in regions metadata handle of each list element (e.g., `gi_list[[1]]@regions@elementMetadata`) in the instance

### Examples

```
df<-data.frame(chr='chr9',start=seq(1e6,10e6,1e6),end=seq(2e6,11e6,1e6))
gi_list<-generate_df_gi_list(df)
feats<-data.frame(chr='chr9',start=seq(1e6,10e6,1e6),gc=runif(10))
gi_list<-add_1D_features(gi_list,feats)
```

---

add_2D_features	<i>add_2D_features</i>
-----------------	------------------------

---

## Description

Adds 2D features to a `gi_list` instance. If any bin on `gi_list` overlaps with multiple feature records, features are aggregated among matches according to the univariate vector valued function `agg` (e.g., `sum`, `mean`). For efficient use of memory, using `add/expand 1D features` (see `?add_1D_features` and `expand_1D_features`) in sequence is recommended instead of using `add_2D_features` directly for each chromosome.

## Usage

```
add_2D_features(gi, df, features = NULL, agg = sum)
```

## Arguments

<code>gi</code>	Element of a valid <code>gi_list</code> instance (restricted to a single chromosome e.g., <code>gi_list[['chr9']]</code> —see <code>?gi_list_validate</code> for a detailed explanation of valid <code>gi_list</code> instances).
<code>df</code>	data frame for a single chromosome containing columns named <code>chr</code> , <code>startI</code> and <code>startJ</code> and features to be added with their respective names (if <code>df</code> contains multiple chromosomes, you can convert it into a list of smaller data.frames for each chromosome and apply this function with <code>sapply</code> ).
<code>features</code>	features to be added. Needs to be subset of <code>colnames(df)</code> . Defaults to all columns in <code>df</code> other than <code>'chr'</code> , <code>'start'</code> , and <code>'end'</code> .
<code>agg</code>	any vector valued function with one data argument: defaults to <code>mean</code> .

## Value

a `gi_list` element with 2D features stored in metadata handle (i.e., `mcols(gi)`).

## Examples

```
df<-data.frame(chr='chr9',start=seq(1e6,10e6,1e6))
gi_list<-generate_df_gi_list(df,Dthreshold=500e3)
feats<-data.frame(chr='chr9',
startI=seq(1e6,10e6,1e6),startJ=seq(1e6,10e6,1e6),counts=rpois(10,lambda=5))
gi_list[['chr9']]<-add_2D_features(gi_list[['chr9']],feats)
```

---

```
add_hicpro_allvalidpairs_counts
    add_hicpro_allvalidpairs_counts
```

---

## Description

This function converts HiC-Pro outputs in allValidPairs format into a `gi_list` instance.

## Usage

```
add_hicpro_allvalidpairs_counts(
    gi_list,
    allvalidpairs_path,
    chrs = NULL,
    binned = TRUE,
    add_inter = FALSE
)
```

## Arguments

<code>gi_list</code>	valid <code>gi_list</code> instance. See <code>?gi_list_validate</code> for details. You can also detect whether a <code>gi_list</code> instance is uniformly binned, along with its bin size using <code>gi_list_binsize_detect</code> .
<code>allvalidpairs_path</code>	allValidPairsfile obtained from HiC-Pro (e.g., 'GSM2572593_con_rep1.allvalidPairs.txt')
<code>chrs</code>	a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to all chromosomes in the <code>gi_list</code> instance.
<code>binned</code>	TRUE if the <code>gi_list</code> instance is uniformly binned (helps faster execution). Defaults to TRUE.
<code>add_inter</code>	Interchromosomal interaction counts added as a 1D feature named 'inter' on regions metadata handle of each <code>gi_list</code> element (e.g., <code>gi_list[[1]]@regions@elementMetadata</code> or not; default FALSE

## Value

`gi_list` instance with counts on the metadata (e.g., `mcols(gi_list[[1]])` handle on each list element, and 'inter' on regions metadata handle of each element if `add_inter=TRUE`.

---

```
add_hicpro_matrix_counts
    add_hicpro_matrix_counts
```

---

### Description

This function converts HiC-Pro matrix and bed outputs into a `gi_list` instance.

### Usage

```
add_hicpro_matrix_counts(
    gi_list,
    absfile_path,
    matrixfile_path,
    chrs = NULL,
    add_inter = FALSE
)
```

### Arguments

<code>gi_list</code>	valid, uniformly binned <code>gi_list</code> instance. See <code>?gi_list_validate</code> and <code>gi_list_binsize_detect</code> for details.
<code>absfile_path</code>	absfile BED out of HiC-Pro (e.g., 'rawdata_10000_abs.bed')
<code>matrixfile_path</code>	matrix count file out of HiC-Pro (e.g., 'rawdata_10000.matrix')
<code>chrs</code>	a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to all chromosomes in the <code>gi_list</code> instance.
<code>add_inter</code>	Interchromosomal interaction counts added as a 1D feature named 'inter' on regions metadata handle of each <code>gi_list</code> element (e.g., <code>gi_list[[1]]@regions@elementMetadata</code> or not; default FALSE)

### Value

`gi_list` instance with counts on the metadata (e.g., `mcols(gi_list[[1]])` handle on each list element, and 'inter' on regions metadata handle of each element if `add_inter=TRUE`).

---

```
add_hic_counts    add_hic_counts
```

---

### Description

This function adds counts from a .hic file into a valid, binned, `gi_list` instance.

**Usage**

```
add_hic_counts(gi_list, hic_path, chrs = NULL, add_inter = FALSE)
```

**Arguments**

<code>gi_list</code>	valid, uniformly binned <code>gi_list</code> instance. See <code>?gi_list_validate</code> and <code>gi_list_binsize_detect</code> for details.
<code>hic_path</code>	path to the <code>.hic</code> file
<code>chrs</code>	a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to all chromosomes in the <code>gi_list</code> instance.
<code>add_inter</code>	Interchromosomal interaction counts added as a 1D feature named 'inter' on regions metadata handle of each <code>gi_list</code> element (e.g., <code>gi_list[[1]]@regions@elementMetadata</code> or not; default FALSE)

**Value**

`gi_list` instance with counts on the metadata (e.g., `mcols(gi_list[[1]])` handle on each list element, and 'inter' on regions metadata handle of each element if `add_inter=TRUE`).

**Examples**

```
gi_list<-generate_binned_gi_list(50e3, chrs='chr22')
gi_list<-add_hic_counts(gi_list,
  hic_path=system.file("extdata", "GSE63525_HMEC_combined_example.hic",
  package = "HiCDCPlus"))
```

---

construct\_features      *construct\_features*

---

**Description**

This function lists all restriction enzyme cutsites of a given genome and genome version with genomic features outlined in Carty et al. (2017) <https://www.nature.com/articles/ncomms15454>; GC content, mappability, and effective length

**Usage**

```
construct_features(
  output_path,
  gen = "Hsapiens",
  gen_ver = "hg19",
  sig = "GATC",
  bin_type = "Bins-uniform",
  binsize = 5000,
  wg_file = NULL,
  chrs = NULL,
  feature_type = "RE-based"
)
```

**Arguments**

output_path	the path to the folder and name prefix you want to place feature files into. The feature file will have the suffix '_bintolen.txt.gz'.
gen	name of the species: e.g., default 'Hsapiens'.
gen_ver	genomic assembly version: e.g., default 'hg19'.
sig	restriction enzyme cut pattern (or a vector of patterns; e.g., 'GATC' or c('GATC','GATC')).
bin_type	'Bins-uniform' if uniformly binned by binsize in bp, or 'Bins-RE-sites' if binned by number of restriction enzyme fragments.
binsize	binsize in bp if bin_type='Bins-uniform' (or number of RE fragment cut sites if bin_type='Bins-RE-sites'), defaults to 5000.
wg_file	path to the bigWig file containing mappability values across the genome of interest.
chrs	select a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to all chromosomes (except Y and M) in the genome specified.
feature_type	'RE-based' if features are to be computed based on restriction enzyme fragments. 'RE-agnostic' ignores restriction enzyme cutsite information and computes features gc and map based on binwide averages. bin_type has to be 'Bins-uniform' if feature_type='RE-agnostic'.

**Value**

a features 'bintolen' file that contains GC, mappability and length features.

**Examples**

```
outdir<-paste0(tempdir(check=TRUE),'/')
construct_features(output_path=outdir,gen='Hsapiens',
gen_ver='hg19',sig=c('GATC','GATC'),bin_type='Bins-uniform',binsize=100000,
wg_file=NULL,chrs=c('chr21'))
```

---

construct\_features\_chr

*construct\_features\_chr*

---

**Description**

This function lists all restriction enzyme cutsites of a given genome and genome version with genomic features outlined in Carty et al. (2017) for a single chromosome. <https://www.nature.com/articles/ncomms15454>; GC content, mappability, and effective length



**Usage**

```
construct_features_chr(
  chrom,
  gen = "Hsapiens",
  gen_ver = "hg19",
  sig = "GATC",
  bin_type = "Bins-uniform",
  binsize = 5000,
  wg_file = NULL,
  feature_type = "RE-based"
)
```

**Arguments**

chrom	select a chromosome.
gen	name of the species: e.g., default 'Hsapiens'.
gen_ver	genomic assembly version: e.g., default 'hg19'.
sig	restriction enzyme cut pattern (or a vector of patterns; e.g., 'GATC' or c('GATC','GATC')).
bin_type	'Bins-uniform' if uniformly binned by binsize in bp, or 'Bins-RE-sites' if binned by number of restriction enzyme fragments.
binsize	binsize in bp if bin_type='Bins-uniform' (or number of RE fragment cut sites if bin_type='Bins-RE-sites'), defaults to 5000.
wg_file	path to the bigWig file containing mappability values across the genome of interest.
feature_type	'RE-based' if features are to be computed based on restriction enzyme fragments. 'RE-agnostic' ignores restriction enzyme cutsite information and computes features gc and map based on binwide averages. bin_type has to be 'Bins-uniform' if feature_type='RE-agnostic'.

**Value**

a features 'bintolen' file that contains GC, mappability and length features.

**Examples**

```
df<-construct_features_chr(chrom='chr22',
  gen='Hsapiens', gen_ver='hg19',sig=c('GATC','GATC'),bin_type='Bins-uniform',
  binsize=100000,wg_file=NULL)
```

---

```
construct_features_parallel
    construct_features_parallel
```

---

### Description

This function lists all restriction enzyme cutsites of a given genome and genome version with genomic features outlined in Carty et al. (2017) <https://www.nature.com/articles/ncomms15454>; GC content, mappability, and effective length

### Usage

```
construct_features_parallel(
  output_path,
  gen = "Hsapiens",
  gen_ver = "hg19",
  sig = "GATC",
  bin_type = "Bins-uniform",
  binsize = 5000,
  wg_file = NULL,
  chrs = NULL,
  feature_type = "RE-based",
  ncore = NULL
)
```

### Arguments

output_path	the path to the folder and name prefix you want to place feature files into. The feature file will have the suffix '_bintolen.txt.gz'.
gen	name of the species: e.g., default 'Hsapiens'.
gen_ver	genomic assembly version: e.g., default 'hg19'.
sig	restriction enzyme cut pattern (or a vector of patterns; e.g., 'GATC' or c('GATC','GATC')).
bin_type	'Bins-uniform' if uniformly binned by binsize in bp, or 'Bins-RE-sites' if binned by number of restriction enzyme fragments.
binsize	binsize in bp if bin_type='Bins-uniform' (or number of RE fragment cut sites if bin_type='Bins-RE-sites'), defaults to 5000.
wg_file	path to the bigWig file containing mappability values across the genome of interest.
chrs	select a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to all chromosomes (except Y and M) in the genome specified.
feature_type	'RE-based' if features are to be computed based on restriction enzyme fragments. 'RE-agnostic' ignores restriction enzyme cutsite information and computes features gc and map based on binwide averages. bin_type has to be 'Bins-uniform' if feature_type='RE-agnostic'.
ncore	Number of cores to parallelize. Defaults to parallel::detectCores()-1.

**Value**

a features 'bintolen' file that contains GC, mappability and length features.

**Examples**

```
outdir<-paste0(tempdir(check=TRUE), '/')
construct_features_parallel(output_path=outdir, gen='Hsapiens',
gen_ver='hg19', sig=c('GATC', 'GATC'), bin_type='Bins-uniform', binsize=100000,
wg_file=NULL, chrs=c('chr21'), ncore=2)
```

---

expand_1D_features	<i>expand_1D_features</i>
--------------------	---------------------------

---

**Description**

Expands 1D features on the regions metadata handle of each list element (e.g., `gi_list[[1]]@regions@elementMetadata`) to the 2D metadata e.g., `mcols(gi_list[[1]])`. Two feature values corresponding to each anchor is summarized as a score using a vector valued function `agg` that takes two vector valued arguments of the same size and outputs a vector of the same size as the input vectors. This defaults to the `transform.vec` function outlined in (Carty et al., 2017). For efficient use of memory, using `add/expand 1D features` (see `?add_1D_features` and `expand_1D_features`) in sequence is recommended instead of using `add_2D_features` directly for each chromosome.

**Usage**

```
expand_1D_features(gi_list, chrs = NULL, features = NULL, agg = transform.vec)
```

**Arguments**

<code>gi_list</code>	List of <code>GenomicInteractions</code> objects where each object named with chromosomes contains intra-chromosomal interaction information (see <code>?gi_list_validate</code> for a detailed explanation of valid <code>gi_list</code> instances).
<code>chrs</code>	a subset of chromosomes' e.g., <code>c('chr21', 'chr22')</code> . Defaults to all chromosomes in the <code>gi_list</code> instance.
<code>features</code>	features to be added. Defaults to all 1D features in elements of <code>gi_list[[1]]@regions@elementMetadata</code> .
<code>agg</code>	any vector valued function with two data arguments: defaults to <code>transform.vec</code> described in HiC-DC (Carty et al., 2017).

**Value**

a `gi_list` element with 2D features stored in metadata handle (i.e., `mcols(gi)`).

**Examples**

```
df<-data.frame(chr='chr9',start=seq(1e6,10e6,1e6),end=seq(2e6,11e6,1e6))
gi_list<-generate_df_gi_list(df)
feats<-data.frame(chr='chr9',start=seq(1e6,10e6,1e6),gc=runif(10))
gi_list<-add_1D_features(gi_list,feats)
gi_list<-expand_1D_features(gi_list)
```

---

```
extract_hic_eigenvectors
```

```
extract_hic_eigenvectors
```

---

**Description**

This function uses Juicer command line tools to extract first eigenvectors across chromosomes from counts data in a .hic file and outputs them to text file of the structure chr start end score where the score column contains the eigenvector elements.

**Usage**

```
extract_hic_eigenvectors(
  hicfile,
  mode = "KR",
  binsize = 250e3,
  chrs = NULL,
  gen = "Hsapiens",
  gen_ver = "hg19"
)
```

**Arguments**

hicfile	path to the input .hic file.
mode	Normalization mode to extract first eigenvectors from Allowable options are: 'NONE' for raw (normalized counts if .hic file is written using hicdc2hic or hic2icenorm_gi_list), 'KR' for Knight-Ruiz normalization, 'VC' for Vanilla-Coverage normalization and 'VC_SQRT' for square root vanilla coverage. Defaults to 'KR'.
binsize	the uniform binning size for compartment scores in bp. Defaults to 100e3.
chrs	a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to all chromosomes except "Y", and "M" for the specified gen and gen_ver.
gen	name of the species: e.g., default 'Hsapiens'.
gen_ver	genomic assembly version: e.g., default 'hg19'.

**Value**

path to the eigenvector text files for each chromosome containing chromosome, start, end and compartment score values that may need to be flipped signs for each chromosome. File paths follow gsub('.hic', '\_<chromosome>\_eigenvectors.txt',hicfile)

**Examples**

```
eigenvector_filepaths<-extract_hic_eigenvectors(
  hicfile=system.file("extdata", "eigenvector_example.hic",
  package = "HiCDCPlus"),
  chrs=c("chr22"),binsize=250e3,mode="NONE")
```

---

```
generate_binned_gi_list
      generate_binned_gi_list
```

---

**Description**

Generates a valid uniformly binned gi\_list instance.

**Usage**

```
generate_binned_gi_list(
  binsize,
  chrs = NULL,
  Dthreshold = 2e+06,
  gen = "Hsapiens",
  gen_ver = "hg19"
)
```

**Arguments**

binsize	Desired binsize in bp, e.g., 5000, 25000.
chrs	a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to all chromosomes except "Y", and "M" for the specified gen and gen_ver.
Dthreshold	maximum distance (included) to check for significant interactions, defaults to 2e6 or maximum in the data; whichever is smaller.
gen	name of the species: e.g., default 'Hsapiens'.
gen_ver	genomic assembly version: e.g., default 'hg19'.

**Value**

a valid, uniformly binned gi\_list instance.

**Examples**

```
gi_list<-generate_binned_gi_list(1e6, chrs='chr22')
```

---

```
generate_bintolen_gi_list
      generate_bintolen_gi_list
```

---

### Description

Generates a `gi_list` instance from a bintolen file generated by `generate.features` (see `?generate.features` for details).

### Usage

```
generate_bintolen_gi_list(
  bintolen_path,
  chrs = NULL,
  Dthreshold = 2e+06,
  binned = TRUE,
  binsize = NULL,
  gen = "Hsapiens",
  gen_ver = "hg19"
)
```

### Arguments

<code>bintolen_path</code>	path to the flat file containing columns named bins and features
<code>chrs</code>	select a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to all chromosomes specified in the bintolen file.
<code>Dthreshold</code>	maximum distance (included) to check for significant interactions, defaults to 2e6 or maximum in the data; whichever is smaller.
<code>binned</code>	TRUE if the bintolen file is uniformly binned. Defaults to TRUE.
<code>binsize</code>	bin size in bp to be generated for the object. Defaults to the binsize in the bintolen file, if exists.
<code>gen</code>	name of the species: e.g., default 'Hsapiens'
<code>gen_ver</code>	genomic assembly version: e.g., default 'hg19'

### Value

a valid `gi_list` instance with genomic features derived from specified restriction enzyme cut patterns when generating the bintolen file using `construct_features` (see `?construct_features` for help). Genomic 1D features are stored in the regions metadata handle of each list element (e.g., `gi_list[[1]]@regions@elementMetadata`).

### Examples

```
chrs<-'chr22'
bintolen_path<-system.file("extdata", "test_bintolen.txt.gz",
  package = "HiCDCPlus")
gi_list<-generate_bintolen_gi_list(bintolen_path,chrs)
```

---

generate\_df\_gi\_list    *generate\_df\_gi\_list*

---

## Description

Generates a `gi_list` instance from a data frame object describing the regions.

## Usage

```
generate_df_gi_list(  
  df,  
  chrs = NULL,  
  Dthreshold = 2e+06,  
  gen = "Hsapiens",  
  gen_ver = "hg19"  
)
```

## Arguments

<code>df</code>	DataFrame with columns named 'chr', 'start', (and optionally 'end', if the regions have gaps) and 1D features with their respective column names.
<code>chrs</code>	select a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to all chromosomes specified in <code>df</code> .
<code>Dthreshold</code>	maximum distance (included) to check for significant interactions, defaults to 2e6 or maximum in the data, whichever is smaller.
<code>gen</code>	name of the species: e.g., default 'Hsapiens'
<code>gen_ver</code>	genomic assembly version: e.g., default 'hg19'

## Value

a valid `gi_list` instance with genomic features supplied from `df`. Genomic 1D features are stored in the regions metadata handle of each list element (e.g., `gi_list[[1]]@regions@elementMetadata`).

## Examples

```
df<-data.frame(chr='chr9',start=seq(1e6,10e6,1e6))  
gi_list<-generate_df_gi_list(df)
```

---

get_chrs	<i>get_chrs</i>
----------	-----------------

---

**Description**

This function finds all chromosomes of a given genome and genome version except for Y and M.

**Usage**

```
get_chrs(gen = "Hsapiens", gen_ver = "hg19")
```

**Arguments**

gen	name of the species: e.g., default 'Hsapiens'
gen_ver	genomic assembly version: e.g., default 'hg19'

**Value**

string vector of chromosomes.

**Examples**

```
get_chrs('Hsapiens', 'hg19')
```

---

get_chr_sizes	<i>get_chr_sizes</i>
---------------	----------------------

---

**Description**

This function finds all chromosome sizes of a given genome, genome version and set of chromosomes.

**Usage**

```
get_chr_sizes(gen = "Hsapiens", gen_ver = "hg19", chrs = NULL)
```

**Arguments**

gen	name of the species: e.g., default 'Hsapiens'
gen_ver	genomic assembly version: e.g., default 'hg19'
chrs	select a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to all chromosomes (except Y and M) in the genome specified.

**Value**

named vector containing names as chromosomes and values as chromosome sizes.



**Examples**

```
get_chr_sizes('Hsapiens', 'hg19', c('chr21', 'chr22'))
```

---

```
get_enzyme_cutsites     get_enzyme_cutsites
```

---

**Description**

This function finds all restriction enzyme cutsites of a given genome, genome version, and set of cut patterns

**Usage**

```
get_enzyme_cutsites(sig, gen = "Hsapiens", gen_ver = "hg19", chrs = NULL)
```

**Arguments**

sig	a set of restriction enzyme cut patterns (e.g., 'GATC' or c('GATC','GATC'))
gen	name of the species: e.g., default 'Hsapiens'
gen_ver	genomic assembly version: e.g., default 'hg19'
chrs	a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to all chromosomes (except Y and M) in the genome specified by gen and gen_ver.

**Value**

list of chromosomes.

**Examples**

```
get_enzyme_cutsites(gen='Hsapiens', gen_ver='hg19',
  sig=c('GATC', 'GATC'), chrs=c('chr22'))
```

---

```
gi_list2HTClist     gi_list2HTClist
```

---

**Description**

This function converts a gi\_list instance into a HTClist instance compatible for use with the R Bioconductor package HiTC <https://bioconductor.org/packages/HiTC/>

**Usage**

```
gi_list2HTClist(gi_list, chrs = NULL)
```

**Arguments**

<code>gi_list</code>	List of GenomicInteractions objects with a counts column where each object named with chromosomes contains intra-chromosomal interaction information (minimally containing counts and genomic distance in <code>mcols(gi_list)</code> — see <code>?gi_list_validate</code> for a detailed explanation of valid <code>gi_list</code> instances).
<code>chrs</code>	select a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to chromosomes in <code>gi_list</code> .

**Value**

a HTClist instance compatible for use with HiTC

**Examples**

```
gi_list<-generate_binned_gi_list(50e3, chrs=c('chr22'))
gi_list<-add_hic_counts(gi_list,
  hic_path<-system.file("extdata", "GSE63525_HMEC_combined_example.hic",
  package = "HiCDCPlus"))
htc_list<-gi_list2HTClist(gi_list)
```

---

`gi_list_binsize_detect`

*gi\_list\_binsize\_detect*

---

**Description**

This function finds the bin size of a uniformly binned valid `gi_list` instance in bp. It raises an error if the `gi_list` instance is not uniformly binned.

**Usage**

```
gi_list_binsize_detect(gi_list)
```

**Arguments**

<code>gi_list</code>	<code>gi_list</code> object to be verified. In order to pass without errors, a <code>gi_list</code> object (1) has to be a list of <code>InteractionSet::GInteractions</code> objects, (2) each list element has to be named as chromosomes and only contain intra-chromosomal interaction information, (3) <code>mcols(.)</code> for each list element should at least contain pairwise genomic distances in a column named 'D' and (4) each list element needs to be uniformly binned
----------------------	---

**Value**

uniform binsize in base pairs or an error if the `gi_list` instance is not uniformly binned.

**Examples**

```
gi_list<-generate_binned_gi_list(1e6, chrs='chr22')
gi_list_binsize_detect(gi_list)
```

---

```
gi_list_Dthreshold.detect
      gi_list_Dthreshold_detect
```

---

**Description**

This function finds the maximum genomic distance in a valid *gi\_list* object.

**Usage**

```
gi_list_Dthreshold.detect(gi_list)
```

**Arguments**

*gi\_list*            A valid *gi\_list* instance. See `?gi_list_validate` for more details about the attributes of a valid *gi\_list* instance.

**Value**

maximum genomic distance in the object

**Examples**

```
gi_list<-generate_binned_gi_list(1e6, chrs='chr22')
gi_list_Dthreshold.detect(gi_list)
```

---

```
gi_list_read            gi_list_read
```

---

**Description**

Reads a written *gi\_list* instance using `gi_list_write` into a valid *gi\_list* instance.

**Usage**

```
gi_list_read(
  fname,
  chrs = NULL,
  Dthreshold = NULL,
  features = NULL,
  gen = "Hsapiens",
  gen_ver = "hg19"
)
```

**Arguments**

<code>fname</code>	path to the file to read from (can end with <code>.txt</code> , <code>.rds</code> , or <code>.txt.gz</code> ).
<code>chrs</code>	select a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to all chromosomes contained in the <code>fname</code> .
<code>Dthreshold</code>	maximum distance (included) to check for significant interactions, defaults to the maximum in the data.
<code>features</code>	Select the subset of features (1-D or 2-D) to be added to the <code>gi_list</code> instance (without the trailing I or J), defaults to all features (score column gets ingested as 'score').
<code>gen</code>	name of the species: e.g., default 'Hsapiens'
<code>gen_ver</code>	genomic assembly version: e.g., default 'hg19'

**Value**

A valid `gi_list` instance with 1D features stored in regions metadata handle of each list element (e.g., `gi_list[[1]]@regions@elementMetadata`) in the instance and with 2D features stored in metadata handle (i.e., `mcols(gi)`).

**Examples**

```
outputdir<-paste0(tempdir(check=TRUE), '/')
gi_list<-generate_binned_gi_list(1e6, chrs='chr22')
gi_list_write(gi_list, paste0(outputdir, 'testgiread.txt'))
gi_list2<-gi_list_read(paste0(outputdir, 'testgiread.txt'))
```

---

`gi_list_topdom`

*gi\_list\_topdom*

---

**Description**

This function converts a `gi_list` instance with ICE normalized counts into TAD annotations through an implementation of TopDom v0.0.2 (<https://github.com/HenrikBengtsson/TopDom>) adapted as TopDom at this package. If you're using this function, please cite TopDom according to the documentation at <https://github.com/HenrikBengtsson/TopDom/blob/0.0.2/docs/>

**Usage**

```
gi_list_topdom(
  gi_list,
  chrs = NULL,
  file_out = FALSE,
  fpath = NULL,
  window.size = 5,
  verbose = FALSE
)
```

**Arguments**

gi_list	List of GenomicInteractions objects where each object named with chromosomes contains intrachromosomal interaction information (see ?gi_list_validate for a detailed explanation of valid gi_list instances).
chrs	select a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to chromosomes in gi_list.
file_out	If true, outputs TAD annotations into files with paths beginning with fpath. Defaults to FALSE
fpath	Outputs TAD annotations into files with paths beginning in fpath.
window.size	integer, number of bins to extend. Defaults to 5.
verbose	TRUE if you would like to troubleshoot TopDom.

**Value**

a list instance with TAD annotation reporting for each chromosome

**Examples**

```
hic_path<-system.file("extdata", "GSE63525_HMEC_combined_example.hic",
package = "HiCDCPlus")
gi_list<-hic2icenorm_gi_list(hic_path,binsize=50e3,chrs='chr22')
tads<-gi_list_topdom(gi_list)
```

---

gi_list_validate	<i>gi_list_validate</i>
------------------	-------------------------

---

**Description**

This function validates a gi\_list instance.

**Usage**

```
gi_list_validate(gi_list)
```

**Arguments**

gi_list	gi_list object to be verified. In order to pass without errors, a gi_list object (1) has to be a list of InteractionSet::GInteractions objects, (2) each list element has to be named as chromosomes and only contain intra-chromosomal interaction information, (3) mcols(.) for each list element should at least contain pairwise genomic distances in a column named 'D'.
---------	---

**Value**

invisible value if the gi\_list instance is valid. Otherwise, an error is raised.

**Examples**

```
gi_list<-generate_binned_gi_list(1e6, chrs='chr22')
gi_list_validate(gi_list)
```

---

```
gi_list_write      gi_list_write
```

---

**Description**

Writes a valid *gi\_list* instance into a file.

**Usage**

```
gi_list_write(
  gi_list,
  fname,
  chrs = NULL,
  columns = "minimal",
  rows = "all",
  significance_threshold = 0.05,
  score = NULL
)
```

**Arguments**

<code>gi_list</code>	List of <code>GenomicInteractions</code> objects where each object named with chromosomes contains intra-chromosomal interaction information (see <code>?gi_list_validate</code> for a detailed explanation of valid <i>gi_list</i> instances).
<code>fname</code>	path to the file to write to (can end with <code>.txt</code> , or <code>.txt.gz</code> ).
<code>chrs</code>	select a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to all chromosomes in the <i>gi_list</i> .
<code>columns</code>	Can be <code>'minimal'</code> , which is just distance and counts (and <code>HiCDCPlus</code> result columns <code>'qvalue'</code> , <code>'pvalue'</code> , <code>'mu'</code> , and <code>'sdev'</code> , if exists; see <code>?HiCDCPlus</code> ) information, <code>'minimal_plus_features'</code> , which is distance, counts, and other calculated 2D features, <code>'minimal_plus_score'</code> , which generates a <code>.hic</code> pre compatible text file, or <code>'all'</code> , which is distance, counts, calculated 2D features, as well as all 1D features. Defaults to <code>'minimal'</code> .
<code>rows</code>	Can be <code>'all'</code> or <code>'significant'</code> , which filters rows according to FDR adjusted pvalue column <code>'qvalue'</code> (this has to exist in <code>mcols(.)</code> ) at <code>significance_threshold</code> . Defaults to <code>'all'</code> .
<code>significance_threshold</code>	Row filtering threshold on <code>'qvalue'</code> . Defaults to 0.05.
<code>score</code>	Score column to extract to <code>.hic</code> pre compatible file. See mode options in <code>?hicdc2hic</code> for more details.

**Value**

a tab separated flat file concatenating all intra-chromosomal interaction information.

**Examples**

```
outputdir<-paste0(tempdir(check=TRUE),'/')
gi_list<-generate_binned_gi_list(1e6,chr='chr22')
gi_list_write(gi_list,paste0(outputdir,'test.txt'))
```

---

hic2icenorm\_gi\_list    *hic2icenorm\_gi\_list*

---

**Description**

This function converts a .hic file into a gi\_list instance with ICE normalized counts on the counts column for TAD annotation using a copy of TopDom (see ?TopDom\_0.0.2) as well as an (optional) .hic file with ICE normalized counts for visualization with Juicebox. This function requires installing the Bioconductor package HiTC.

**Usage**

```
hic2icenorm_gi_list(
  hic_path,
  binsize = 50000,
  chrs = NULL,
  hic_output = FALSE,
  gen = "Hsapiens",
  gen_ver = "hg19",
  Dthreshold = Inf
)
```

**Arguments**

hic_path	Path to the .hic file.
binsize	Desired bin size in bp (default 50000).
chrs	select a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to chromosomes in gen and gen_ver except 'chrY' and 'chrM'.
hic_output	If TRUE, a .hic file with the name gsub("\.hic\$", "_icenorm.hic", hic_path) is generated containing the ICE normalized counts under 'NONE' normalization.
gen	name of the species: e.g., default 'Hsapiens'
gen_ver	genomic assembly version: e.g., default 'hg19'
Dthreshold	maximum distance (included) to check for significant interactions, defaults to maximum in the data.

**Value**

a thresholded `gi_list` instance with ICE normalized intra-chromosomal counts for further use with this package, `HiCDCPlus`.

**Examples**

```
hic_path<-system.file("extdata", "GSE63525_HMEC_combined_example.hic",
package = "HiCDCPlus")
gi_list=hic2icenorm_gi_list(hic_path,binsize=50e3,chr=c('chr22'))
```

---

hicdc2hic

*hicdc2hic*

---

**Description**

This function converts various modes from `HiCDCPlus gi_list` (uniformly binned) instance back into a `.hic` file with the mode passed as counts that can be retrieved using `Juicer Dump` (<https://github.com/aidenlab/juicer/wiki/Extraction>) with 'NONE' normalization.

**Usage**

```
hicdc2hic(
  gi_list,
  hicfile,
  mode = "normcounts",
  chrs = NULL,
  gen_ver = "hg19",
  memory = 8
)
```

**Arguments**

<code>gi_list</code>	List of <code>GenomicInteractions</code> objects where each object named with chromosomes contains intra-chromosomal interaction information (minimally containing counts and genomic distance in <code>mcols(gi_list)</code> — see <code>?gi_list_validate</code> for a detailed explanation of valid <code>gi_list</code> instances).
<code>hicfile</code>	the path to the <code>.hic</code> file
<code>mode</code>	What to put to the <code>.hic</code> file as score. Allowable options are: 'pvalue' for $-\log_{10}$ significance p-value, 'qvalue' for $-\log_{10}$ FDR corrected p-value, 'normcounts' for raw counts/expected counts, and 'zvalue' for standardized counts (raw counts-expected counts)/modeled standard deviation of expected counts and 'raw' to pass-through 'raw counts. Defaults to 'normcounts'.
<code>chrs</code>	select a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to chromosomes in <code>gi_list</code> .
<code>gen_ver</code>	genomic assembly version: e.g., default 'hg19'
<code>memory</code>	Java memory to generate <code>.hic</code> files. Defaults to 8. Up to 64 is recommended for higher resolutions.



**Value**

path of the .hic file.

**Examples**

```
outdir<-paste0(tempdir(check=TRUE), '/')
gi_list<-generate_binned_gi_list(50e3, chrs='chr22')
gi_list<-add_hic_counts(gi_list,
  hic_path=system.file("extdata", "GSE63525_HMEC_combined_example.hic",
  package = "HiCDCPlus"))
hicdc2hic(gi_list, hicfile=paste0(outdir, 'out.hic'),
  mode='raw')
```

---

hicdcdiff

*hicdcdiff*

---

**Description**

This function calculates differential interactions for a set of chromosomes across conditions and replicates. You need to install DESeq2 from Bioconductor to use this function.

**Usage**

```
hicdcdiff(
  input_paths,
  filter_file,
  output_path,
  bin_type = "Bins-uniform",
  binsize = 5000,
  granularity = 5000,
  chrs = NULL,
  Dmin = 0,
  Dmax = 2e+06,
  diagnostics = FALSE,
  DESeq.save = FALSE,
  fitType = "local"
)
```

**Arguments**

input_paths	a list with names as condition names and values as paths to gi_list RDS objects (see ?gi_list_validate for a detailed explanation of valid gi_list instances) saved with saveRDS or paths to .hic files for each replicate. e.g., list(CTCF=c('~Downloads/GM_CTCF_rep1_MAPQ30_10kb.rds', '~Downloads/GM_CTCF_rep2_MAPQ30_10kb.rds'))
filter_file	path to the text file containing columns chr', startI, and startJ denoting the name of the chromosomes and starting coordinates of 2D interaction bins to be compared across conditions, respectively.

output_path	the path to the folder and name prefix you want to place DESeq-processed matrices (in a .txt file), plots (if diagnostics=TRUE) and DESeq2 objects (if DESeq.save=TRUE). Files will be generated for each chromosome.
bin_type	'Bins-uniform' if uniformly binned by binsize in bp, or 'Bins-RE-sites' if binned by number of restriction enzyme fragment cutsites!
binsize	binsize in bp if bin_type='Bins-uniform' (or number of RE fragments if bin_type='Bins-RE-sites'), e.g., default 5000
granularity	Desired distance granularity to base dispersion parameters on in bp. For uniformly binned analysis (i.e., bin_type=='Bins-uniform'), this defaults to the bin size. Otherwise, it is 5000.
chrs	select a subset of chromosomes' e.g., c('chr21','chr22'). Defaults to all chromosomes (except Y and M) in the filter_file.
Dmin	minimum distance (included) to check for significant interactions, defaults to 0. Put Dmin=1 to ignore D=0 bins in calculating normalization factors.
Dmax	maximum distance (included) to check for significant interactions, defaults to 2e6 or maximum in the data; whichever is minimum.
diagnostics	if TRUE, generates diagnostic plots of the normalization factors, geometric means of such factors by distance bin, as well as MA Plots (see DESeq documentation for details about MA plots). Defaults to FALSE.
DESeq.save	if TRUE, saves the DESeq objects for each chromosome as an .rds file in the output_path. Defaults to FALSE.
fitType	follows fitType in DESeq2::estimateDispersions. Allowable options are 'parametric' (parametric regression), 'local' (local regression), and 'mean' (constant across interaction bins). Default is 'local'.

## Value

paths of a list of three entities. outputpaths will have differential bins among those in filter\_file. deseq2paths will have the DESeq2 object stored as an .rds file. Available if DESeq.save=TRUE p1otpaths will have diagnostic plots (e.g., MA, dispersion, PCA) if diagnostics=TRUE.

## Examples

```
outputdir<-paste0(tempdir(check=TRUE), '/')
hicdcdiff(input_paths=list(NSD2=c(
  system.file("extdata", "GSE131651_NSD2_LOW_arima_example.hic",
  package = "HiCDCPlus"),
  system.file("extdata", "GSE131651_NSD2_HIGH_arima_example.hic",
  package = "HiCDCPlus")),
  TKO=c(system.file("extdata", "GSE131651_TKOCTCF_new_example.hic",
  package = "HiCDCPlus"),
  system.file("extdata", "GSE131651_NTKOCTCF_new_example.hic",
  package = "HiCDCPlus"))),
  filter_file=system.file("extdata", "GSE131651_analysis_indices.txt.gz",
  package = "HiCDCPlus"),
  chrs='chr22',
  output_path=outputdir,
```

```

fitType = 'mean',
binsize=50000,
diagnostics=FALSE)

```

---

HiCDCPlus

*HiCDCPlus*


---

## Description

This function finds significant interactions in a HiC-DC readable matrix and expresses statistical significance of counts through the following: 'pvalue': significance  $P$ -value, 'qvalue': FDR corrected  $P$ -value, 'mu': expected counts, 'sdev': modeled standard deviation of expected counts.

## Usage

```

HiCDCPlus(
  gi_list,
  covariates = NULL,
  chrs = NULL,
  distance_type = "spline",
  model_distribution = "nb",
  binned = TRUE,
  df = 6,
  Dmin = 0,
  Dmax = 2e+06,
  ssize = 0.01,
  splineknottting = "uniform",
  model_filepath = NULL
)

```

## Arguments

<code>gi_list</code>	List of GenomicInteractions objects where each object named with chromosomes contains intrachromosomal interaction information (minimally containing counts and genomic distance in <code>mcols(gi_list[[1]])</code> —see <code>?gi_list_validate</code> for a detailed explanation of valid <code>gi_list</code> instances).
<code>covariates</code>	covariates to be considered in addition to genomic distance $D$ . Defaults to all covariates besides 'D', 'counts', 'mu', 'sdev', 'pvalue', 'qvalue' in <code>mcols(gi_list[[1]])</code>
<code>chrs</code>	select a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to all chromosomes in the <code>gi_list</code> .
<code>distance_type</code>	distance covariate form: 'spline' or 'log'. Defaults to 'spline'.
<code>model_distribution</code>	'nb' uses a Negative Binomial model, 'nb_vardisp' uses a Negative Binomial model with a distance specific dispersion parameter inferred from the data, 'nb_hurdle' uses the legacy HiCDC model.

binned	TRUE if uniformly binned or FALSE if binned by restriction enzyme fragment cutsites
df	degrees of freedom for the genomic distance spline function if distance_type='spline'. Defaults to 6, which corresponds to a cubic spline as explained in Carty et al. (2017)
Dmin	minimum distance (included) to check for significant interactions, defaults to 0
Dmax	maximum distance (included) to check for significant interactions, defaults to 2e6 or maximum in the data; whichever is minimum.
ssize	Distance stratified sampling size. Can decrease for large chromosomes. Increase recommended if model fails to converge. Defaults to 0.01.
splineknotting	Spline knotting strategy. Either "uniform", uniformly spaced in distance, or placed based on distance distribution of counts "count-based" (i.e., more closely spaced where counts are more dense).
model_filepath	Outputs fitted HiC-DC model object as an .rds file per chromosome. Defaults to NULL (no output).

### Value

A valid `gi_list` instance with additional `mcols(.)` for each chromosome: `pvalue`: significance  $P$ -value, `qvalue`: FDR corrected  $P$ -value, `mu`: expected counts, `sdev`: modeled standard deviation of expected counts.

### Examples

```
gi_list<-generate_binned_gi_list(50e3, chrs='chr22')
gi_list<-add_hic_counts(gi_list,
  hic_path<-system.file("extdata", "GSE63525_HMEC_combined_example.hic",
  package = "HiCDCPlus"))
gi_list<-HiCDCPlus(gi_list)
```

---

HiCDCPlus\_chr

*HiCDCPlus\_chr*

---

### Description

This function finds significant interactions in a HiC-DC readable matrix restricted to a single chromosome and expresses statistical significance of counts through the following: `pvalue`: significance  $P$ -value, `qvalue`: FDR corrected  $P$ -value, `mu`: expected counts, `sdev`: modeled standard deviation of expected counts.

### Usage

```
HiCDCPlus_chr(
  gi,
  covariates = NULL,
  distance_type = "spline",
```

```

model_distribution = "nb",
binned = TRUE,
df = 6,
Dmin = 0,
Dmax = 2e+06,
ssize = 0.01,
splineknotting = "uniform",
model_filepath = NULL
)

```

### Arguments

<code>gi</code>	Instance of a single chromosome <code>GenomicInteractions</code> object containing intra-chromosomal interaction information (minimally containing counts and genomic distance).
<code>covariates</code>	covariates to be considered in addition to genomic distance <code>D</code> . Defaults to all covariates besides <code>'D'</code> , <code>'counts'</code> , <code>'mu'</code> , <code>'sdev'</code> , <code>'pvalue'</code> , <code>'qvalue'</code> in <code>mcols(gi)</code>
<code>distance_type</code>	distance covariate form: <code>'spline'</code> or <code>'log'</code> . Defaults to <code>'spline'</code> .
<code>model_distribution</code>	<code>'nb'</code> uses a Negative Binomial model, <code>'nb_vardisp'</code> uses a Negative Binomial model with a distance specific dispersion parameter inferred from the data, <code>'nb_hurdle'</code> uses the legacy HiC-DC model.
<code>binned</code>	TRUE if uniformly binned or FALSE if binned by restriction enzyme fragment cut sites.
<code>df</code>	degrees of freedom for the genomic distance spline function if <code>distance_type='spline'</code> . Defaults to 6, which corresponds to a cubic spline as explained in Carty et al. (2017)
<code>Dmin</code>	minimum distance (included) to check for significant interactions, defaults to 0
<code>Dmax</code>	maximum distance (included) to check for significant interactions, defaults to 2e6 or maximum in the data; whichever is minimum.
<code>ssize</code>	Distance stratified sampling size. Can decrease for large chromosomes. Increase recommended if model fails to converge. Defaults to 0.01.
<code>splineknotting</code>	Spline knotting strategy. Either <code>"uniform"</code> , uniformly spaced in distance, or placed based on distance distribution of counts <code>"count-based"</code> (i.e., more closely spaced where counts are more dense).
<code>model_filepath</code>	Outputs fitted HiC-DC model object as an <code>.rds</code> file with chromosome name indicated on it. Defaults to NULL (no output).

### Value

A valid `gi` instance with additional `mcols(.)`: `'pvalue'`: significance *P*-value, `'qvalue'`: FDR corrected *P*-value, `'mu'`: expected counts, `'sdev'`: modeled standard deviation of expected counts.

**Examples**

```

gi_list<-generate_binned_gi_list(50e3,chr='chr22')
gi_list<-add_hic_counts(gi_list,
  hic_path<-system.file("extdata", "GSE63525_HMEC_combined_example.hic",
  package = "HiCDCPlus"))
gi<-HiCDCPlus_chr(gi_list[[1]])

```

---

HiCDCPlus\_parallel      *HiCDCPlus\_parallel*

---

**Description**

This function finds significant interactions in a HiC-DC readable matrix and expresses statistical significance of counts through the following with a parallel implementation (using sockets; compatible with Windows): 'pvalue': significance *P*-value, 'qvalue': FDR corrected *P*-value, 'mu': expected counts, 'sdev': modeled standard deviation of expected counts.

**Usage**

```

HiCDCPlus_parallel(
  gi_list,
  covariates = NULL,
  chrs = NULL,
  distance_type = "spline",
  model_distribution = "nb",
  binned = TRUE,
  df = 6,
  Dmin = 0,
  Dmax = 2e+06,
  ssize = 0.01,
  splineknotting = "uniform",
  ncore = NULL
)

```

**Arguments**

<code>gi_list</code>	List of GenomicInteractions objects where each object named with chromosome contains intrachromosomal interaction information (minimally containing counts and genomic distance in <code>mcols(gi_list[[1]])</code> —see <code>?gi_list_validate</code> for a detailed explanation of valid <code>gi_list</code> instances).
<code>covariates</code>	covariates to be considered in addition to genomic distance <i>D</i> . Defaults to all covariates besides 'D', 'counts', 'mu', 'sdev', 'pvalue', 'qvalue' in <code>mcols(gi)</code>
<code>chrs</code>	select a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to all chromosomes in the <code>gi_list</code> .
<code>distance_type</code>	distance covariate form: 'spline' or 'log'. Defaults to 'spline'.

model_distribution	'nb' uses a Negative Binomial model, 'nb_vardisp' uses a Negative Binomial model with a distance specific dispersion parameter inferred from the data, 'nb_hurdle' uses the legacy HiC-DC model.
binned	TRUE if uniformly binned or FALSE if binned by restriction enzyme fragment cutsites
df	degrees of freedom for the genomic distance spline function if distance_type='spline'. Defaults to 6, which corresponds to a cubic spline as explained in Carty et al. (2017)
Dmin	minimum distance (included) to check for significant interactions, defaults to 0
Dmax	maximum distance (included) to check for significant interactions, defaults to 2e6 or maximum in the data; whichever is minimum.
ssize	Distance stratified sampling size. Can decrease for large chromosomes. Increase recommended if model fails to converge. Defaults to 0.01.
splineknotting	Spline knotting strategy. Either "uniform", uniformly spaced in distance, or placed based on distance distribution of counts "count-based" (i.e., more closely spaced where counts are more dense).
ncore	Number of cores to parallelize. Defaults to parallel::detectCores()-1.

### Value

A valid `gi_list` instance with additional `mcols(.)` for each chromosome: `pvalue`: significance  $P$ -value, `qvalue`: FDR corrected  $P$ -value, `mu`: expected counts, `sdev`: modeled standard deviation of expected counts.

### Examples

```
gi_list<-generate_binned_gi_list(50e3,chr='chr22')
gi_list<-add_hic_counts(gi_list,
  hic_path=system.file("extdata", "GSE63525_HMEC_combined_example.hic",
  package = "HiCDCPlus"))
gi<-HiCDCPlus_parallel(gi_list,ncore=1)
```

---

HTClist2gi\_list

*HTClist2gi\_list*


---

### Description

This function converts a `HTClist` instance into a `gi_list` instance with counts for further use with this package, `HiCDCPlus`

### Usage

```
HTClist2gi_list(htc_list, chrs = NULL, Dthreshold = 2e+06)
```

**Arguments**

<code>htc_list</code>	A valid HTCList instance (see vignette("HiTC"))
<code>chrs</code>	select a subset of chromosomes' e.g., <code>c('chr21','chr22')</code> . Defaults to chromosomes in <code>htc_list</code> .
<code>Dthreshold</code>	maximum distance (included) to check for significant interactions, defaults to <code>2e6</code> or maximum in the data; whichever is smaller.

**Value**

a thresholded `gi_list` instance with intra-chromosomal counts for further use with HiCDCPlus

**Examples**

```
gi_list<-generate_binned_gi_list(50e3, chrs=c('chr22'))
gi_list<-add_hic_counts(gi_list,
  hic_path=system.file("extdata", "GSE63525_HMEC_combined_example.hic",
  package = "HiCDCPlus"))
htc_list<-gi_list2HTCList(gi_list)
gi_list2<-HTCList2gi_list(htc_list, Dthreshold=Inf)
```

---

 straw

---

*straw*


---

**Description**

Adapted C++ implementation of Juicer's dump. Reads the .hic file, finds the appropriate matrix and slice of data, and outputs as an R DataFrame.

**Usage**

```
straw(norm, fn, ch1, ch2, u, bs)
```

**Arguments**

<code>norm</code>	Normalization to apply. Must be one of NONE/VC/VC_SQRT/KR. VC is vanilla coverage, VC_SQRT is square root of vanilla coverage, and KR is Knight-Ruiz or Balanced normalization.
<code>fn</code>	path to the .hic file
<code>ch1</code>	first chromosome location (e.g., "1")
<code>ch2</code>	second chromosome location (e.g., "8")
<code>u</code>	BP (BasePair) or FRAG (restriction enzyme FRAGment)
<code>bs</code>	The bin size. By default, for BP, this is one of <2500000, 1000000, 500000, 250000, 100000, 50000, 25000, 10000, 5000> and for FRAG this is one of <500, 200, 100, 50, 20, 5, 2, 1>.



**Details**

Usage: straw <NONE/VC/VC\_SQRT/KR> <hicFile(s)> <chr1>[:x1:x2] <chr2>[:y1:y2] <BP/FRAG> <binsize>

**Value**

Data.frame of a sparse matrix of data from hic file. x,y,counts

---

straw\_dump

*straw\_dump*


---

**Description**

Interface for Juicer's dump in case C++ straw fails (known to fail on Windows due to zlib compression not being OS agnostic and particularly not preserving null bytes, which .hic files are delimited with). This function reads the .hic file, finds the appropriate matrix and slice of data, writes it to a temp file, reads and modifies it, and outputs as an R Dataframe (and also deletes the temp file).

**Usage**

```
straw_dump(norm, fn, ch1, ch2, u, bs)
```

**Arguments**

norm	Normalization to apply. Must be one of NONE/VC/VC_SQRT/KR. VC is vanilla coverage, VC_SQRT is square root of vanilla coverage, and KR is Knight-Ruiz or Balanced normalization.
fn	path to the .hic file
ch1	first chromosome location (e.g., "1")
ch2	second chromosome location (e.g., "8")
u	BP (BasePair) or FRAG (restriction enzyme FRAGMENT)
bs	The bin size. By default, for BP, this is one of <2500000, 1000000, 500000, 250000, 100000, 50000, 25000, 10000, 5000> and for FRAG this is one of <500, 200, 100, 50, 20, 5, 2, 1>.

**Details**

Usage: straw\_dump <oe/observed> <NONE/VC/VC\_SQRT/KR> <hicFile(s)> <chr1>[:x1:x2] <chr2>[:y1:y2] <BP/FRAG> <binsize> <outfile>

**Value**

Data.frame of a sparse matrix of data from hic file. x,y,counts

# Index

[add\\_1D\\_features](#), 3  
[add\\_2D\\_features](#), 4  
[add\\_hic\\_counts](#), 6  
[add\\_hicpro\\_allvalidpairs\\_counts](#), 5  
[add\\_hicpro\\_matrix\\_counts](#), 6

[construct\\_features](#), 7  
[construct\\_features\\_chr](#), 8  
[construct\\_features\\_parallel](#), 10

[expand\\_1D\\_features](#), 11  
[extract\\_hic\\_eigenvectors](#), 12

[generate\\_binned\\_gi\\_list](#), 13  
[generate\\_bintolen\\_gi\\_list](#), 14  
[generate\\_df\\_gi\\_list](#), 15  
[get\\_chr\\_sizes](#), 16  
[get\\_chrs](#), 16  
[get\\_enzyme\\_cutsites](#), 17  
[gi\\_list2HTClist](#), 17  
[gi\\_list\\_binsize\\_detect](#), 18  
[gi\\_list\\_Dthreshold.detect](#), 19  
[gi\\_list\\_read](#), 19  
[gi\\_list\\_topdom](#), 20  
[gi\\_list\\_validate](#), 21  
[gi\\_list\\_write](#), 22

[hic2icenorm\\_gi\\_list](#), 23  
[hicdc2hic](#), 24  
[hicdcdiff](#), 25  
[HiCDCPlus](#), 27  
[HiCDCPlus\\_chr](#), 28  
[HiCDCPlus\\_parallel](#), 30  
[HTClist2gi\\_list](#), 31

[straw](#), 32  
[straw\\_dump](#), 33