

# Package ‘chicane’

July 28, 2020

**Title** Capture Hi-C Analysis Engine

**Type** Package

**Version** 0.1.2

**Date** 2020-07-27

**Author** Erle Holgersen [aut, cre],  
Olivia Leavy [aut],  
Olivia Fletcher [aut],  
Frank Dudbridge [aut],  
Syed Haider [aut]

**Maintainer** Erle Holgersen <erle.holgersen@gmail.com>

**Description** Toolkit for processing and calling interactions in capture Hi-C data. Converts BAM files into counts of reads linking restriction fragments, and identifies pairs of fragments that interact more than expected by chance. Significant interactions are identified by comparing the observed read count to the expected background rate from a count regression model.

**Depends** R (>= 3.5.0), gamlss.tr, gamlss, data.table

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**biocViews**

**Imports** MASS, stats, foreach, doParallel, iterators, bedr, knitr,  
rmarkdown

**SystemRequirements** bedtools

**Suggests** testthat, GenomicInteractions, GenomicRanges, Gviz, countreg

**Additional\_repositories** <http://R-Forge.R-project.org/>

**VignetteBuilder** knitr

**RoxygenNote** 7.1.0

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2020-07-27 23:00:14 UTC

**R topics documented:**

add.covariates . . . . .	3
add.fragment.coordinates . . . . .	4
bedtools.installed . . . . .	4
bre80 . . . . .	5
check.model.numerical.fit . . . . .	6
check.split.data.numerical.fit . . . . .	6
chicane . . . . .	7
combine.replicates . . . . .	9
compare.replicates . . . . .	11
convert.bam . . . . .	11
convert.hicup.digest.bed . . . . .	12
convert.standard.format . . . . .	13
convert.to.one.based . . . . .	13
create.locus.plot . . . . .	14
create.modelfit.plot . . . . .	15
distance.bin . . . . .	16
distance.split . . . . .	17
fill.in.zeros . . . . .	18
filter.fragments . . . . .	18
fit.glm . . . . .	19
fit.model . . . . .	20
get.combination.count . . . . .	22
get.components . . . . .	22
get.distance . . . . .	23
get.id.components . . . . .	24
get.interaction.id . . . . .	24
get.trans.counts . . . . .	25
is.glm.nb.maxiter.warning . . . . .	25
is.glm.nb.theta.error . . . . .	26
is.glm.nb.theta.warning . . . . .	26
model.rows.sanity.check . . . . .	27
model.try.catch . . . . .	27
multiple.testing.correct . . . . .	28
prepare.data . . . . .	29
read.bed . . . . .	31
run.model.fitting . . . . .	31
smart.split . . . . .	32
stratified.enrichment.sample . . . . .	33
test.enrichment . . . . .	33
verify.interaction.data . . . . .	34

---

add.covariates	<i>add.covariates</i>
----------------	-----------------------

---

## Description

Add model covariates (trans counts and distance) to an interactions data table.

## Usage

```
add.covariates(interaction.data)
```

## Arguments

`interaction.data`  
data.table with interaction data. Must contain columns `bait.id`, `target.id`, `bait.chr`, `bait.start`, `bait.end`, `target.chr`, `target.start`, `target.end` and `count`.

## Value

Updated data table with new columns

`bait.trans.count`  
number of trans interactions of bait fragment

`target.trans.count`  
number of trans interactions of target fragment

`distance`      distance between bait and target fragment, or NA if trans

## Author(s)

Erle Holgersen <Erle.Holgersen@icr.ac.uk>

## Examples

```
data(bre80);  
input.cols <- c('bait.id', 'target.id', 'bait.chr', 'bait.start',  
'bait.end', 'target.chr', 'target.start', 'target.end', 'count');  
output <- add.covariates(bre80[, input.cols, with = FALSE]);
```

```
add.fragment.coordinates  
    add.fragment.coordinates
```

---

**Description**

Expand target and bait IDs of the form chrN:start-end to separate coordinate columns in the data table

**Usage**

```
add.fragment.coordinates(id.data)
```

**Arguments**

id.data            data table containing columns target.id and/or bait.id to be expanded

**Value**

Data table with added coordinate columns for target and bait (as applicable).

**Author(s)**

Erle Holgersen <Erle.Holgersen@icr.ac.uk>

**Examples**

```
data(bre80);  
add.fragment.coordinates(bre80[, .(bait.id, target.id)]);
```

---

```
bedtools.installed    bedtools.installed
```

---

**Description**

Check if bedtools exists in PATH

**Usage**

```
bedtools.installed()
```

**Value**

Logical indicating if bedtools was found in PATH

**Author(s)**

Erle Holgersen <Erle.Holgersen@icr.ac.uk>

**Examples**

```
bedtools.installed();
```

---

bre80

*Bre80 Cell Line*

---

**Description**

A dataset containing processed data from a capture Hi-C experiment in the Bre80 normal epithelial breast tissue cell line. The experiment targeted several breast cancer risk loci, and reads that mapped to the 2q35 SNPs rs13387042 and rs16857609 are included in the dataset.

Data was prepared using the `prepare.data` function. Coordinates are GRCh38.

**Usage**

```
data(bre80)
```

**Format**

A data table object with 47,766 rows and 13 columns.

The variables are as follows:

- `target.id` String in chrN:start-end format identifying target fragment
- `bait.id` String in chrN:start-end format identifying bait fragment
- `target.chr` Chromosome of target fragment
- `target.start` Start coordinate of target fragment (zero-based)
- `target.end` End coordinate of target fragment
- `bait.chr` Chromosome of bait fragment
- `bait.start` Start coordinate of bait fragment (zero-based)
- `bait.end` End coordinate of bait fragment
- `bait.to.bait` Boolean indicating if the interaction is bait-to-bait (i.e. the fragment listed as target is also a bait)
- `bait.trans.count` The number of reads linking the bait to fragments in trans (a measure of "interactability")
- `target.trans.count` The number of reads linking the target to fragments in trans (a measure of "interactability")
- `distance` Distance between the midpoints of the bait and target fragments (basepairs). NA for trans interactions
- `count` The number of reads linking the two fragments

**References**

Baxter, Joseph S., et al. "Capture Hi-C identifies putative target genes at 33 breast cancer risk loci." *Nature Communications* 9.1 (2018): 1028.

---

```
check.model.numerical.fit  
  check.model.identifiability
```

---

**Description**

Check if chicane model can be fit on a given dataset. `glm.nb` does not work when all responses are constant, or there are only two unique values and a covariate is a perfect predictor.

**Usage**

```
check.model.numerical.fit(interaction.data)
```

**Arguments**

```
interaction.data  
  Data table of interaction data on which model is to be fit
```

**Value**

boolean indicating if model can be fit

---

```
check.split.data.numerical.fit  
  check.split.data.numerical.fit
```

---

**Description**

Helper function to check if the chicane model can be fit on each element of a split data list.

**Usage**

```
check.split.data.numerical.fit(split.data)
```

**Arguments**

```
split.data    List of data.table objects with fragment interaction data
```

**Value**

Logical indicating if the model can be fit

---

chicane	<i>chicane</i>
---------	----------------

---

## Description

Run full method for detecting significant interactions in capture Hi-C experiments, starting either from a BAM file or preprocessed data from `prepare.data`

## Usage

```
chicane(
  bam = NULL,
  baits = NULL,
  fragments = NULL,
  interactions = NULL,
  replicate.merging.method = "sum",
  distribution = "negative-binomial",
  include.zeros = "none",
  bait.filters = c(0, 1),
  target.filters = c(0, 1),
  distance.bins = NULL,
  multiple.testing.correction = c("bait-level", "global"),
  adjustment.terms = NULL,
  remove.adjacent = FALSE,
  temp.directory = NULL,
  keep.files = FALSE,
  maxit = 100,
  epsilon = 1e-08,
  cores = 1,
  trace = FALSE,
  verbose = FALSE,
  interim.data.dir = NULL
)
```

## Arguments

<code>bam</code>	Path to a BAM file
<code>baits</code>	Path to a BED file containing the baits
<code>fragments</code>	Path to a BED file containing all restriction fragments in the genome
<code>interactions</code>	Data table or path to a text file detailing fragment interactions, typically from <code>prepare.data</code> . Can be used instead of <code>bam/baits/fragments</code> specification if the text files have already been prepared.
<code>replicate.merging.method</code>	Method that should be used for merging replicates, if applicable
<code>distribution</code>	Name of distribution of the counts. Options are 'negative-binomial', 'poisson', 'truncated-poisson', and 'truncated-negative-binomial'

<code>include.zeros</code>	String specifying what zero counts to include. Options are none (default), cis, and all.
<code>bait.filters</code>	Vector of length two, where the first element corresponds to the lower-end filter and the second to the upper-end filter. When global multiple testing correction is performed, altering the bait filtering settings may affect the number of significant results.
<code>target.filters</code>	Vector of length two, giving lower and higher filter, respectively. Changing this filtering setting may affect multiple testing correction by altering the number of tests performed.
<code>distance.bins</code>	Number of bins to split distance into. Models are fit separately in each bin.
<code>multiple.testing.correction</code>	String specifying how multiple testing correction should be performed, by bait or globally.
<code>adjustment.terms</code>	Character vector of extra terms to adjust for in the model fit.
<code>remove.adjacent</code>	Logical indicating whether to remove all reads mapping to adjacent restriction fragments.
<code>temp.directory</code>	Directory where temporary files should be stored. Defaults to current directory.
<code>keep.files</code>	Logical indicating whether to keep temporary files
<code>maxit</code>	Maximum number of IWLS iterations for fitting the model (passed to <code>glm.control</code> )
<code>epsilon</code>	Positive convergence tolerance for Poisson and negative binomial models. Passed to <code>glm.control</code>
<code>cores</code>	Integer value specifying how many cores to use to fit model for cis-interactions.
<code>trace</code>	Logical indicating if output should be produced for each of model fitting procedure. Passed to <code>glm.control</code> or <code>gamlss.control</code>
<code>verbose</code>	Logical indicating whether to print progress reports.
<code>interim.data.dir</code>	Path to directory to store intermediate QC data and plots. NULL indicate skip intermediate results.

## Value

Data table with columns

<code>target.id</code>	String in chrN:start-end format identifying target fragment
<code>bait.id</code>	String in chrN:start-end format identifying bait fragment
<code>target.chr</code>	Chromosome of target fragment
<code>target.start</code>	Start coordinate of target fragment (zero-based)
<code>target.end</code>	End coordinate of target fragment
<code>bait.chr</code>	Chromosome of bait fragment
<code>bait.start</code>	Start coordinate of bait fragment (zero-based)
<code>bait.end</code>	End coordinate of bait fragment



bait.to.bait	Boolean indicating if the interaction is bait-to-bait (i.e. the fragment listed as target is also a bait)
bait.trans.count	The number of reads linking the bait to fragments in trans (a measure of "interactability")
target.trans.count	The number of reads linking the target to fragments in trans (a measure of "interactability")
distance	Distance between the midpoints of the bait and target fragments (basepairs). NA for trans interactions
count	The number of reads linking the two fragments
expected	The expected number of reads linking the two fragments under the fitted model
p.value	P-value for test of the observed number of reads significantly exceeding the expected count
q.value	FDR-corrected p-value

**Author(s)**

Erle Holgersen <Erle.Holgersen@icr.ac.uk>

**Examples**

```
# start from BAM file
bam <- system.file('extdata', 'Bre80_2q35.bam', package = 'chicane');
baits <- system.file('extdata', '2q35.bed', package = 'chicane');
fragments <- system.file('extdata', 'GRCh38_HindIII_chr2.bed.gz', package = 'chicane');
results <- chicane(
  bam = bam,
  baits = baits,
  fragments = fragments
);

# start from pre-processed data
data(bre80);
results <- chicane(interactions = bre80);
```

---

combine.replicates      *combine.replicates*

---

**Description**

Merge biological replicates.

**Usage**

```
combine.replicates(replicates, method = c("sum", "weighted-sum"))
```

**Arguments**

replicates	list of data table objects from <code>prepare.data</code>
method	string specifying the method for merging replicates. Options are 'sum' and 'weighted-sum'.

**Details**

The parameter `method` determines which method is used for merging replicates. Available options are `weighted-sum` and `sum`.

'weighted-sum' implements the size factor scaling approach used in DEseq, rounded to the closest integer. See Anders and Huber 2010 for details.

'sum' is the naive sum of counts across biological replicates.

**Value**

Data table object containing merged data, where counts are stored in columns

<code>count.i</code>	count of interaction in <code>i</code> th replicate
<code>count</code>	count after merging replicates

**References**

Anders, Simon, and Wolfgang Huber. "Differential expression analysis for sequence count data." *Genome biology* 11.10 (2010): R106.

**Examples**

```
# preprocess data
bam <- system.file('extdata', 'Bre80_2q35.bam', package = 'chicane');
baits <- system.file('extdata', '2q35.bed', package = 'chicane');
fragments <- system.file('extdata', 'GRCh38_HindIII_chr2.bed.gz', package = 'chicane');
input.data <- prepare.data(
  bam = bam,
  baits = baits,
  fragments = fragments
);

# combined two datasets into one
merged <- combine.replicates(list(input.data, input.data));
```

---

compare.replicates      *compare.replicates*

---

**Description**

Compare replicates in a pairwise manner and further stratified by distance

**Usage**

```
compare.replicates(interaction.data = NULL, output.directory = ".")
```

**Arguments**

`interaction.data`  
A named vector specifying paths to files created using {prepare.data()}  
`output.directory`  
Path to the output directory where pairwise plots are generated

**Value**

TRUE if pairwise plots were successfully created

**Author(s)**

Syed Haider

**Examples**

```
# TODO
```

---

convert.bam      *convert.bam*

---

**Description**

Convert a BAM file to a format that can be used for replicate merging.

**Note:** This function does not process data enough to be used for interaction calling. Use `prepare.data` for full preprocessing.

**Usage**

```
convert.bam(bam, baits, fragments, temp.directory = NULL, keep.files = FALSE)
```

**Arguments**

<code>bam</code>	Path to a BAM file
<code>baits</code>	Path to a BED file containing the baits
<code>fragments</code>	Path to a BED file containing all restriction fragments in the genome
<code>temp.directory</code>	Directory where temporary files should be stored. Defaults to current directory.
<code>keep.files</code>	Logical indicating whether to keep temporary files

**Author(s)**

Erle Holgersen <Erle.Holgersen@icr.ac.uk>

**See Also**

[prepare.data](#)

---

`convert.hicup.digest.bed`  
*convert.hicup.digest.bed*

---

**Description**

Convert a HiCUP digest file to BED format.

**Usage**

```
convert.hicup.digest.bed(hicup.digest, file.name = "")
```

**Arguments**

<code>hicup.digest</code>	Path to HiCUP digest
<code>file.name</code>	Path to output file. A blank string indicates output to the console.

**Examples**

```
hicup.digest <- system.file('extdata', 'HiCUP_digest_example.txt', package = 'chicane');  
convert.hicup.digest.bed(hicup.digest);
```

---

```
convert.standard.format  
    convert.standard.format
```

---

**Description**

Create a file in standard format for cross compatability including with WashU Epigenome Browser.

**Usage**

```
convert.standard.format(chicane.results, file.name = "")
```

**Arguments**

```
chicane.results  
    Path to CHiCANE interaction calls file  
file.name      Path to output file
```

**Value**

TRUE if output files are created successfully

**Author(s)**

Andrea Gillespie, Syed Haider

**Examples**

```
chicane.results <- system.file(  
  'extdata', 'T47D_2q35_filtered_chicane_calls.txt',  
  package = 'chicane'  
);  
output.file = file.path(tempdir(), 'temp_standard_format.txt');  
convert.standard.format(chicane.results, file.name = output.file);
```

---

```
convert.to.one.based    convert.to.one.based
```

---

**Description**

Convert zero-based region in format chr:start-end to 1-based

**Usage**

```
convert.to.one.based(id)
```

**Arguments**

id                    string in format chr:start-end

**Value**

one-converted ID

---

create.locus.plot      *create.locus.plot*

---

**Description**

Create a file compatible with WashU Epigenome Browser from CHiCANE interaction calls.

**Usage**

```
create.locus.plot(
  genome = "hg38",
  chr = NULL,
  start = NULL,
  end = NULL,
  gene.data = NULL,
  genomic.features = NULL,
  feature.name = NULL,
  fdr.filter = 0.05,
  interaction.data = NULL,
  file.name = NULL,
  height = 5.5,
  width = 8.5,
  track.heights = c(0.2, 0.5, 0.8, 0.5, 1.5, 2)
)
```

**Arguments**

genome                Name of genome build (e.g. 'hg38' or 'hg37')

chr                    Chromosome number for desired locus including 'chr' (e.g. 'chr1')

start                  Start coordinate of desired locus

end                    End coordinate of desired locus

gene.data             Path to chosen genome annotation file in .gtf format

genomic.features     Path to BED file with coordinates of desired feature track

feature.name          Title to appear above genomic features

fdr.filter            Q-value filter threshold for interaction calls to be included

interaction.data      Path to unfiltered CHiCANE calls output

file.name	Path to output file
height	Height in inches for desired plot
width	Width in inches of desired plot
track.heights	Vector of length 6 indicating desired height of individual tracks

**Value**

TRUE if plot was successfully created

**Author(s)**

Andrea Gillespie, Syed Haider

**Examples**

```
# In order to conserve memory only significant interactions are included in example
# interaction.data file. However, in order to show raw counts, unfiltered calls should be
# included and only significant interactions (as set by fdr.filter) will be displayed
```

```
gene.data <- system.file('extdata', 'encode_2q35.gtf', package = 'chicane');
genomic.features <- system.file('extdata', '2q35.bed', package = 'chicane');
interaction.data <- system.file(
  'extdata', 'T47D_2q35_filtered_chicane_calls.txt',
  package = 'chicane'
);
file.name <- file.path(tempdir(), "chr2_interactions.pdf");
```

```
create.locus.plot(
  genome = 'hg38',
  chr = 'chr2',
  start = 216600000,
  end = 217200000,
  gene.data = gene.data,
  genomic.features = genomic.features,
  feature.name = 'baits',
  interaction.data = interaction.data,
  file.name = file.name
);
```

---

create.modelfit.plot    *create.modelfit.plot*

---

**Description**

create a plot representing model's fit

**Usage**

```
create.modelfit.plot(model, file.name = NULL, resolution = 600)
```

**Arguments**

model	An object of fitted model
file.name	A string specifying plotting file name
resolution	A numeric specifying plot's resolution

**Value**

TRUE if plot was successfully created

**Author(s)**

Syed Haider

---

distance.bin	<i>distance.bin</i>
--------------	---------------------

---

**Description**

Assign distances to a meaningful category

**Usage**

```
distance.bin(distance)
```

**Arguments**

distance	Vector of distances that should be mapped to a distance bin
----------	---

**Value**

vector of same length as distance containing assigned distance bins



---

distance.split	<i>distance.split</i>
----------------	-----------------------

---

### Description

Split interaction data into subsets that are large enough for the chicane model to be fit (see Details), based on distance. This step allows the distance term in the model to be fit in a piecewise linear fashion.

### Usage

```
distance.split(
  interaction.data,
  distance.bins = NULL,
  min.rows.bin = 50,
  verbose = FALSE
)
```

### Arguments

interaction.data	Data table of interaction data, typically from prepare.data
distance.bins	Number of distance bins desired. If NULL, a number is chosen to ensure that the negative binomial can be fit in all bins.
min.rows.bin	The minimum number of expected rows in a distance bin. Ignored if distance.bins is set
verbose	Logical indicating whether to print progress reports

### Details

Fitting `glm.nb` fails when there is a lack of overdispersion in the data. The chicane method contains logic to catch these errors and instead fit a Poisson model. However, to avoid this happening more than necessary, an attempt is made to avoid distance splits that will clearly result in numerical errors. This includes bins of data where the count is the same for all rows, or a covariate is a perfect predictor of count.

### Value

List where each element corresponds to a specified distance bin, and the final one corresponding to trans-interactions (if present)

### Examples

```
data(bre80);
distance.split(bre80);
```

---

<code>fill.in.zeros</code>	<i>fill.in.zeros</i>
----------------------------	----------------------

---

**Description**

Add zero counts to interaction data

**Usage**

```
fill.in.zeros(interaction.data, baits, fragments)
```

```
fill.in.zeroes(interaction.data, baits, fragments)
```

**Arguments**

<code>interaction.data</code>	Data table containing interaction data
<code>baits</code>	Vector of bait IDs used in the experiment, in format chrN:start-end
<code>fragments</code>	Vector of potential fragments the baits can link up to, in format chrN:start-end

**Value**

Data table containing origiina

**Examples**

```
data(bre80);
bait.file <- system.file('extdata', '2q35.bed', package = 'chicane');
fragment.file <- system.file('extdata', 'GRCh38_HindIII_chr2.bed.gz', package = 'chicane');
results <- fill.in.zeros(
  bre80,
  baits = read.bed(bait.file),
  fragments = read.bed(fragment.file)
);
```

---

<code>filter.fragments</code>	<i>filter.fragments</i>
-------------------------------	-------------------------

---

**Description**

Filter low and high-interacting restriction fragments based on the total number of trans counts

**Usage**

```
filter.fragments(  
  interaction.data,  
  bait.filters = c(0, 1),  
  target.filters = c(0, 1),  
  verbose = FALSE  
)
```

**Arguments**

<code>interaction.data</code>	Data table containing interactions
<code>bait.filters</code>	Vector of length two, where the first element corresponds to the lower-end filter and the second to the upper-end filter. When global multiple testing correction is performed, altering the bait filtering settings may affect the number of significant results.
<code>target.filters</code>	Vector of length two, giving lower and higher filter, respectively. Changing this filtering setting may affect multiple testing correction by altering the number of tests performed.
<code>verbose</code>	Logical indicating whether to print progress reports.

**Value**

Data table containing fragments that passed all filters

**Author(s)**

Erle Holgersen <Erle.Holgersen@icr.ac.uk>

**Examples**

```
# filter out lowest 10% of baits  
filter.fragments(bre80, bait.filters = c(0.1, 1))
```

---

fit.glm

*fit.glm*

---

**Description**

Fit GLM according to a specified distribution. This needs to be done separately from `glm` in order to include negative binomial and truncated distributions as options.

**Usage**

```
fit.glm(
  formula,
  data,
  distribution = c("negative-binomial", "poisson", "truncated-poisson",
    "truncated-negative-binomial"),
  start = NULL,
  init.theta = NULL,
  maxit = 100,
  epsilon = 1e-08,
  trace = FALSE
)
```

**Arguments**

formula	Formula specifying model of interest
data	Data frame containing variables specified in formula
distribution	Name of distribution of the counts. Options are 'negative-binomial', 'poisson', 'truncated-poisson', and 'truncated-negative-binomial'
start	Starting values for model coefficients
init.theta	Initial value of theta if fitting the negative binomial distribution
maxit	Maximum number of IWLS iterations for fitting the model (passed to glm.control)
epsilon	Positive convergence tolerance for Poisson and negative binomial models. Passed to glm.control
trace	Logical indicating if output should be produced for each of model fitting procedure. Passed to glm.control or gamlss.control

**Value**

List with elements

model	model object
expected.values	vector of expected values for each element in original data
p.values	vector of p-values for test of significantly higher response than expected

---

fit.model

*fit.model*

---

**Description**

Fit negative binomial model to obtain p-values for interactions.

**Usage**

```
fit.model(
  interaction.data,
  distance.bins = NULL,
  distribution = "negative-binomial",
  bait.filters = c(0, 1),
  target.filters = c(0, 1),
  adjustment.terms = NULL,
  maxit = 100,
  epsilon = 1e-08,
  cores = 1,
  trace = FALSE,
  verbose = FALSE,
  interim.data.dir = NULL
)
```

**Arguments**

<code>interaction.data</code>	data.table object containing interaction counts. Must contain columns distance, count, and bait_trans_count.
<code>distance.bins</code>	Number of bins to split distance into. Models are fit separately in each bin.
<code>distribution</code>	Name of distribution of the counts. Options are 'negative-binomial', 'poisson', 'truncated-poisson', and 'truncated-negative-binomial'
<code>bait.filters</code>	Vector of length two, where the first element corresponds to the lower-end filter and the second to the upper-end filter. When global multiple testing correction is performed, altering the bait filtering settings may affect the number of significant results.
<code>target.filters</code>	Vector of length two, giving lower and higher filter, respectively. Changing this filtering setting may affect multiple testing correction by altering the number of tests performed.
<code>adjustment.terms</code>	Character vector of extra terms to adjust for in the model fit.
<code>maxit</code>	Maximum number of IWLS iterations for fitting the model (passed to <code>glm.control</code> )
<code>epsilon</code>	Positive convergence tolerance for Poisson and negative binomial models. Passed to <code>glm.control</code>
<code>cores</code>	Integer value specifying how many cores to use to fit model for cis-interactions.
<code>trace</code>	Logical indicating if output should be produced for each of model fitting procedure. Passed to <code>glm.control</code> or <code>gamlss.control</code>
<code>verbose</code>	Logical indicating whether to print progress reports.
<code>interim.data.dir</code>	Path to directory to store intermediate QC data and plots.

**Details**

Fit a negative binomial model for obtaining p-value for interactions. The data is first sorted by distance, and models are fit separately in each quantile of the distance-sorted data.

**Value**

Interactions data with expected number of interactions and p-values added.

**Examples**

```
data(bre80);
fit.model(bre80);
```

---

```
get.combination.count  get.combination.count
```

---

**Description**

Calculate the number of possible combinations between baits and fragments, excluding self-ligations and only counting bait-to-bait interactions once (e.g. a-b, not b-a)

**Usage**

```
get.combination.count(baits, fragments, cis.only = FALSE)
```

**Arguments**

baits	vector of bait IDs in form chrN:start-end
fragments	vector of fragment IDs in form chrN:start-end
cis.only	logical indicating whether cis-interactions only should be considered

**Value**

total number of possible combinations

---

```
get.components  get.components
```

---

**Description**

Split a fragment in format chr:start-end to a list of corresponding elements

**Usage**

```
get.components(id)
```

**Arguments**

id                    string in format chr:start-end

**Value**

list with entries 'chr', 'start', 'end'

---

<i>get.distance</i>	<i>get.distance</i>
---------------------	---------------------

---

**Description**

Calculate distance between bait and target region

**Usage**

```
get.distance(interaction.data)
```

**Arguments**

interaction.data  
data.table with interaction data. Must contain columns bait.chr, bait.start, bait.end, target.chr, target.start, target.end

**Value**

vector of absolute distances (NA for trans-interactions)

**Examples**

```
data(bre80);  
input.cols <- c('bait.chr', 'bait.start', 'bait.end',  
              'target.chr', 'target.start', 'target.end');  
get.distance( bre80[, input.cols, with = FALSE]);
```

---

`get.id.components`      *get.id.components*

---

**Description**

Split a segment ID in form chrN:start-end into its different components

**Usage**

```
get.id.components(id)
```

**Arguments**

`id`                      segment ID of form chrN:start-end

**Value**

A character vector of length three, where the elements are chromosome, start, and end, respectively. If `id` is a vector, a list of the same length is returned

**Examples**

```
get.id.components('chrX:6-30');  
get.id.components(c('3:4-10', '22:1000-20000'))
```

---

`get.interaction.id`      *get.interaction.id*

---

**Description**

Generate a unique identifying ID for each interaction

**Usage**

```
get.interaction.id(bait, other.end, bait.to.bait, zero.based = FALSE)
```

**Arguments**

`bait`                      id of bait in format chr:start-end  
`other.end`                id of other end in format chr:start-end  
`bait.to.bait`            logical indicating whether both ends are baits  
`zero.based`              logical indicating if IDs are zero-based

**Value**

string identifying interaction



---

get.trans.counts	<i>get.trans.counts</i>
------------------	-------------------------

---

**Description**

Calculate the number of trans-interactions per fragment, accounting for the fact that baits can be listed either as bait or target.

**Usage**

```
get.trans.counts(interaction.data)
```

**Arguments**

interaction.data  
Data table containing interactions

**Value**

Data table with columns fragment.id and trans.count.

fragment.id	ID of restriction fragment in chrN:start-end format
trans.count	Number of trans interactions involving the fragment

**Examples**

```
data(bre80);  
get.trans.counts(bre80[, .(bait.chr, target.chr, bait.id, target.id, count)]);
```

---

is.glm.nb.maxiter.warning	<i>is.glm.nb.maxiter.warning</i>
---------------------------	----------------------------------

---

**Description**

Check if a warning object is an iteration limit reached warning from glm.nb

**Usage**

```
is.glm.nb.maxiter.warning(w)
```

**Arguments**

w  
Warning object

**Value**

Logical indicating if warning matches iteration limit reached warning

---

`is.glm.nb.theta.error` *check.glm.nb.theta.error*

---

**Description**

Check if an error matches the error raised by `glm.nb` due to an inflated theta estimate. This happens when the variance of the negative binomial does not exceed the mean (i.e. there is no overdispersion). In such cases, the Poisson distribution may be a suitable alternative.

**Usage**

```
is.glm.nb.theta.error(e)
```

**Arguments**

`e` Error object

**Value**

Boolean indicating if error matches

---

`is.glm.nb.theta.warning`  
*is.glm.nb.theta.warning*

---

**Description**

Check if a warning matches the square root warning raised by `glm.nb` due to an inflated theta estimate. This happens when the variance of the negative binomial does not exceed the mean (i.e. there is no overdispersion). In such cases, the Poisson distribution may be a suitable alternative.

**Usage**

```
is.glm.nb.theta.warning(w)
```

**Arguments**

`w` Warning object

**Value**

Boolean indicating if warning matches

---

model.rows.sanity.check  
*model.rows.sanity.check*

---

### Description

Check that the model fit contains the same number of rows as the data used to fit it, and throw an error if not

### Usage

```
model.rows.sanity.check(model.data, model)
```

### Arguments

model.data	Data used to fit model
model	Resulting negative binomial model object

### Value

None

---

model.try.catch      *model.try.catch*

---

### Description

Internal function for fitting model within a tryCatch loop, handling numerical errors gracefully.

### Usage

```
model.try.catch(  
  model.formula,  
  data,  
  distribution = "negative-binomial",  
  maxit = 100,  
  epsilon = 1e-08,  
  init.theta = NULL,  
  start = NULL,  
  trace = FALSE,  
  verbose = FALSE  
)
```

**Arguments**

<code>model.formula</code>	formula
<code>data</code>	model data
<code>distribution</code>	Name of distribution of the counts. Options are 'negative-binomial', 'poisson', 'truncated-poisson', and 'truncated-negative-binomial'
<code>maxit</code>	Maximum number of IWLS iterations for fitting the model (passed to <code>glm.control</code> )
<code>epsilon</code>	Positive convergence tolerance for Poisson and negative binomial models. Passed to <code>glm.control</code>
<code>init.theta</code>	Initial value of theta in negative binomial model
<code>start</code>	starting values of coefficients in linear predictor
<code>trace</code>	Logical indicating if output should be produced for each of model fitting procedure. Passed to <code>glm.control</code> or <code>gamlss.control</code>
<code>verbose</code>	Logical indicating whether to print progress reports.

**Value**

	List with elements
<code>model</code>	model object. Set to NULL if no model could be fit.
<code>expected.values</code>	vector of expected values for each element in original data, or vector of NAs if no model could be fit
<code>p.values</code>	vector of p-values for test of significantly higher response than expected, or vector of NAs if no model could be fit

---

`multiple.testing.correct`

*multiple.testing.correct*

---

**Description**

Perform multiple testing correction on p-values from interaction test. By default, multiple testing correction is applied per bait. To change this to a global multiple testing correction, set `bait.level = FALSE`.

**Usage**

```
multiple.testing.correct(interaction.data, bait.level = TRUE)
```

**Arguments**

<code>interaction.data</code>	Data table of interaction calls. Must contain columns <code>p.value</code> and <code>bait.id</code> .
<code>bait.level</code>	Logical indicating whether multiple testing correction should be performed per bait.

**Value**

Original data table with new column

q.value            FDR-corrected p-value

**Examples**

```
data(bre80);
results <- fit.model(bre80);
adjusted.results <- multiple.testing.correct(results);
```

---

prepare.data	<i>prepare.data</i>
--------------	---------------------

---

**Description**

Prepare data for running interaction calling. Takes a BAM file and baits and restriction fragments as input, and returns a data table with data ready for analysis.

**Usage**

```
prepare.data(
  bam,
  baits,
  fragments,
  replicate.merging.method = "sum",
  include.zeros = c("none", "cis", "all"),
  remove.adjacent = FALSE,
  temp.directory = NULL,
  keep.files = FALSE,
  verbose = FALSE
)
```

**Arguments**

bam	Path to a BAM file
baits	Path to a BED file containing the baits
fragments	Path to a BED file containing all restriction fragments in the genome
replicate.merging.method	Method that should be used for merging replicates, if applicable
include.zeros	String specifying what zero counts to include. Options are none (default), cis, and all.
remove.adjacent	Logical indicating whether to remove all reads mapping to adjacent restriction fragments.

temp.directory Directory where temporary files should be stored. Defaults to current directory.  
 keep.files Logical indicating whether to keep temporary files  
 verbose Logical indicating whether to print progress reports.

**Value**

Data table object with columns

target.id	String in chrN:start-end format identifying target fragment
bait.id	String in chrN:start-end format identifying bait fragment
target.chr	Chromosome of target fragment
target.start	Start coordinate of target fragment (zero-based)
target.end	End coordinate of target fragment
bait.chr	Chromosome of bait fragment
bait.start	Start coordinate of bait fragment (zero-based)
bait.end	End coordinate of bait fragment
bait.to.bait	Boolean indicating if the interaction is bait-to-bait (i.e. the fragment listed as target is also a bait)
count	The number of reads linking the two fragments
bait.trans.count	The number of reads linking the bait to fragments in trans (a measure of "interactability")
target.trans.count	The number of reads linking the target to fragments in trans (a measure of "interactability")
distance	Distance between the midpoints of the bait and target fragments (basepairs). NA for trans interactions

**Examples**

```
bam <- system.file('extdata', 'Bre80_2q35.bam', package = 'chicane');
baits <- system.file('extdata', '2q35.bed', package = 'chicane');
fragments <- system.file('extdata', 'GRCh38_HindIII_chr2.bed.gz', package = 'chicane');
input.data <- prepare.data(
  bam = bam,
  baits = baits,
  fragments = fragments
);
```

---

read.bed	<i>read.bed</i>
----------	-----------------

---

**Description**

Read a BED file and return regions in chrN:start-end format

**Usage**

```
read.bed(bed.path, zero.based = TRUE)
```

**Arguments**

bed.path	Path to bed file
zero.based	Whether to return ID in zero-based coordinates

**Value**

vector of region IDs

**Examples**

```
bait.file <- system.file('extdata', '2q35.bed', package = 'chicane');  
baits <- read.bed(bait.file);
```

---

run.model.fitting	<i>run.model.fitting</i>
-------------------	--------------------------

---

**Description**

Run model fitting procedure for either bait-to-bait or other interactions. Meant for internal use only.

**Usage**

```
run.model.fitting(  
  interaction.data,  
  distance.bins = NULL,  
  distribution = "negative-binomial",  
  bait.to.bait = FALSE,  
  adjustment.terms = NULL,  
  maxit = 100,  
  epsilon = 1e-08,  
  cores = 1,  
  trace = FALSE,  
  verbose = FALSE,  
  interim.data.dir = NULL  
)
```

**Arguments**

interaction.data	data.table object containing interaction counts. Must contain columns distance, count, and bait_trans_count.
distance.bins	Number of bins to split distance into. Models are fit separately in each bin.
distribution	Name of distribution of the counts. Options are 'negative-binomial', 'poisson', 'truncated-poisson', and 'truncated-negative-binomial'
bait.to.bait.adjustment.terms	Logical indicating if model should be fit as bait-to-bait Character vector of extra terms to adjust for in the model fit
maxit	Maximum number of IWLS iterations for fitting the model (passed to glm.control)
epsilon	Positive convergence tolerance for Poisson and negative binomial models. Passed to glm.control
cores	Integer value specifying how many cores to use to fit model for cis-interactions.
trace	Logical indicating if output should be produced for each of model fitting procedure. Passed to glm.control or gamlss.control
verbose	Logical indicating whether to print progress reports.
interim.data.dir	Path to directory to store intermediate QC data and plots.

**Value**

Interactions data with expected number of interactions and p-values added.

---

smart.split

*smart.split*

---

**Description**

Split a data frame into a prespecified number of bins, using split and cut. Unlike the default R functions, this does not fail when asked to split the data into a single bin.

**Usage**

```
smart.split(dat, bins)
```

**Arguments**

dat	Data frame or data table to be split
bins	Number of bins to split data into

**Value**

List with bins elements. Each element corresponds to one portion of the data



---

```
stratified.enrichment.sample
      stratified.enrichment.sample
```

---

**Description**

Generate a stratified sample matching distance distribution of significant interactions.

**Usage**

```
stratified.enrichment.sample(nonsignificant.results, significant.results)
```

**Arguments**

```
nonsignificant.results
      Data table containing non-significant interactions that should be sampled from
significant.results
      Data table of significant results. Used to determine size of strata in stratified
      sampling procedure.
```

---

```
test.enrichment      test.enrichment
```

---

**Description**

test.enrichment

**Usage**

```
test.enrichment(
  interaction.data,
  feature.bed,
  significance.cutoff = 0.05,
  span = 0,
  n = 1000,
  remove.bait.to.bait = TRUE
)
```

**Arguments**

```
interaction.data
      Data table containing details on interactions
feature.bed      BED file with regions of features
significance.cutoff
      q-value threshold for significant interactions
```

span            Distance around target restriction fragment to consider. If set to zero (default), only features that overlap with the restriction fragment itself are considered.

n                Number of random samples to consider

remove.bait.to.bait    Logical specifying whether to exclude bait-to-bait interactions

**Value**

list with elements

observed        observed overlap between significant interactions and features

random         vector of length n giving overlap between random samples and features

**Author(s)**

Erle Holgersen <Erle.Holgersen@icr.ac.uk>

**Examples**

```
data(bre80);
ctcf.bed <- system.file('extdata', 'T47D_chr2_CTCF.bed.gz', package = 'chicane');

results <- chicane(interactions = bre80);
test.enrichment(results, ctcf.bed, significance.cutoff = 0.25);
```

---

verify.interaction.data  
*verify.interaction.data*

---

**Description**

Verify that interaction.data object is in expected format. Throws an error if object does not fit requirements.

**Usage**

```
verify.interaction.data(interaction.data)
```

**Arguments**

interaction.data  
Object to be verified.

**Value**

None

# Index

## \* datasets

- bre80, 5
- add.covariates, 3
- add.fragment.coordinates, 4
- bedtools.installed, 4
- bre80, 5
- check.model.numerical.fit, 6
- check.split.data.numerical.fit, 6
- chicane, 7
- combine.replicates, 9
- compare.replicates, 11
- convert.bam, 11
- convert.hicup.digest.bed, 12
- convert.standard.format, 13
- convert.to.one.based, 13
- create.locus.plot, 14
- create.modelfit.plot, 15
- distance.bin, 16
- distance.split, 17
- fill.in.zeroes (fill.in.zeros), 18
- fill.in.zeros, 18
- filter.fragments, 18
- fit.glm, 19
- fit.model, 20
- get.combination.count, 22
- get.components, 22
- get.distance, 23
- get.id.components, 24
- get.interaction.id, 24
- get.trans.counts, 25
- is.glm.nb.maxiter.warning, 25
- is.glm.nb.theta.error, 26
- is.glm.nb.theta.warning, 26
- model.rows.sanity.check, 27
- model.try.catch, 27
- multiple.testing.correct, 28
- prepare.data, 12, 29
- read.bed, 31
- run.model.fitting, 31
- smart.split, 32
- stratified.enrichment.sample, 33
- test.enrichment, 33
- verify.interaction.data, 34