CHiCAGO Vignette

Cairns*, Freire-Pritchett*... Spivakov et al.
Additional file 3.

Introduction

CHiCAGO is a method for detecting statistically significant interaction events in Capture HiC data. This vignette will walk you through a typical CHiCAGO analysis.

NOTE: A wrapper to perform this type of analysis, called *runChicago.R*, is provided as part of *chicagoTools*, which is available from our Bitbucket repository. Refer to the *chicagoTools* README for more information.

The statistical foundations of CHiCAGO will be presented in a separate paper, currently in preparation. Briefly, CHiCAGO uses a convolution noise model accounting for both 'Brownian' (distance-dependent) and 'technical' noise. It borrows information across interactions (with appropriate normalisation) to estimate these noise components separately on different subsets of data. CHiCAGO then uses a p-value weighting procedure based on the expected true positive rates at different distance ranges (estimated from data), with scores representing soft-thresholded -log weighted p-values. The score threshold of 5 is a suggested stringent score threshold for calling significant interactions.

WARNING: The data set used in this tutorial comes from the package *PCHiCdata*. This package contains small parts of the real data sets (two chromosomes each) from published Promoter Capture HiC analyses in mouse ESCs (Schoenfelder et al. 2015) and GM12878 cells, derived from human LCLs (Mifsud et al. 2015) (note that both papers used a different interaction calling algorithm and we are only reusing raw data from them). Do not use any of these sample input data for purposes other than training.

In this vignette, we use the GM12878 data (Mifsud et al. 2015):

```
library(PCHiCdata)
```

Input files required

Before you start, you will need:

1. Five restriction map information files:

- Restriction map file (.rmap) - a bed file containing coordinates of the restriction fragments. By default, 4 columns: chr, start, end, fragmentID.
- Bait map file (.baitmap) - a bed file containing coordinates of the baited restriction fragments, and their associated annotations. By default, 5 columns: chr, start, end, fragmentID, geneName.
- *nperbin* file (.npb), *nbaitsperbin* file (.nbpb), *proxOE* file (.poe) - Precompute these tables from the .rmap and .baitmap files, using the Python scripts from *chicagoTools* at our Bitbucket repository: `countNperBin.py`, `countNbaitsPerBin.py` and `getProxOE.py`. Refer to the `chicagoTools` README file for more details.

We recommend that you put all five of these files into the same directory. An examples of a valid design folder, for a two-chromosome sample of the GM12878 data used in this vignette, is provided in the PCHiCdata package, as follows.

```
dataPath <- system.file("extdata", package="PCHiCdata")
testDesignDir <- file.path(dataPath, "hg19TestDesign")
dir(testDesignDir)
```

```
## [1] "h19_chr20and21.baitmap" "h19_chr20and21.nbpb"
## [3] "h19_chr20and21.npb"     "h19_chr20and21.poe"
## [5] "h19_chr20and21.rmap"
```

2. You will also need input data files. These should be in CHiCAGO input format, *.chinput*. *.chinput* files can be obtained from aligned Capture HiC BAM files by running `bam2chicago.sh`, available as part of `chicagoTools`. (To obtain BAM files from raw fastq files, run them through a HiC alignment & QC pipeline such as HiCUP

Example *.chinput* files are provided in the PCHiCdata package, as follows:

```
testDataPath <- file.path(dataPath, "GMchinputFiles")
dir(testDataPath)
```

```
## [1] "GM_rep1.chinput" "GM_rep2.chinput" "GM_rep3.chinput"
```

```
files <- c(
    file.path(testDataPath, "GM_rep1.chinput"),
    file.path(testDataPath, "GM_rep2.chinput"),
    file.path(testDataPath, "GM_rep3.chinput")
  )
```

OPTIONAL: Because this data set is smaller than usual, we need to input some custom settings:

```
settingsFile <- file.path(system.file("extdata", package="PCHiCda
                          "sGM12878Settings", "sGM12878.settingsF
```

Normally, you will not have to do this.

Example workflow

We run CHiCAGO on the test data as follows. First, we create a blank `chicagoData` object, and we tell it where the design files are. For this example, we also provide the optional settings file.

```
library(Chicago)

cd <- setExperiment(designDir = testDesignDir, settingsFile = set
```

The properties of `chicagoData` objects are discussed more in The chicagoData object.

Next, we read in the input data files:

```
cd <- readAndMerge(files=files, cd=cd)
```

Finally, we run the pipeline with `chicagoPipeline()`:

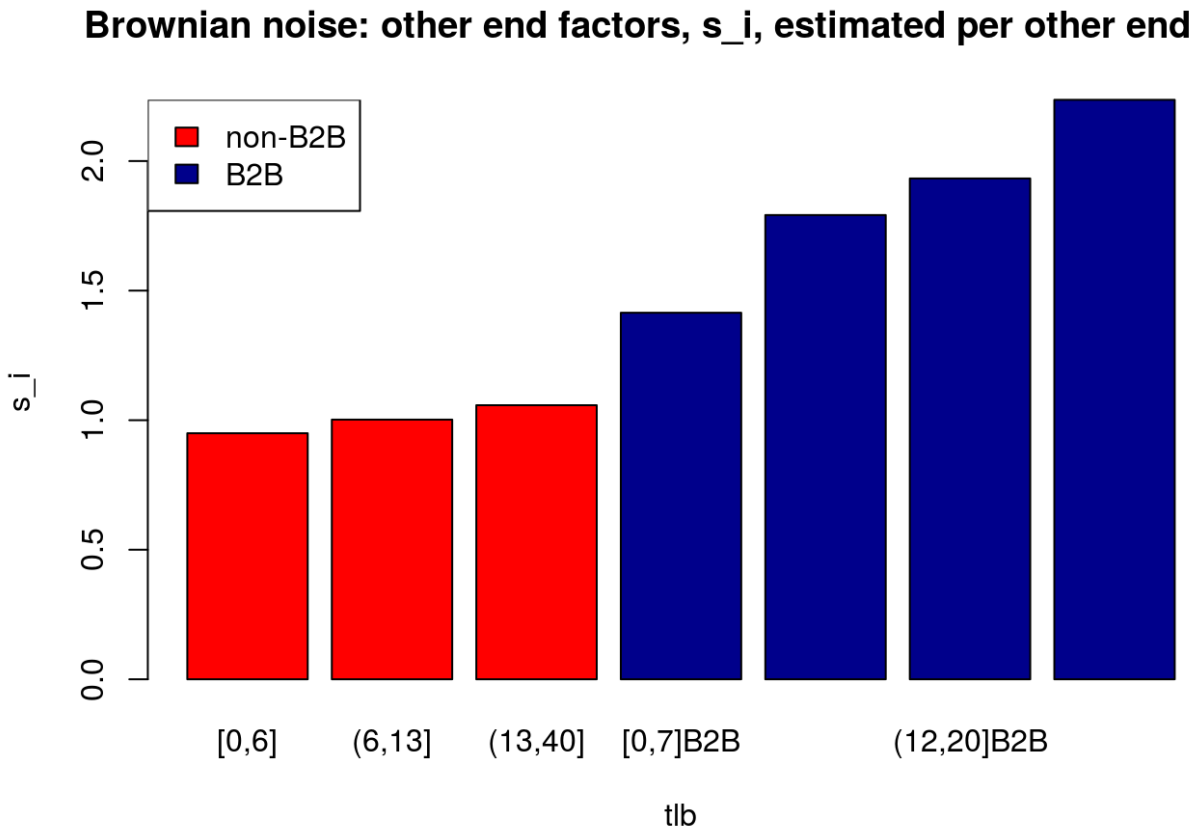
```
cd <- chicagoPipeline(cd)
```

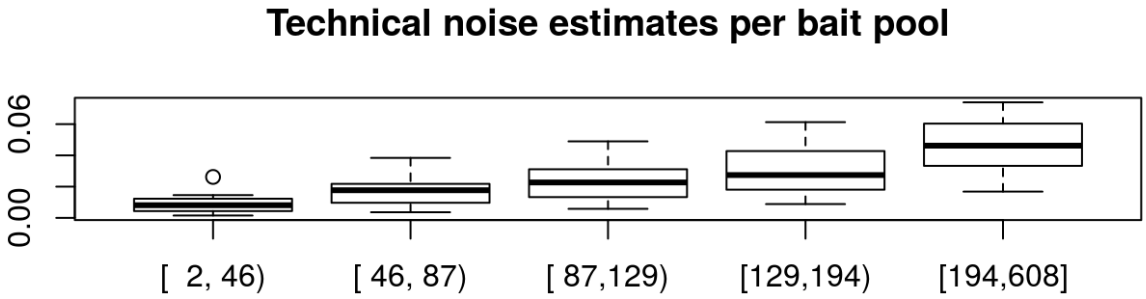
Output plots

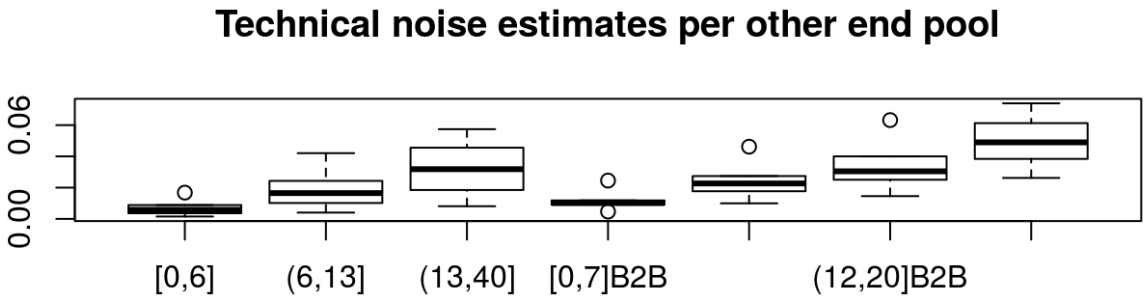
`chicagoPipeline()` produces a number of plots. You can save these to disk by setting the `outprefix` argument in `chicagoPipeline()`.

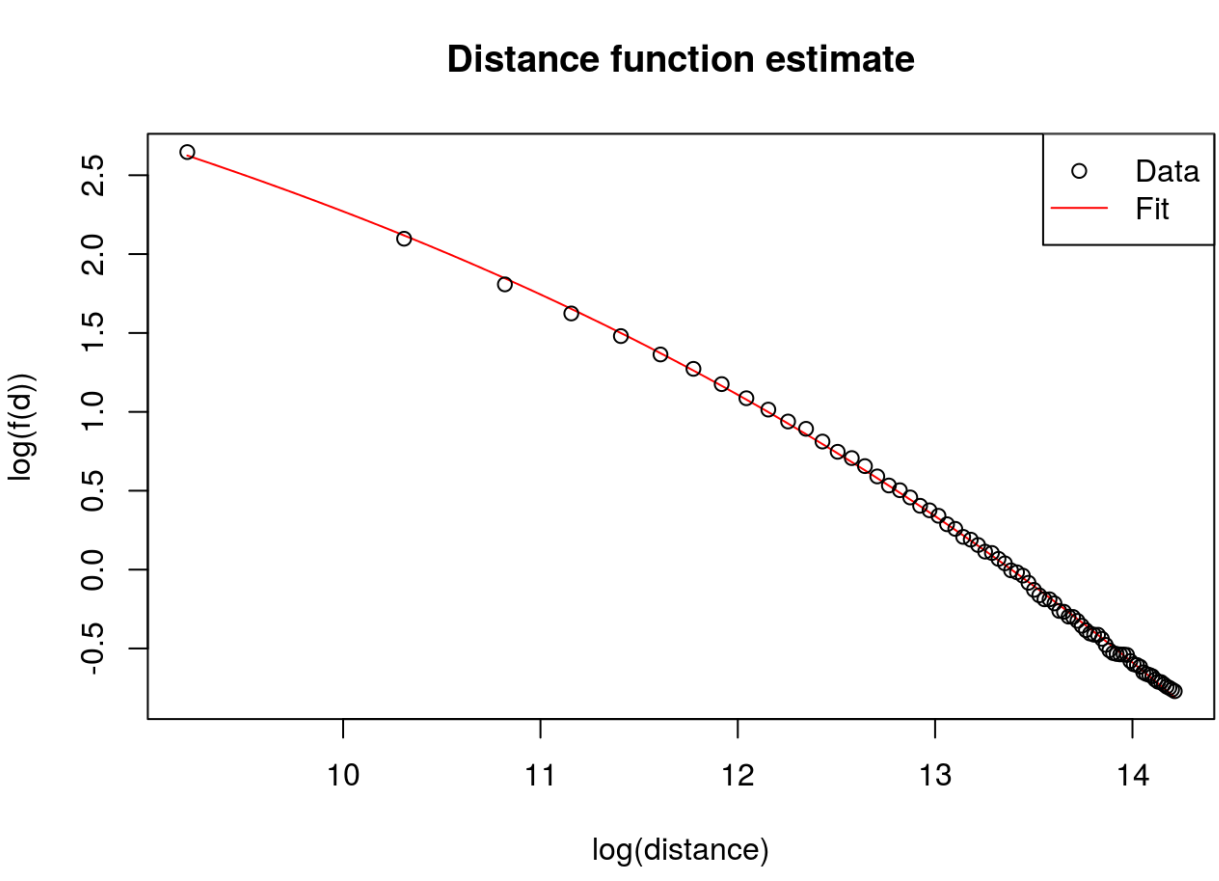
The plots are as follows:

1. Brownian noise other end factors: The adjustment made to the mean Brownian noise count, based on the class of the other end. ("tlb" refers to the number of trans reads the other end has, in total. "B2B" stands for a "bait-to-bait" interaction.)
2. Technical noise estimates: The mean number of technical noise reads expected for other ends and baits, respectively, in certain classes. These classes, displayed on the x axis, again refer to the number of trans reads that the other end has.
3. Distance function: The mean number of Brownian noise reads expected for an `average` bait, as a function of distance, plotted on a log-log scale.




Technical noise estimates per bait pool
0.00
0.06
[2, 46)
[46, 87)
[87,129)
[129,194)
[194,608]


Technical noise estimates per other end pool
0.00
0.06
[0,6]
(6,13]
(13,40]
[0,7]B2B
(12,20]B2B


Distance function estimate
Data
Fit
log(f(d))
-0.5
0.0
0.5
1.0
1.5
2.0
2.5
10
11
12
13
14
log(distance)

Output files

You can export the results to disk, using `exportResults()`. (If you use *runChicago.R*, the files appear in ./<results-folder>/data). By default, the function outputs three different output file formats:

```
outputDirectory <- tempdir()
exportResults(cd, file.path(outputDirectory, "vignetteOutput"))
```

```
## Reading the restriction map file...
## Reading the bait map file...
## Preparing the output table...
## Writing out for seqMonk...
## Writing out interBed...
## Preprocessing for WashU outputs...
## Writing out text file for WashU browser upload...
```

Each called interaction is assigned a score that represents how strong CHiCAGO believes the interaction is (formally, it is based on -log(adjusted P-value)). Thus, a larger score represents a stronger interaction. In each case, the score threshold of 5 is applied.

Summary of output files:

ibed format (ends with …ibed):

- each row represents an interaction
- first four columns give information about the chromosome, start, end and name of the bait fragment
- next four columns give information about the chromosome, start, end and name of the other end that interacts with the bait fragment
- N_reads is the number of reads
- score is as defined above

seqmonk format (ends with …seqmonk.txt)

- Can be read by seqmonk.
- An interaction is represented by two rows: the first row is the bait, the second the other end. Thus, the file alternates: bait1, otherEnd1, bait2, otherEnd2, …
- Columns are: chromosome, start, end, name, number of reads, interaction score (see above)

washU_text format (ends with …washU_text.txt)

- Can be read by the WashU browser

- Upload via the “Got text files instead? Upload them from your computer” link.
- Note - Advanced users may wish to export to washU_track format instead. See the help page for `exportResults()`.

For bait-to-bait interactions, the interaction can be tested either way round (i.e. either fragment can be considered the “bait”). In most output formats, both of these tests are preserved. The exception is washU output, where these scores are consolidated by taking the maximum.

Peak enrichment for features

`peakEnrichment4Features()` tests the hypothesis that other ends in the CHiCAGO output are enriched for genomic features of interest - for example, histone marks associated with enhancers. We find out how many overlaps are expected under the null hypothesis (i.e. that there is no enrichment) by shuffling the other ends around in the genome, while preserving the overall distribution of distances over which interactions span.

You will need additional files to perform this analysis - namely, a .bed file for each feature. We include ChIP-seq data from the ENCODE consortium (The ENCODE Project Consortium 2012), also restricted to chr20 and chr21. (Data accession numbers: Bernstein lab GSM733752, GSM733772, GSM733708, GSM733664, GSM733771, GSM733758)

First, we find the folder that contains the features, and construct a list of the features to use:

```
featuresFolder <- file.path(dataPath, "GMfeatures")
dir(featuresFolder)
```

```
## [1] "featuresGM.txt"
## [2] "spp.wgEncodeBroadHistoneGm12878CtcfStdAln_chr20and21.narr
## [3] "wgEncodeBroadHistoneGm12878H3k27acStdAln_chr20and21.narro
## [4] "wgEncodeBroadHistoneGm12878H3k27me3StdAln_chr20and21.narr
## [5] "wgEncodeBroadHistoneGm12878H3k4me1StdAln_chr20and21.narro
## [6] "wgEncodeBroadHistoneGm12878H3k4me3StdAln_chr20and21.narro
## [7] "wgEncodeBroadHistoneGm12878H3k9me3StdAln_chr20and21.narro
```

```
featuresFile <- file.path(featuresFolder, "featuresGM.txt")
featuresTable <- read.delim(featuresFile, header=FALSE, as.is=TRU
featuresList <- as.list(featuresTable$V2)
names(featuresList) <- featuresTable$V1
featuresList
```

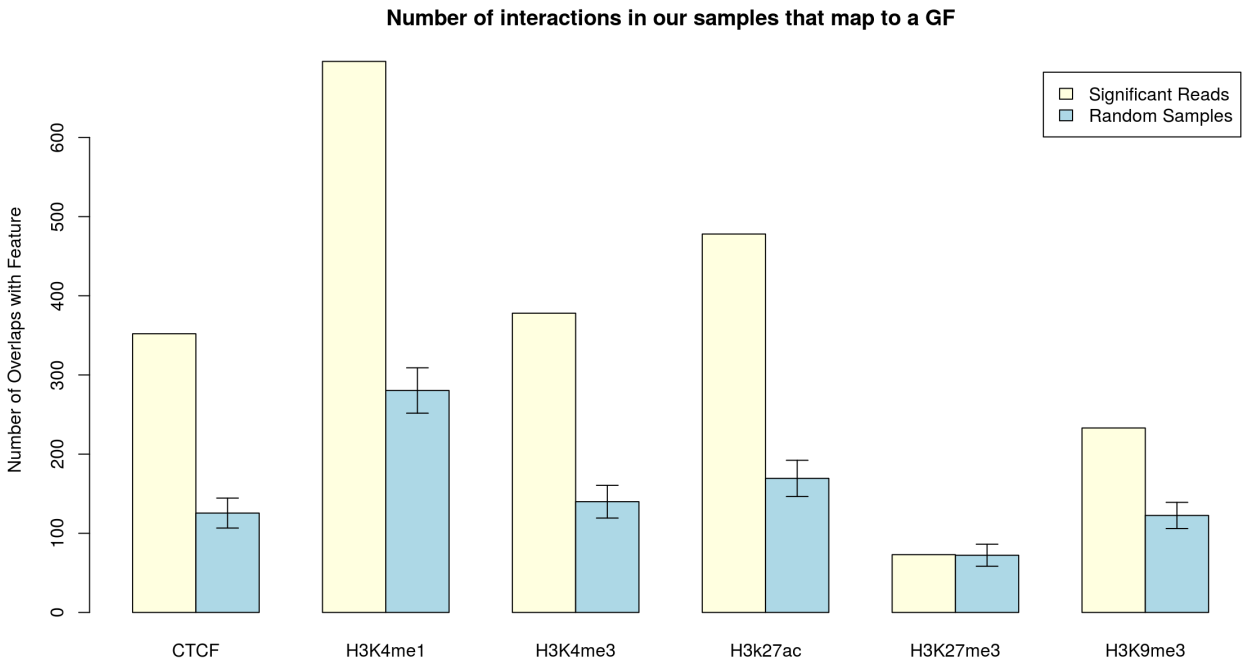
```
## $CTCF
## [1] "spp.wgEncodeBroadHistoneGm12878CtcfStdAln_chr20and21.narr
##
## $H3K4me1
## [1] "wgEncodeBroadHistoneGm12878H3k4me1StdAln_chr20and21.narrc
##
## $H3K4me3
## [1] "wgEncodeBroadHistoneGm12878H3k4me3StdAln_chr20and21.narrc
##
## $H3k27ac
## [1] "wgEncodeBroadHistoneGm12878H3k27acStdAln_chr20and21.narrc
##
## $H3K27me3
## [1] "wgEncodeBroadHistoneGm12878H3k27me3StdAln_chr20and21.narr
##
## $H3K9me3
## [1] "wgEncodeBroadHistoneGm12878H3k9me3StdAln_chr20and21.narrc
```

Next, we feed this information into the `peakEnrichment4Features()` function.

As part of the analysis, `peakEnrichment4Features()` takes a distance range (by default, the full distance range over which interactions are observed), and divides it into some number of bins. We must select the number of bins; here, we choose that number to ensure that the bin size is approximately 10kb. If the defaults are changed, a different number of bins is more appropriate. See `?peakEnrichment4Features` for more information.

```
no_bins <- ceiling(max(abs(cd@x$distSign), na.rm = T)/1e4)

enrichmentResults <- peakEnrichment4Features(cd, folder=featuresF
                                             no_bins=no_bins, sam
```



Note the plot produced by this function. For each feature type, the yellow bar represents the number of features that overlap with interaction other ends. The blue bar represents what would be expected by chance, with a 95% confidence interval for the mean number of overlaps plotted. If the yellow bar lies outside of this interval, we reject the null hypothesis, thus concluding that there is enrichment/depletion of that feature.

The information displayed in the plot is also returned in tabular form:

```
enrichmentResults
```

```
##           OLwithSI MeanOLwithSamples SDOLwithSample   LowerCI
## CTCF           352             125.54       9.659266 106.60784 1
## H3K4me1        696             280.36      14.614591 251.71540 3
## H3K4me3        378             139.91      10.549637 119.23271 1
## H3k27ac        478             169.38      11.659981 146.52644 1
## H3K27me3        73              72.29       7.111372  58.35171
## H3K9me3        233             122.53       8.435681 105.99607 1
```

The chicagoData object

In the above workflows, *cd* is a *chicagoData* object. It contains three elements:

- cd@x is a *data.table* (note: not a *data.frame*) that contains information about

fragment pairs.
- `cd@settings` is a list of settings, usually set with the setExperiment() function.
- `cd@params` is a list of parameters. This list is populated as the pipeline runs, and CHiCAGO estimates them in turn.

A closer look at `cd@x`:

```
head(cd@x, 2)
```

```
##    baitID otherEndID distbin       s_j otherEndLen distSign is
## 1: 403463     403833      NA 0.2368791        2579  1652804
## 2: 403463     403843      NA 0.2368791        6302  1690808
##    N.1 N.2 N.3 N refBinMean       s_i NNb NNboe    tlb      tb
## 1:   0   1   0 1         NA 0.9494934   4     4  [0,6] [  2, 4
## 2:   0   1   0 1         NA 1.0021336   4     4 (6,13] [  2, 4
##          Tmean      Bmean     log.p    log.w     log.q score
## 1: 0.001485762 0.08970408 -2.443145 1.406451 -3.849596     0
## 2: 0.004021026 0.09243682 -2.389579 1.350609 -3.740187     0
```

Columns:

- baitID: ID of baited fragment
- otherEndID: ID of other end fragment
- s_j: bait-specific scaling factor (Brownian noise)
- otherEndLen: The length of the other end fragment
- distSign: The distance from the baited fragment to the other end fragment. Positive and negative values indicate that the other end is, respectively, $5'$ and $3'$ of the baited fragment. NA indicates a trans interactions.
- isBait2Bait: TRUE if the other end fragment is also a baited fragment
- N.1, N.2, …: Raw read counts per replicate (see `?mergeSamples`).
- N: Merged count (see `?mergeSamples`) or raw count in the case of single-replicate interaction calling.
- refBinMean: Can be ignored. (see `?normaliseBaits`)
- s_i: other end-specific scaling factor (Brownian noise)
- NNb: “N normalised for baits”, a count scaled up by accounting for s_j. May be useful for visualization.
- NNboe: “N normalised for baits and other ends”; may be useful for visualization.
- tlb: Class of other end, based on the number of fragments on other chromosomes that have read pairs.
- tblb: As tlb, for the bait fragment.
- Tmean: Expected count from technical noise.
- Bmean: Expected count from Brownian noise. (Thus, the expected count under the null hypothesis is Tmean + Bmean.)
- log.p: p-value associated with fragment pair, on log-scale.

- log.w: p-value weight, on log-scale.
- log.q: weighted p-value, on log-scale.
- score: Final CHiCAGO score.

WARNING: Many functions in CHiCAGO update `cd@x` by reference, which means that `cd@x` can change even when you do not explicitly assign to it. To avoid this behaviour, copy the *chicagoData* object first:

```
newCd = copyCD(cd)
```

Session info

```
sessionInfo()
```

```
## R version 3.2.1 (2015-06-18)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.2 LTS
##
## locale:
##  [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_GB.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_GB.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods
##
## other attached packages:
## [1] BiocStyle_1.6.0  devtools_1.8.0   PCHiCdata_0.1.4  Chicago
## [5] data.table_1.9.4
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.11.6         formatR_1.2         git2r_0.10.1
##  [4] RColorBrewer_1.1-2  plyr_1.8.3          bitops_1.0-6
##  [7] tools_3.2.1         rpart_4.1-9         digest_0.6.8
## [10] evaluate_0.7        memoise_0.2.1       gtable_0.1.2
## [13] lattice_0.20-31     yaml_2.1.13         proto_0.3-10
## [16] gridExtra_0.9.1     knitr_1.10.5        stringr_1.0.0
## [19] cluster_2.0.2       rversions_1.0.0     grid_3.2.1
## [22] nnet_7.3-9          Delaporte_2.2-2     XML_3.98-1.1
## [25] survival_2.38-2     foreign_0.8-63      rmarkdown_0.7
## [28] latticeExtra_0.6-26 Formula_1.2-1       ggplot2_1.0.1
## [31] reshape2_1.4.1      magrittr_1.5        Hmisc_3.16-0
## [34] scales_0.2.5        matrixStats_0.14.2  htmltools_0.2.6
## [37] MASS_7.3-40         splines_3.2.1       colorspace_1.2-6
## [40] stringi_0.5-2       acepack_1.3-3.3     RCurl_1.95-4.6
## [43] munsell_0.4.2       chron_2.3-47
```

References

Mifsud, B., F. Tavares-Cadete, A. N. Young, R. Sugar, S. Schoenfelder, L. Ferreira, S. W. Wingett, et al. 2015. “Mapping long-range promoter contacts in human cells with high-resolution capture Hi-C.” *Nat. Genet.* 47 (6): 598–606.

Schoenfelder, S., M. Furlan-Magaril, B. Mifsud, F. Tavares-Cadete, R. Sugar, B. M. Javierre, T. Nagano, et al. 2015. “The pluripotent regulatory circuitry connecting promoters to their long-range interacting elements.” *Genome Res.* 25 (4): 582–97.

The ENCODE Project Consortium. 2012. “An Integrated Encyclopedia of DNA Elements in the Human Genome.” *Nature* 489 (7414): 57–74.