

# Package ‘HiCool’

April 17, 2024

**Version** 1.3.0

**Date** 2022-11-04

**Title** HiCool

**Description** HiCool provides an R interface to process and normalize Hi-C paired-end fastq reads into .(m)cool files. .(m)cool is a compact, indexed HDF5 file format specifically tailored for efficiently storing HiC-based data. On top of processing fastq reads, HiCool provides a convenient reporting function to generate shareable reports summarizing Hi-C experiments and including quality controls.

**License** MIT + file LICENSE

**URL** <https://github.com/js2264/HiCool>

**BugReports** <https://github.com/js2264/HiCool/issues>

**Depends** R (>= 4.2), HiCExperiment

**Imports** BiocIO, S4Vectors, GenomicRanges, IRanges, InteractionSet, vroom, basilisk, reticulate, rmarkdown, rmdformats, plotly, dplyr, stringr, sessioninfo, utils

**Suggests** HiContacts, HiContactsData, AnnotationHub, BiocFileCache, BiocStyle, testthat, knitr, rmarkdown

**biocViews** HiC, DNA3DStructure, DataImport

**Encoding** UTF-8

**VignetteBuilder** knitr

**LazyData** false

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Config/testthat/edition** 3

**StagedInstall** no

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

**git\_branch** devel

**git\_last\_commit** 28c1b02**git\_last\_commit\_date** 2023-10-24**Repository** Bioconductor 3.19**Date/Publication** 2024-04-17**Author** Jacques Serizay [aut, cre]**Maintainer** Jacques Serizay <jacquesserizay@gmail.com>

## Contents

getLoops . . . . .	2
HiCool . . . . .	3
HiCReport . . . . .	5
<b>Index</b>	<b>7</b>

---

getLoops	<i>Finding loops in contact map</i>
----------	-------------------------------------

---

## Description

Find loops using chromosight

## Usage

```
getLoops(
  x,
  resolution = NULL,
  output_prefix = file.path("chromosight", "chromo"),
  norm = "auto",
  max.dist = "auto",
  min.dist = "auto",
  min.separation = "auto",
  n.mads = 5L,
  pearson = "auto",
  nreads = "no",
  ncores = 1L
)
```

## Arguments

x	A HiCExperiment object
resolution	Which resolution to use to search loops
output_prefix	Prefix to chromosight output (default: "chromosight/chromo")
norm	Normalization parameter for chromosight

<code>min.dist, max.dist</code>	Min and max distance to use to filter for significant loops
<code>min.separation</code>	Minimum separation between anchors of potential loops
<code>n.mads</code>	Number of MADs to use to filter relevant bins to search for loops
<code>pearson</code>	Minimum Pearson correlation score to use to filter for significant loops
<code>nreads</code>	Number of reads to subsample to before searching for loops
<code>ncores</code>	Number of cores for chromosight

### Value

A HiCExperiment object with a new "loops" topologicalFeatures storing significant interactions identified by chromosight, and an additional chromosight\_args metadata entry.

### Examples

```
contacts_yeast <- contacts_yeast()
contacts_yeast <- getLoops(contacts_yeast)
S4Vectors::metadata(contacts_yeast)$chromosight_args
topologicalFeatures(contacts_yeast, 'loops')
```

### Description

`HiCool::HiCool()` automatically processes paired-end HiC sequencing files by performing the following steps:

1. Automatically setting up an appropriate conda environment using basilisk;
2. Mapping the reads to the provided genome reference using `hicstuff` and filtering of irrelevant pairs;
3. Filtering the resulting pairs file to remove unwanted chromosomes (e.g. chrM);
4. Binning the filtered pairs into a cool file at a chosen resolution;
5. Generating a multi-resolution mcool file;
6. Normalizing matrices at each resolution by iterative correction using cooler.

The filtering strategy used by `hicstuff` is described in Cournac et al., BMC Genomics 2012.

**Usage**

```

HiCool(
  r1,
  r2,
  genome,
  restriction = "DpnII,HinfI",
  resolutions = NULL,
  iterative = TRUE,
  balancing_args = " --min-nnz 10 --mad-max 5 ",
  threads = 1L,
  exclude_chr = "Mito|chrM|MT",
  output = "HiCool",
  keep_bam = FALSE,
  build_report = TRUE,
  scratch = tempdir()
)

importHiCoolFolder(output, hash, resolution = NULL)

getHiCoolArgs(log)

getHicStats(log)

```

**Arguments**

r1	Path to fastq file (R1 read)
r2	Path to fastq file (R2 read)
genome	Genome used to map the reads on, provided either as a fasta file (in which case the bowtie2 index will be automatically generated), or as a prefix to a bowtie2 index (e.g. mm10 for mm10.*.bt2 files). Genome can also be a unique ID for the following references: hg38, mm10, dm6, R64-1-1, GRZc10, WBce1235, Galgal4.
restriction	Restriction enzyme(s) used in HiC (Default: "DpnII,HinfI")
resolutions	Resolutions used to bin the final mcool file (Default: 5 levels of resolution automatically inferred according to genome size)
iterative	Should the read mapping be performed iteratively? (Default: TRUE)
balancing_args	Balancing arguments for cooler. See cooler documentation <a href="#">here</a> for a list of all available balancing arguments. These defaults match those used by the 4DN consortium.
threads	Number of CPUs used for parallelization. (Default: 1)
exclude_chr	Chromosomes excluded from the final .mcool file. This will not affect the pairs file. (Default: "Mito chrM MT")
output	Output folder used by HiCool.
keep_bam	Should the bam files be kept? (Default: FALSE)
build_report	Should an automated report be computed? (Default: TRUE)
scratch	Path to temporary directory where processing will take place. (Default: tempdir())

hash	Unique 6-letter ID used to identify files from a specific HiCool processing run.
resolution	Resolution used to import the mcool file
log	Path to log file generated by hicstuff/hicool

**Value**

A CoolFile object with prefilled pairsFile and metadata slots.

**HiCool utils**

- `importHiCoolFolder(folder, hash)` automatically finds the different processed files associated with a specific `HiCool::HiCool()` processing hash ID.
- `getHiCoolArgs()` parses the log file generated by `HiCool::HiCool()` during processing to recover which arguments were used.
- `getHicStats()` parses the log file generated by `HiCool::HiCool()` during processing to recover pre-computed stats about pair numbers, filtering thresholds, etc.

**Examples**

```
r1 <- HiContactsData::HiContactsData(sample = 'yeast_wt', format = 'fastq_R1')
r2 <- HiContactsData::HiContactsData(sample = 'yeast_wt', format = 'fastq_R2')
hcf <- HiCool(r1, r2, genome = 'R64-1-1', output = './HiCool/')
hcf
getHiCoolArgs(S4Vectors::metadata(hcf)$log)
getHicStats(S4Vectors::metadata(hcf)$log)
readLines(S4Vectors::metadata(hcf)$log)
```

---

HiCReport

*HiC processing report*


---

**Description**

HiC processing report

**Usage**

```
HiCReport(x, output = NULL)
```

**Arguments**

x	an CoolFile object, generated from <code>HiCool::HiCool()</code> or <code>HiCool::importHiCoolFolder()</code> , or directly from calling <code>HiCExperiment::CoolFile()</code> .
output	Path to save output HTML file.

**Value**

String to the generated HTML report file

**Examples**

```
mcool_path <- HiContactsData::HiContactsData('yeast_wt', 'mcool')
pairs_path <- HiContactsData::HiContactsData('yeast_wt', 'pairs.gz')
log_path <- HiContactsData::HiContactsData(sample = 'yeast_wt', format = 'HiCool_log')
cf <- CoolFile(mcool_path, pairs = pairs_path, metadata = list(log = log_path))
HiCReport(cf)
```

# Index

`getHiCoolArgs (HiCool)`, [3](#)

`getHicStats (HiCool)`, [3](#)

`getLoops`, [2](#)

`HiCool`, [3](#)

`HiCReport`, [5](#)

`importHiCoolFolder (HiCool)`, [3](#)