

HiCue

User Manual

Version 0.4.2

Maelys Delouis
Institut Pasteur
maelys.delouis@pasteur.fr

1. Overview

HiCue is an open-source Python command-line tool for the extraction, aggregation, and visualisation of chromatin interaction data produced by Hi-C and Micro-C experiments. It reads data stored in the widely used Cooler format (.cool and .mcool files) and produces publication-quality figures such as pileup (aggregate) contact maps, individual submatrix panels, and batched overviews.

The primary use case is pileup analysis: given a set of genomic positions of interest (e.g. loop anchors, gene starts, CTCF binding sites), HiCue extracts the contact submatrix centred on each position, aggregates all submatrices by median or mean, and renders the result as a heatmap. This statistical averaging approach greatly reduces noise and reveals the average spatial organisation around a class of regulatory elements.

In addition to standard pileup computation, HiCue supports:

- Overlay of continuous genomic signals (BigWig tracks) on contact maps.
- Distance-law detrending ($P(s)$ normalisation) and null-model detrending (patch subtraction) to remove background contact frequency biases.
- Strand-aware display: submatrices can be flipped to align forward- and reverse-strand features before aggregation.
- Flexible position separation by transcription direction, chromosome, or user-defined genomic regions.
- Multi-resolution support through .mcool files, allowing the same analysis to be repeated at multiple bin sizes in a single command.
- Parallel execution via a multi-threaded producer-consumer pipeline, enabling efficient processing of large datasets.

HiCue is designed for researchers working in genomics and epigenomics who need a scriptable, reproducible workflow for Hi-C data exploration without the overhead of a graphical interface.

2. Quickstart

The commands below install HiCue and run a minimal pileup extraction on a set of loop anchors. They assume Python 3.10 or later is already installed.

Install

```
pip install hicue
hicue --help
```

Run a basic pileup

```
# Extract a pileup centred on loop anchors, 50 kb window, 1 kb resolution
hicue extract results/ anchors.bed experiment.mcool \
    --windows 50000 \
    --binnings 1000
```

This single command will create a results/ directory containing one PDF pileup figure per Cool file, organised by binning resolution. No other configuration is required for a first run.

3. Installation

3.1 Requirements

HiCue requires:

- Python 3.10 or later (the match/case statement used internally requires this minimum version).
- The following Python packages, all installed automatically by pip: cooler (≥ 0.10), pyBigWig, bcbio-gff, click, numpy, pandas (≥ 1.5), matplotlib, chromosight.

3.2 Installation from PyPI

The recommended installation method uses pip:

```
pip install hicue
```

It is strongly recommended to install HiCue inside a dedicated virtual environment to avoid dependency conflicts:

```
python -m venv hicue-env
source hicue-env/bin/activate # Linux / macOS
hicue-env\Scripts\activate   # Windows
pip install hicue
```

Or in a conda environment:

```
conda create -n hicue-env
conda activate hicue-env
pip install hicue
```

3.3 Installation from source (development)

To install the latest development version or to contribute to the project:

```
git clone https://github.com/Mae-4815162342/HiCue.git
cd HiCue
pip install -e .[dev]
```

The `-e` flag installs the package in editable mode so that changes to the source code are reflected immediately without reinstalling.

3.4 Verifying the installation

```
hicue --help
```

A help message listing the available sub-commands (extract, tracks, regions) should be printed. If the command is not found, ensure that the virtual environment is activated and that the Python scripts directory is in your PATH.

3.5 Optional: running the test suite

A test suite is provided for developers who want to validate a local installation:

```
pip install hicue[test]
pytest tests/
```

4. Features

HiCue exposes three main commands. Each command shares a large set of common options and adds a small number of mode-specific parameters.

4.1 extract – Pileup from a position file

The extract command is the primary workflow. It takes a genomic position file (BED, BED2D, BEDPE, or GFF/GTF) and one or more Cooler files (provided as a comma-separated list or as a .txt file with a Cooler file per line), extracts contact submatrices centred on every listed locus, and aggregates them into a pileup figure.

Syntax:

```
hicue extract OUTPUT_DIR POSITIONS COOL_FILES... [OPTIONS]
```

Key options:

Option	Type	Default	Description
--pileup / --no-pileup	flag	on	Compute and display aggregate pileup figures.
--loci / --no-loci	flag	off	Save individual submatrix figures, one per locus.
--batch / --no-batch	flag	off	Save batched submatrix figures (up to 64 per page).
-w / --windows	int,...	30000	Half-window size(s) in bp. Multiple values accepted as comma-separated list.
-b / --binnings	int,...	1000	Bin size(s) in bp for .mcool files. Multiple values accepted.
-d / --detrending	choice	none	Detrending strategy: none, ps (distance-law), or patch (null-model subtraction).
-m / --method	choice	median	Aggregation method for pileup: median, mean, or sum.
-f / --flip	flag	off	Flip reverse-strand submatrices to strand-normalise the pileup.
-r / --raw	flag	off	Use raw (unbalanced) contact counts.
--loops	flag	off	Centre submatrices on pairs of loci (BED2D / loop-anchor mode).
--trans	flag	off	Include trans-chromosomal contacts when --loops is active.
--separate_by	str,...	(none)	Stratify results by: direction, regions, or chroms.
--gff	path	(none)	GFF/GTF annotation file for automatic position labelling.
--threads	int	8	Number of worker threads for parallel extraction.

<code>--format</code>	str,...	pdf	Output figure format(s): pdf, png, svg, etc.
<code>--cmap_color</code>	str	seismic	Matplotlib colormap for pileup figures.
<code>--center</code>	choice	start	Which positional anchor to use: start, center, or end of each feature.
<code>--save_tmp</code>	flag	off	Save intermediate CSV files (positions, pairs) to the output directory.

4.2 tracks – Pileup from a BigWig signal

The tracks command is equivalent to extract but derives genomic positions automatically from peaks detected in a BigWig track file. This is useful when positions of interest are defined by a continuous signal (e.g. ChIP-seq or ATAC-seq enrichment) rather than a pre-computed coordinate list. Note that in the case a positions file is provided, the peak selection will be restricted to the positions it contains.

Syntax:

```
hicue tracks OUTPUT_DIR TRACK_FILE COOL_FILES... [OPTIONS]
```

All options from extract are supported. Additional tracks-specific options:

Option	Type	Default	Description
<code>-t / --threshold</code>	choice+float	(none)	Absolute signal threshold: min keeps values above the threshold, max keeps values below.
<code>-p / --percentage</code>	choice+int	(none)	Keep the top (high) or bottom (low) N% of positions by signal value.
<code>--min_sep</code>	int	1000	Minimum distance in bp between two retained peaks, to avoid redundant selection.
<code>--positions</code>	path	(none)	Restrict peak selection to positions listed in an additional BED/GFF file.
<code>--gff_type</code>	str	""	Feature type to select when --positions is a GFF file.

4.3 regions – Variable-size region pileup

The regions command extracts and aggregates submatrices of variable size, resizing each one to a common pixel dimension before computing the pileup. This is designed for features of different genomic lengths (e.g. genes, TADs, compartments) where a fixed-window approach would be inappropriate.

Syntax:

```
hicue regions OUTPUT_DIR POSITIONS COOL_FILES... [OPTIONS]
```

All options from extract are supported. Additional regions-specific options:

Option	Type	Default	Description
-p / --padding	float	1.0	Padding ratio added symmetrically around each region. A value of 0.5 adds 50% of the region size on each side.
-s / --min_region_size	int	20000	Minimum region size in bp. Regions smaller than this value are skipped.
-e / --expected_sizes	int,...	51	Target pixel dimensions for resizing. Multiple values produce separate outputs.

5. Methods

This section describes the internal processing steps and the class architecture of HiCue.

5.1 Pipeline overview

HiCue uses a multi-threaded producer-consumer architecture. All inter-stage communication passes through Python queue.Queue objects with a DONE sentinel value to signal termination. The pipeline for extract and tracks proceeds as follows:

1. FileStreamer reads the input position or track file line by line and pushes lines to a shared queue.
2. Parser workers consume lines and emit (i) individual position records and (ii) pairings of position indices (i, j) representing the contact to extract.
3. Annotator workers (optional) enrich each position with strand information and feature labels from GFF/GTF or BigWig annotation files.
4. Separator workers stratify positions into groups (by direction, chromosome, or custom region) according to `--separate_by`.
5. PairFormater workers consume the pairing queue and resolve each (i, j) index pair into concrete genomic coordinates, applying distance filters and circular-chromosome wrapping.
6. Extracter workers query the Cooler file for each resolved pair and retrieve the raw contact submatrix.
7. SubmatrixFormater workers normalise, detrend, and resize each submatrix.
8. Aggregator workers collect formatted submatrices into the shared Pileup object.
9. Display workers (AsyncDisplays) render figures asynchronously on per-thread event loops to avoid matplotlib contention.

5.2 Detrending methods

Three detrending strategies are available through the `--detrending` option:

none (default)

No background correction is applied. The raw balanced contact matrix is used directly. Appropriate when comparing loci within the same distance class or when absolute contact frequency is meaningful.

ps (P(s) normalisation)

Each submatrix is divided element-wise by the expected contact frequency at the corresponding genomic distance, as estimated from the genome-wide distance-decay curve $P(s)$. This removes the strong background created by the polymer scaling law and makes enrichments at specific distances directly comparable across experiments.

By default the $P(s)$ curve is estimated across all cis chromosomes. Set `--ps_all_chrom False` to compute a per-chromosome $P(s)$ curve instead.

```
# Example: P(s) detrending at two resolutions
```



```
hicue extract results/ anchors.bed experiment.mcool \  
  --detrending ps \  
  --binnings 1000,5000 \  
  --windows 50000
```

patch (null-model subtraction)

For each focal locus, HiCue selects `--nb_pos` random loci at a similar genomic distance (within `--rand_max_dist` bp) and extracts their submatrices as a null model. The aggregate null matrix is then subtracted from the signal pileup. This approach is particularly useful for loop detection studies where local background must be estimated precisely.

The random positions can be saved and reused across runs with `--save_tmp` and `--random_path` to ensure reproducibility.

```
# Example: patch detrending with 3 random positions and a 150 kb search radius  
hicue extract results/ loops.bed2d experiment.mcool \  
  --loops \  
  --detrending patch \  
  --nb_pos 3 \  
  --rand_max_dist 150000 \  
  --binnings 1000
```

5.3 Separation strategies

The `--separate_by` option accepts one or more of the following keywords as a comma-separated list. Results are written into separate sub-directories for each group.

direction

Positions are grouped by their strand annotation (forward strand vs. reverse strand). Requires strand information in the input BED file (column 6) or from a GFF annotation. Combine with `--flip` to align all features in the same orientation before aggregation.

```
hicue extract results/ genes.bed experiment.mcool \  
  --separate_by direction --flip
```

regions

Positions are attributed to user-defined genomic intervals provided as a CSV file via `--separation_regions`. The CSV format is: Id, Chromosome, Start, End. A single feature can span multiple rows with the same Id, allowing discontinuous intervals.

```
hicue extract results/ genes.bed experiment.mcool \  
  --separate_by regions \  
  --separation_regions compartments.csv
```

chroms

Each chromosome is treated as a separate group. Useful for identifying chromosome-specific contact patterns or for quality-control purposes.

```
hicue extract results/ genes.bed experiment.mcool \  
  --separate_by chroms
```

5.4 Class architecture

The following table summarises the main classes, their locations, and their roles:

Class	Module	Role
Reader	<code>classes/Reader.py</code>	Entry point for reading position files and launching the parsing pipeline.
Pileup	<code>classes/Pileup.py</code>	Thread-safe accumulator of submatrices. Computes the final aggregate by median, mean, or sum.
PairFormatter	<code>classes/PairFormatter.py</code>	Resolves (i, j) index pairs into genomic coordinates, applies distance filters and circular-chromosome logic.
RandomSelector	<code>classes/RandomSelector.py</code>	Selects random control loci at a specified genomic distance for patch detrending.
TrackReader	<code>classes/TrackReader.py</code>	Reads BigWig files and applies threshold or percentage-based peak selection.
AsyncDisplays	<code>classes/AsyncDisplays.py</code>	Manages asynchronous matplotlib rendering in worker threads using persistent event loops.
MatrixExtractorLauncher	<code>classes/MatrixExtractorLauncher.py</code>	Orchestrates the parallel extraction pool and distributes work across Extracter threads.
PositionList	<code>classes/PositionList.py</code>	Lightweight wrapper around the positions DataFrame for downstream queries.
FileStreamer (worker)	<code>workers/FileStreamer.py</code>	Streams lines from the input file to a queue.
Parser (worker)	<code>workers/Parser.py</code>	Parses BED, BED2D, BEDPE, GFF, and BigWig lines into position records.
Annotator (worker)	<code>workers/Annotator.py</code>	Enriches positions with GFF feature labels and BigWig signal values.
Separator (worker)	<code>workers/Separator.py</code>	Routes positions into group-specific queues based on <code>--separate_by</code> .

Extractor (worker)	workers/Extractor.py	Queries the Cooler file and retrieves a contact submatrix for each pair.
SubmatrixFormater (worker)	workers/SubmatrixFormater.py	Normalises, detrends, and optionally resizes each submatrix before aggregation.
Aggregator (worker)	workers/Aggregator.py	Feeds formatted submatrices into the Pileup object.

5.5 Output structure

HiCue organises all output files under the specified output directory following a deterministic hierarchy:

```
OUTPUT_DIR/  
  {cool_name}/  
    {sep_id}/  
      binning_{binning}/  
        individual_{window}kb_window/  
          GeneName.pdf          <- per-locus submatrix  
          ...  
        batched_{window}kb_window/  
          batch#1.pdf           <- 64-panel batch figure  
          batch#1_references.csv  
          ...  
        pileup_{window}kb_window.pdf
```

A timestamped log file (YYYYMMDD_HHMMSS_log.txt) is also written to OUTPUT_DIR for every run, recording the exact command and all option values.

6. Usage

The examples below use synthetic file names. Replace them with your actual paths when reproducing these analyses. Will be replaced by the test dataset upon cleaning.

6.1 extract – Worked examples

Example 1: Basic pileup on gene TSS

Compute a median pileup centred on the transcription start sites of a set of genes, using balanced contacts at 1 kb resolution with a 50 kb half-window.

```
hicue extract results/gene_tss/ genes_hg38.bed control.mcool \  
  --windows 50000 \  
  --binnings 1000 \  
  --center start
```

Expected output: results/gene_tss/control/default/binning_1000/pileup_50kb_window.pdf

Example 2: Strand-aware pileup with flip

Same as Example 1 but with submatrices flipped so that the transcription direction is always left-to-right. Individual locus figures are also saved.

```
hicue extract results/gene_tss_flipped/ genes_hg38.bed control.mcool \  
  --windows 50000 \  
  --binnings 1000 \  
  --center start \  
  --flip \  
  --loci
```

Example 3: Loop pileup with P(s) detrending

Extract contact matrices centred on pairs of loop anchors listed in a BED2D file, apply P(s) detrending, and save results in both PDF and PNG.

```
hicue extract results/loops_ps/ loops_hg38.bed2d treated.mcool control.mcool \  
  --loops \  
  --detrending ps \  
  --windows 50000,100000 \  
  --binnings 1000 \  
  --format pdf,png
```

Example 4: Pileup separated by chromosome

Compute one pileup per chromosome to check for chromosome-specific patterns.

```
hicue extract results/by_chrom/ genes_hg38.bed control.mcool \  
  --windows 50000 \  
  --binnings 5000 \  
  --separate_by chroms
```

Example 5: Patch detrending with saved random positions

Run patch detrending on a first experiment and save the random control positions. Re-use those same positions on a second experiment for a paired comparison.

```
# First run: generate and save random positions
```

```
hicue extract results/patch_ctrl/ loops.bed2d control.mcool \  
  --loops \  
  --detrending patch \  
  --nb_pos 3 \  
  --save_tmp  
  
# Second run: reuse the same random positions  
hicue extract results/patch_treated/ loops.bed2d treated.mcool \  
  --loops \  
  --detrending patch \  
  --nb_pos 3 \  
  --random_path results/patch_ctrl/loops
```

6.2 tracks – Worked examples

Example 6: Pileup on ChIP-seq peaks

Select the top 20% of Smc1 ChIP-seq signal bins and compute a pileup centred on those positions.

```
hicue tracks results/smc1_peaks/ Smc1_chipseq.bw experiment.mcool \  
  --percentage high 20 \  
  --windows 50000 \  
  --binnings 1000
```

Example 7: Peak selection with a minimum signal threshold

Retain only positions with a normalised signal above 3.5 and a minimum separation of 5 kb between peaks.

```
hicue tracks results/high_signal/ H3K27ac.bw experiment.mcool \  
  --threshold min 3.5 \  
  --min_sep 5000 \  
  --windows 30000 \  
  --binnings 1000
```

Example 8: Constrained selection to annotated genes

Apply the BigWig threshold only within gene bodies defined by a GFF file, and show individual locus figures in addition to the pileup.

```
hicue tracks results/gene_signal/ RNAPol2.bw experiment.mcool \  
  --positions genes.gff \  
  --threshold min 5.0 \  
  --windows 50000 \  
  --binnings 1000 \  
  --loci --batch
```

6.3 regions – Worked examples

Example 9: Variable-size pileup on TADs

Extract and resize contact matrices for a set of topologically associating domains (TADs) of variable size, using a 51 x 51 pixel target and a padding ratio of 0.5.

```
hicue regions results/tads/ tads_hg38.bed experiment.mcool \  
  --expected_sizes 51 \  
  --padding 0.5
```

```
--padding 0.5 \  
--binnings 5000
```

Example 10: Multi-resolution region pileup

Run the same TAD pileup at 5 kb and 10 kb resolutions simultaneously, requesting both 20-pixel and 51-pixel output sizes.

```
hicue regions results/tads_multi/ tads_hg38.bed experiment.mcool \  
  --expected_sizes 20,51 \  
  --binnings 5000,10000 \  
  --padding 1.0
```

7. Input File Formats

Format	Extension(s)	Notes
BED	.bed	Minimum 3 columns: chrom, start, end. Column 4 = name, column 6 = strand (+/-). Required for -flip and --separate_by direction.
BED2D / BEDPE	.bed2d, .bedpe	6-column format: chrom1, start1, end1, chrom2, start2, end2. Used with --loops. Name column (7) optional.
GFF / GTF	.gff, .gtf	Standard feature annotation. Used for position annotation via --gff or as a primary position file in tracks mode.
Cooler (single)	.cool	Single-resolution Hi-C matrix. The --binnings option is ignored for .cool files.
Cooler (multi)	.mcool	Multi-resolution Hi-C matrix. Use --binnings to select one or more resolutions.
Text	.txt	File containing the .cool/.mcool files to use as input, as a list of one file per line. All .cool/.mcool files provided in the .txt will be treated simultaneously by the command.
BigWig	.bw	Continuous genomic signal track. Used as the primary input in tracks mode or for annotation overlay.
Regions CSV	.csv	Comma-separated: Id, Chromosome, Start, End. Used with --separate_by regions.

8. Glossary

Term	Definition
BED	Browser Extensible Data: a tab-separated text format for storing genomic intervals.
BED2D / BEDPE	An extension of BED for paired genomic intervals (e.g. chromatin loop anchors).
BigWig (.bw)	A binary genome-wide signal format commonly used for ChIP-seq, ATAC-seq, or RNA-seq coverage tracks.
Cooler (.cool / .mcool)	A hierarchical data format (HDF5-based) for storing sparse Hi-C or Micro-C contact matrices at one or multiple resolutions.
GFF / GTF	General Feature Format: a standard format for genome annotation (genes, exons, regulatory elements).
Hi-C / Micro-C	Chromosome conformation capture techniques that measure genome-wide 3D chromatin contacts.
Patch detrending	A null-model approach that subtracts a pileup of randomly selected control loci from the signal pileup.
Pileup	The aggregate contact matrix obtained by averaging submatrices extracted around a set of genomic positions.
P(s)	Contact probability as a function of genomic distance. Dividing a raw contact matrix by P(s) removes the distance-decay background.
Submatrix	A square slice of the genome-wide contact matrix centred on a single genomic locus or pair of loci.
TAD	Topologically Associating Domain: a genomic region within which contacts are enriched compared to the surrounding genome.

9. Known Issues & Troubleshooting

9.1 Blank or corrupted output figures

Symptom: output PDF or PNG files are blank, contain the wrong data, or the process crashes with a RuntimeError from the Agg renderer.

Cause: a race condition in the previous asynchronous display implementation when `asyncio.run()` was called inside worker threads.

Fix: applied in version 0.4.2. Each display worker now maintains a single persistent event loop. Ensure you are running HiCue \geq 0.4.2.

9.2 Missing binning resolution in .mcool file

Symptom: HiCue raises an error such as Resolution X not found in the .mcool file.

Fix: list the available resolutions with `cooler ls experiment.mcool` and pass one of the listed values to `--binnings`.

```
cooler ls experiment.mcool
# Output: /resolutions/1000 /resolutions/5000 /resolutions/10000

hicue extract results/ anchors.bed experiment.mcool --binnings 5000
```

9.3 No positions selected in tracks mode

Symptom: HiCue reports zero positions after applying the threshold or percentage filter.

Fix: check the signal distribution in your BigWig file and relax the threshold. Use `--percentage high 50` as a starting point.

9.4 Memory usage with large position sets

HiCue pre-allocates a float32 matrix of shape $(N_{\text{loci}}, \text{window_pixels}^2)$ inside the Pileup class. For very large locus sets ($> 100,000$) at high resolution this can require several gigabytes of RAM. Reduce `--threads` or split the position file into smaller subsets if memory is limited.