# hictk Documentation

**Roberto Rossini**

**Nov 12, 2025**

# INSTALLATION

hictk is a blazing fast toolkit to work with .hic and .cool files.

hictk is capable of reading and writing files in .cool, .mcool, .scool, and .hic format (including hic v9).

CHAPTER

# ONE

# DOCUMENTATION FORMATS

You are reading the PDF version of the documentation.

The live HTML version of the documentation is available at https://hictk.readthedocs.io/en/stable/.

## Installation

hictk is developed on Linux and tested on Linux, macOS, and Windows. CLI tools can be installed in several different ways. Refer to *Installation* for more details.

hictk can be compiled on most UNIX-like systems (including many Linux distributions and macOS) as well as Windows. See the *build instructions* for more details.

hictk can be used from languages other than C++ through the following bindings:

- Python bindings through hictkpy
- R bindings through hictkR

## How to cite this project?

Please use the following BibTeX template to cite hictk in scientific discourse:

```
@article{hictk,
    author = {Rossini, Roberto and Paulsen, Jonas},
    title = "{hictk: blazing fast toolkit to work with .hic and .cool files}",
    journal = {Bioinformatics},
    volume = {40},
    number = {7},
    pages = {btae408},
    year = {2024},
    month = {06},
    issn = {1367-4811},
    doi = {10.1093/bioinformatics/btae408},
    url = {https://doi.org/10.1093/bioinformatics/btae408},
    eprint = {https://academic.oup.com/bioinformatics/article-pdf/40/7/btae408/
↪58385157/btae408.pdf},
}
```

CHAPTER

TWO

# INSTALLATION

## 2.1 Conda (bioconda)

The hictk package for Linux and macOS is available on bioconda and can be installed as follows:

```
user@dev:/tmp$ conda create -n hictk -c conda-forge -c bioconda hictk

user@dev:/tmp$ conda activate hictk

(hictk) user@dev:/tmp$ whereis hictk
hictk: /home/user/.miniconda3/envs/hictk/bin/hictk

(hictk) user@dev:/tmp$ hictk --version
hictk-v2.2.0-bioconda
```

## 2.2 Containers (Docker or Singularity/Apptainer)

First, ensure you have followed the instructions on how to install Docker or Singularity/Apptainer on your OS.

The following instructions assume you have root/admin permissions.

- Linux
- macOS
- Windows

On some Linux distributions, simply installing Docker is not enough. You also need to start (and optionally enable) the appropriate service(s). This is usually done with one of the following:

```
sudo systemctl start docker
sudo systemctl start docker.service
```

Refer to Docker or your OS/distribution documentation for more details.

### 2.2.1 Pulling hictk Docker image

hictk Docker images are available on GHCR.io and DockerHub.

Downloading and running the latest stable release can be done as follows:

```
# Using Docker, may require sudo
user@dev:/tmp$ docker run ghcr.io/paulsengroup/hictk:2.2.0 --help

# Using Singularity/Apptainer
user@dev:/tmp$ singularity run ghcr.io/paulsengroup/hictk:2.2.0 --help

Blazing fast tools to work with .hic and .cool files.
```

(continues on next page)

(continued from previous page)

```
Usage: hictk [OPTIONS] SUBCOMMAND
Options:
  -h,--help                   Print this help message and exit
  -V,--version                Display program version information and exit
Subcommands:
  balance                     Balance Hi-C files using ICE, SCALE, or VC.
  convert                     Convert Hi-C files between different formats.
  dump                        Read interactions and other kinds of data from .hic and␣
↪Cooler files and write them to stdout.
  fix-mcool                   Fix corrupted .mcool files.
  load                        Build .cool and .hic files from interactions in various␣
↪text formats.
  merge                       Merge multiple Cooler or .hic files into a single file.
  metadata                    Print file metadata to stdout.
  rename-chromosomes, rename-chroms
                              Rename chromosomes found in Cooler files.
  validate                    Validate .hic and Cooler files.
  zoomify                     Convert single-resolution Cooler and .hic files to␣
↪multi-resolution by coarsening.
```

The above will print hictk's help message, and is equivalent to running `hictk --help` from the command line (assuming hictk is available on your machine).

## 2.3 Installing from source

Please refer to hictk's *build instructions*.

CHAPTER

THREE

# INSTALLATION (SOURCE)

Instructions assume hictk is being built on a UNIX environment.

Building on Windows follows the same logic but some of the commands may be slightly different.

## 3.1 Build instructions

hictk can be compiled on most UNIX-like systems (including many Linux distributions, macOS) and Windows.

### 3.1.1 Build requirements

Compiling hictk requires a compiler toolchain supporting C++17, such as:

- GCC 8+
- Clang 8+
- Apple-Clang 10.0+
- MSVC 19.12+

Based on our testing, hictk binaries compiled on Linux using Clang are noticeably faster than those compiled with GCC. For this reason, we recommend building hictk using a modern version of Clang whenever possible.

Furthermore, the following tools are required:

- CMake 3.25+
- Conan 2+
- git 2.7+
- make or ninja
- Python3.6+ (including `pip`, required to install Conan)

We recommend installing CMake and Conan in a Python virtualenv, but you are free to install build dependencies however you prefer.

```
python3 -m venv /tmp/venv
/tmp/venv/bin/python3 -m pip install pip setuptools --upgrade
/tmp/venv/bin/python3 -m pip install 'cmake>=3.25' 'conan>=2' ninja

# NOTE: It is important to activate the venv after installing CMake
. /tmp/venv/bin/activate

whereis cmake  # cmake: /tmp/venv/bin/cmake
whereis conan  # conan: /tmp/venv/bin/conan
whereis ninja  # ninja: /tmp/venv/bin/ninja

cmake --version
conan --version
```

(continues on next page)

(continued from previous page)

```
# Detect the compiler toolchain. It is generally recommended to explicitly set CC and␣
↪CXX
# Use CC=gcc CXX=g++ if clang is not installed on your machine
CC=clang CXX=clang++ conan profile detect --force
```

### 3.1.2 Getting the source code

Download from the Release page (recommended).

```
mkdir /tmp/hictk
curl -L 'https://github.com/paulsengroup/hictk/archive/refs/tags/v2.2.0.tar.gz' | tar␣
↪--strip-components=1 -C /tmp/hictk -xzf -
```

Using git.

```
git clone https://github.com/paulsengroup/hictk.git /tmp/hictk

cd /tmp/hictk
git checkout v2.2.0  # Skip this step if you want to build the latest commit from main
```

### 3.1.3 Compiling hictk

```
# Activate venv
. /tmp/venv/bin/activate

# Set these variables to the number of CPU cores available on your machine
# You can check this with e.g.
# python -c 'import multiprocessing as mp; print(mp.cpu_count())'
export CONAN_CPU_COUNT=8
export CMAKE_BUILD_PARALLEL_LEVEL=8
export CMAKE_POLICY_VERSION_MINIMUM=3.5

# Install/build dependencies with Conan
conan install --build=missing \
              -pr default \
              -s build_type=Release \
              -s compiler.cppstd=17 \
              --output-folder=./build/ \
              -o 'hictk/*:with_cli_tool_deps=True' \
              -o 'hictk/*:with_telemetry_deps=True' \
              -o 'hictk/*:with_unit_testing_deps=True' \
              .


# Do not pass -G Ninja if you want CMake to use make instead of ninja
# Use clang whenever possible, as that usually leads to significantly faster hictk␣
↪binaries.
# If clang is not installed on your machine, then replace clang and clang++ with e.g.,
↪ gcc and g++
# -DCMAKE_C_COMPILER=gcc
# -DCMAKE_CXX_COMPILER=g++
cmake -DCMAKE_BUILD_TYPE=Release \
      -DCMAKE_C_COMPILER=clang \
      -DCMAKE_CXX_COMPILER=clang++ \
      -DCMAKE_PREFIX_PATH="$PWD/build" \
```

(continues on next page)

(continued from previous page)

```
      -DHICTK_ENABLE_TESTING=ON \
      -DHICTK_ENABLE_FUZZY_TESTING=OFF \
      -DHICTK_BUILD_TOOLS=ON \
      -DHICTK_BUILD_BENCHMARKS=OFF \
      -DHICTK_BUILD_EXAMPLES=OFF \
      -DHICTK_WITH_ARROW=OFF \
      -DHICTK_WITH_EIGEN=OFF \
      -G Ninja \
      -S /tmp/hictk \
      -B /tmp/hictk/build

cmake --build /tmp/hictk/build

# If you are compiling hictk on Windows you need to pass the build config as well
# cmake --build /tmp/hictk/build --config Release
```

### Troubleshooting build errors

- I get an error while building `boost` with Conan:

  If you are getting an error like:

  ```
  ConanException: These libraries were built, but were not used in any boost module
  ```

  This is likely due to Conan deciding to use a buggy version of `b2` (e.g., v5.3.0) to build `boost`.

  You can work around this by overriding the version of `b2` in your Conan profile.

  To do this:

  1. Locate the Conan profile with e.g., `conan profile path default`
  2. Add the following lines at the end of the profile:

     ```
     [tool_requires]
     boost/*: b2/5.2.1
     ```

- When building dependencies with Conan I am getting errors like:

  ```
  b2: relocation error: b2: symbol _ZNSt7__cxx1112basic_stringIcSt11char_
  ↪traitsIcESaIcEE10_M_replaceEmmPKcm, version GLIBCXX_3.4.21 not defined in file␣
  ↪libstdc++.so.6 with link time reference
  ```

  This is due to ABI incompatibilities between your build environment and the environment used by the Conan Center Index to build e.g., `b2`.

  You can work around this bug by forcing Conan to build `b2` (and any other package causing similar errors) from source:

  ```
  conan install -pr:b default -pr:h default --requires 'b2/5.2.1'
  ```

  For the `b2` package specifically, compiling the package with `clang` on Linux is prone to issues and often fails.

  If you run into problems, try compiling `b2` using gcc instead:

  ```
  CC=gcc CXX=g++ conan profile detect --name gcc --exist-ok
  conan install -pr:b gcc -pr:h gcc --requires 'b2/5.2.1'
  ```

### Tweaking hictk's build options

hictk build can be customized by providing one or more build flags when invoking CMake. All hictk-specific build options are defined in CMakeLists.txt file located in the project root.

If you only want to build the hictk executable, then you should pass `-DHICTK_WITH_ARROW=OFF` and `-DHICTK_WITH_EIGEN=OFF`. You may also want to remove `arrow`, `boost`, `eigen`, and `pybind11` from the list of requirements in the `conanfile.py`.

We always recommend building and running the unit tests. However, if you are really sure you do not want to build the tests, feel free to also pass `-DHICTK_ENABLE_TESTING=OFF`.

If you do not intend to run any of the automated tests, then you can also disable the automatic download of the test datasets with `-DHICTK_DOWNLOAD_TEST_DATASET=OFF`.

Table 1: Complete list of build options

| Option | Description | Default |
|---|---|---|
| BUILD_SHARED_LIBS | Build library and binaries using dynamic linking | OFF |
| HICTK_ENABLE_TESTING | Build the suite of unit tests | ON |
| HICTK_ENABLE_FUZZY_TESTING | Build hictk's fuzzer | OFF |
| HICTK_ENABLE_GIT_VERSION_TRACKI | Attempt to retrieve project version and metadata from git | ON |
| HICTK_BUILD_EXAMPLES | Build hictk's example programs | OFF |
| HICTK_BUILD_BENCHMARKS | Build hictk benchmarks | OFF |
| HICTK_WITH_ARROW | Build with Arrow support | ON |
| HICTK_WITH_ARROW_SHARED | Force dynamic linking to Arrow libraries | OFF |
| HICTK_WITH_EIGEN | Build with Eigen3 support | ON |
| HICTK_BUILD_TOOLS | Build the hictk binary | ON |
| HICTK_DOWNLOAD_TEST_DATASET | Download datasets required by unit and integration tests | ON |
| HICTK_ENABLE_TELEMETRY | Build CLI tools with support for telemetry | ON |

There are several other options that are disabled unless `-DENABLE_DEVELOPER_MODE=ON`. Keep in mind that enabling this option will enable the sanitizers, pass `-Werror` to the compiler, run clang-tidy, and much more. Binaries compiled with `-DENABLE_DEVELOPER_MODE=ON` are much bigger and slower than the regular binaries, and should only be used for debugging purposes. To further tweak the options enabled when `-DENABLE_DEVELOPER_MODE=ON`, please use `ccmake` and have a look at options prefixed with `OPT_`.

Finally, you can override the version metadata embedded in the hictk binary and version headers by tweaking any of the `HICTK_GIT_` variables defined in file cmake/Versioning.cmake (e.g., `-DHICTK_GIT_AUTHOR_NAME='Alan Turing'`).

### Tweaking hictk's dependencies

The `conanfile.py` file contains the dependencies to compile the entire `hictk` project.

Most users don't require all the dependencies.

The following table shows the dependency matrix for `hictk` and `libhictk`.

| Feature | HICTK_BUILD_ | HICTK_WITH_ | HICTK_WITH_ | HICTK_ENABLE_ | HICTK_ENABLE_FUZZY_TESTING= |
|---|---|---|---|---|---|
| arrow | N | Y | N | Y4 | Y4 |
| boost | Y3 | Y3 | N | Y4 | Y4 |
| bshoshany-thread-pool | Y | Y | Y | Y | Y |
| bzip2 | Y1 | N | N | N | N |
| catch2 | N | N | N | Y | N |
| cli11 | Y | N | N | N | Y |
| concurren-tqueue | Y | Y | Y | Y | Y |
| eigen | N | N | Y | Y4 | Y4 |
| fast_float | Y | Y | Y | Y | Y |
| fmt | Y | Y | Y | Y | Y |
| hdf5 | Y | Y | Y | Y | Y |
| highfive | Y | Y | Y | Y | Y |
| libarchive | Y | N | N | N | N |
| libcurl | Y2 | N | N | N | N |
| libdeflate | Y | Y | Y | Y | Y |
| lz4 | Y1 | N | N | N | N |
| lzo | Y1 | N | N | N | N |
| nlohmann_js | Y | N | N | N | N |
| opentelemeti cpp | Y2 | N | N | N | N |
| parallel-hashmap | Y | Y | Y | Y | Y |
| pybind11 | N | N | N | N | N |
| reader-writerqueue | Y | Y | Y | Y | Y |
| span-lite | Y | Y | Y | Y | Y |
| spdlog | Y | Y | Y | Y | Y |
| tomlplus-plus | Y | N | N | N | N |
| xz_utils | Y1 | N | N | N | N |
| zstd | Y | Y | Y | Y | Y |
| zlib | Y1 | N | N | N | N |

[1] required depending on how libarchive was compiled
[2] required to compile with telemetry enabled
[3] only `Boost::headers` is required
[4] required if the corresponding functionality should be tested

## 3.2 Running automated tests

The steps outlined in this section are optional but highly recommended.

### 3.2.1 Unit tests

```
# Activate venv
. /tmp/venv/bin/activate

cd /tmp/hictk
```

(continues on next page)

(continued from previous page)

```
ctest --test-dir build/   \
      --schedule-random   \
      --output-on-failure \
      --no-tests=error    \
      --timeout 120       \
      -j8  # Change this to the number of available CPU cores
```

A successful run of the test suite will produce an output like the following:

```
user@dev:/tmp/hictk$ ctest --test-dir build/ ...
...
 96/106 Test  #62: Cooler: dataset linear iteration - LONG ..........................
↪    Passed    2.26 sec
 97/106 Test #104: Transformers (hic) - SHORT .......................................
↪    Passed    2.81 sec
 98/106 Test   #7: Balancing: SCALE (gw) - SHORT ....................................
↪    Passed    2.49 sec
 99/106 Test  #17: Balancing: SCALE (edge cases) - MEDIUM ..........................
↪    Passed    2.78 sec
100/106 Test  #15: Balancing: ICE (inter) - MEDIUM .................................
↪    Passed    3.17 sec
101/106 Test   #6: Balancing: SCALE (inter) - SHORT ................................
↪    Passed    3.06 sec
102/106 Test   #8: Balancing: AtomicBitSet - SHORT .................................
↪    Passed    3.52 sec
103/106 Test  #66: Cooler: utils merge - LONG ......................................
↪    Passed    3.88 sec
104/106 Test  #61: Cooler: dataset random iteration - MEDIUM .......................
↪    Passed   10.41 sec
105/106 Test  #63: Cooler: dataset large read/write - LONG .........................
↪    Passed   12.10 sec
106/106 Test  #92: HiC: HiCFileWriter - LONG .......................................
↪    Passed   13.03 sec
100% tests passed, 0 tests failed out of 106

Total Test time (real) = 101.97 sec
```

**All tests are expected to pass. Do not ignore test failures!**

### Troubleshooting test failures

If one or more tests fail, try the following troubleshooting steps before reaching out for help.

1. Ensure you are running `ctest` from the root of the source tree (`/tmp/hictk` if you are following the instructions).
2. Ensure you are passing the correct build folder to `--test-dir`. Pass the absolute path if necessary (i.e., `--test-dir=/tmp/hictk/build/` if you are following the instructions).
3. Re-run `ctest` with `-j1`. This can be necessary on machines with very little memory (e.g., less than 2GB).
4. Before running `ctest`, create a temporary folder where your user has read-write permissions and where there are at least 100-200MB of space available. Then set variable `TMPDIR` to that folder and re-run `ctest`.
5. Checksum the test dataset located under `test/data/` by running `sha256sum -c checksums.sha256`. If the checksumming fails or the folder doesn't exist, download and extract the `.tar.zst` file listed in file `cmake/FetchTestDataset.cmake`. Make sure you run `tar -xf` from the root of the repository (`/tmp/hictk` if you are following the instructions).

Example:

```
# Activate venv
. /tmp/venv/bin/activate

cd /tmp/hictk

# Make sure this is the URL listed in file cmake/FetchTestDataset.cmake
curl -L 'https://zenodo.org/records/13849053/files/hictk_test_data.tar.zst?download=1
↪' | zstdcat | tar -xf -

# This should print "OK" if the check is successful
(cd test/data && sha256sum --quiet -c checksums.sha256 && 2>&1 echo OK)

mkdir ~/hictk-test-dir  # Remember to delete this folder

TMPDIR="$HOME/hictk-test-dir"      \
ctest --test-dir=/tmp/hictk/build/ \
      --schedule-random            \
      --output-on-failure          \
      --no-tests=error             \
      --timeout 600                \
      -j1

# rm -r ~/hictk-test-dir
```

If after trying the above steps the tests are still failing, please feel free to start a discussion asking for help.

### 3.2.2 Integration tests

The integration test suite is implemented in Python, requires 3.11 or newer, and can be installed using pip:

```
# Activate venv
. /tmp/venv/bin/activate

pip install test/integration

hictk_integration_suite --help
```

Once installed, the full integration suite can be run as follows:

```
# Activate venv
. /tmp/venv/bin/activate

cd /tmp/hictk

hictk_integration_suite \
   build/src/hictk/hictk \
   test/integration/config.toml \
   --data-dir test/data \
   --threads 8 \
   --result-file results.json

# To run specific parts of the integration suite, pass e.g. --suites=metadata,validate
```

## 3.3 Installation

Once all tests have passed, `hictk` can be installed as follows:

```
# Activate venv
user@dev:/tmp$ . /tmp/venv/bin/activate

# Install system-wide (requires root/admin rights)
user@dev:/tmp$ cmake --install /tmp/hictk/build
-- Install configuration: "Release"
-- Installing: /usr/local/bin/hictk
-- Set non-toolchain portion of runtime path of "/usr/local/bin/hictk" to ""
-- Installing: /usr/local/share/licenses/hictk/LICENSE
-- Installing: /usr/local/include/hictk
...

# Alternatively, install to custom path
user@dev:/tmp$ cmake --install /tmp/hictk/build --prefix "$HOME/.local/"
-- Install configuration: "Release"
-- Installing: /home/user/.local/bin/hictk
-- Set non-toolchain portion of runtime path of "/home/user/.local/bin/hictk" to ""
-- Installing: /home/user/.local/share/licenses/hictk/LICENSE
-- Installing: /home/user/.local/include/hictk
...

# Install the hictk binary only (i.e. without the header files required for␣
↪development)
user@dev:/tmp$ cmake --install /tmp/hictk/build --component Runtime
-- Install configuration: "Release"
-- Installing: /usr/local/bin/hictk
-- Set non-toolchain portion of runtime path of "/usr/local/bin/hictk" to ""
-- Installing: /usr/local/share/licenses/hictk/LICENSE
```

## 3.4 Cleaning build artifacts

After successfully compiling hictk the following folders can safely be removed:

- Python virtualenv: `/tmp/venv`
- hictk source tree: `/tmp/hictk`

If you are not using Conan in any other project feel free to delete Conan's folder `~/.conan2/`.

CHAPTER

# FOUR

# FREQUENTLY ASKED QUESTIONS (FAQ)

## 4.1 .hic files created with hictk appear empty when opened in JuiceBox

The .hic files created by hictk are in the latest format revision (.hic v9).

Versions of Juicebox available on github.com/aidenlab/Juicebox are not capable of reading .hic v9 files. Instead, you should download the latest version of JuiceBox from the github.com/aidenlab/JuiceboxGUI repository (e.g., JuiceboxGUI v3.1.4).

## 4.2 I am trying to install hictk using conda and I am running into package incompatibilities. What can I do?

This is a common issue when working with Conda environments (and is not unique to hictk). Here are some options:

- Install all the dependencies at once, when creating the environment.

  **Instead of:**

  ```
  conda create -n myenv
  conda install -c conda-forge -c bioconda package1
  conda install -c conda-forge -c bioconda package2
  conda install -c conda-forge -c bioconda hictk
  ```

  **do:**

  ```
  conda create -n myenv -c conda-forge -c bioconda package1 package2 hictk
  ```

- **Use containers:**

  ```
  # Note that this may require using sudo
  docker run --rm paulsengroup/hictk --help
  ```

- Compile from source:

  Package incompatibilities encountered with Conda are due to the way applications are compiled and packaged by conda (i.e., using dynamic linking).

  In contrast, when building hictk from source, all of hictk's dependencies are statically linked into the hictk executable. This means that external dependencies are embedded into the hictk executable itself and are thus only needed while compiling hictk.

  After compilation, you are free to remove all of hictk's dependencies, make copies of the hictk binary, and even share the same binary across multiple machines (provided that system libraries such as libc and libstdc++ are ABI compatible).

  Detailed instructions on how to build hictk from source are available *here*.

## 4.3 How do I turn off telemetry?

The simplest way is to ensure you are defining the environment variable `HICTK_NO_TELEMETRY` before running hictk:

```
HICTK_NO_TELEMETRY=1 hictk dump ...
```

You can double-check whether hictk will collect telemetry with:

```
user@dev:/tmp$ HICTK_NO_TELEMETRY=1 hictk --help-telemetry

hictk was compiled WITH support for telemetry.
Telemetry data won't be collected as the environment variable "HICTK_NO_TELEMETRY" is␣
↪defined.
See https://hictk.readthedocs.io/en/latest/telemetry.html for more details.

user@dev:/tmp$ hictk --help-telemetry

hictk was compiled WITH support for telemetry.
Telemetry data will be collected as the environment variable "HICTK_NO_TELEMETRY" is␣
↪not defined.
See https://hictk.readthedocs.io/en/latest/telemetry.html for more details.
```

For more details, refer to the *Telemetry* page in the documentation.

## 4.4 When fetching expected or observed/expected interactions from .hic files I don't get interactions for every pixel. How come?

This is the intended behavior (and it is also how straw deals with data from expected matrices).

Despite the relative simplicity of the idea behind a matrix of expected genomic interactions, there is no consensus on exactly how this matrix should be calculated.

Thus, almost every tool calculates this matrix in slightly different ways.

When developing `hictk` we refrained from introducing a new way of computing expected interactions, and instead opted to mimic the behavior of straw.

## 4.5 I am getting an error like "(Virtual File Layer) Unable to lock file" when balancing Cooler files with hictk balance

Example:

```
[2025-04-11 11:51:22.095] [info]: Writing weights to /tmp/4DNFIZ1ZVXC8.mcool::/
↪resolutions/1000/bins/GW_ICE...
HDF5-DIAG: Error detected in HDF5 (1.14.5):
  #000: src/src/H5F.c line 827 in H5Fopen(): unable to synchronously open file
    major: File accessibility
    minor: Unable to open file
  #001: src/src/H5F.c line 788 in H5F__open_api_common(): unable to open file
    major: File accessibility
    minor: Unable to open file
  #002: src/src/H5VLcallback.c line 3680 in H5VL_file_open(): open failed
    major: Virtual Object Layer
    minor: Can't open object
```

(continues on next page)

(continued from previous page)

```
  #003: src/src/H5VLcallback.c line 3514 in H5VL__file_open(): open failed
    major: Virtual Object Layer
    minor: Can't open object
  #004: src/src/H5VLnative_file.c line 128 in H5VL__native_file_open(): unable to␣
↪open file
    major: File accessibility
    minor: Unable to open file
  #005: src/src/H5Fint.c line 1963 in H5F_open(): unable to lock the file
    major: File accessibility
    minor: Unable to lock file
  #006: src/src/H5FD.c line 2402 in H5FD_lock(): driver lock request failed
    major: Virtual File Layer
    minor: Unable to lock file
  #007: src/src/H5FDsec2.c line 956 in H5FD__sec2_lock(): unable to lock file, errno␣
↪= 11, error message = 'Resource temporarily unavailable'
    major: Virtual File Layer
    minor: Unable to lock file
[2025-04-11 11:51:22.095] [critical]: FAILURE! hictk balance encountered the␣
↪following error: Unable to open file /tmp/4DNFIZ1ZVXC8.mcool (Virtual File Layer)␣
↪Unable to lock file
```

After computing the balancing weights, `hictk balance` needs to write the weight vectors to the given Cooler file.

This requires that:

- You have write permissions on that file
- The file is not opened in any other process (e.g., Higlass, cooler, hictk, a Jupyter notebook etc.)

If you can't figure out which process is keeping the file open, you can make a copy of the file and run `hictk balance` that copy.

## 4.6 How should I cite hictk?

Thanks for taking the time to check how to properly cite hictk!

- DOI: doi.org/10.1093/bioinformatics/btae408
- Plain text:

```
Roberto Rossini, Jonas Paulsen, hictk: blazing fast toolkit to work with .hic␣
↪and .cool files Bioinformatics,
Volume 40, Issue 7, July 2024, btae408, https://doi.org/10.1093/bioinformatics/
↪btae408
```

- Bibtex:

```
@article{hictk,
    author = {Rossini, Roberto and Paulsen, Jonas},
    title = "{hictk: blazing fast toolkit to work with .hic and .cool files}",
    journal = {Bioinformatics},
    volume = {40},
    number = {7},
    pages = {btae408},
    year = {2024},
    month = {06},
    issn = {1367-4811},
    doi = {10.1093/bioinformatics/btae408},
    url = {https://doi.org/10.1093/bioinformatics/btae408},
```

(continues on next page)

(continued from previous page)

```
    eprint = {https://academic.oup.com/bioinformatics/article-pdf/40/7/btae408/
↪58385157/btae408.pdf},
}
```

CHAPTER

FIVE

# QUICKSTART (CLI)

First, install hictk with one of the methods listed in the *Installation* section.

Next, verify that hictk was installed correctly with:

```
user@dev:/tmp$ hictk --version
hictk-v2.2.0
```

## 5.1 Command line interface

hictk CLI supports performing common operations on .hic and .cool files directly from the shell.

### 5.1.1 Verifying file integrity

```
hictk validate interactions.cool --validate-index

hictk validate interactions.hic
```

For more detailed examples refer to *File validation*

### 5.1.2 Reading interactions

hictk supports reading interactions from .hic and .cool files through the `hictk dump` command:

```
user@dev:/tmp$ hictk dump interactions.cool
0     0       1745
0     1       2844
0     2       409
...

user@dev:/tmp$ hictk dump interactions.cool --join
chr2L 0       10000   chr2L   0       10000   1745
chr2L 0       10000   chr2L   10000   20000   2844
chr2L 0       10000   chr2L   20000   30000   409
...
```

For more detailed examples refer to *Reading interactions*

### 5.1.3 Other operations

- *Balancing Hi-C matrices*
- *Converting single-resolution files to multi-resolution*
- *Creating .cool and .hic files*
- *Dumping tabular information to stdout*

- *File validation*
- *Format conversion*
- *Reading file metadata*

## 5.2 API

Refer to *Quickstart (API)*.

CHAPTER

SIX

# QUICKSTART (API)

The C++ library component of hictk, libhictk, can be installed and configured in several ways.

If you are looking for the documentation for the Python or R API, please refer to:

- Python API: hictkpy.readthedocs.io
- R API: paulsengroup.github.io/hictkR

## 6.1 Installing libhictk

### 6.1.1 Installing using Conan

To install libhictk using Conan, first create a conanfile.txt like the following:

```
[requires]
hictk/2.2.0

[generators]
CMakeDeps

[layout]
cmake_layout
```

Next, install hictk as follows:

```
conan install conanfile.txt --build=missing --output-folder=conan_deps
```

Folder `conan_deps` will contain all CMake module and config files required to include hictk in an application using CMake as build generator.

Finally, add `find_package(hictk REQUIRED)` to your `CMakeLists.txt` and pass the full path to folder `conan_deps` to CMake through the `CMAKE_PREFIX_PATH` variable:

```
cmake -DCMAKE_PREFIX_PATH='/path/to/conan_deps' ... -B build/ -S .
```

For more options and details refer to hictk page on ConanCenter.

### 6.1.2 Installing using CMake FetchContent

Before beginning, please ensure that all of hictk's dependencies have been installed. Refer to conanfile.py for an up-to-date list of hictk dependencies.

To install and configure hictk using FetchContent, first write a `CMakeLists.txt` file like the following:

```
cmake_minimum_required(VERSION 3.25)

project(myproject LANGUAGES C CXX)
```

(continues on next page)

(continued from previous page)

```
include(FetchContent)
FetchContent_Declare(
  hictk
  GIT_REPOSITORY  "https://github.com/paulsengroup/hictk.git"
  GIT_TAG         v2.2.0
  SYSTEM)

# Customize hictk build flags
set(HICTK_ENABLE_TESTING OFF)
set(HICTK_BUILD_EXAMPLES OFF)
set(HICTK_BUILD_BENCHMARKS OFF)
set(HICTK_BUILD_TOOLS OFF)
set(HICTK_INSTALL OFF)

FetchContent_MakeAvailable(hictk)

add_executable(main main.cpp)
target_link_libraries(main PRIVATE hictk::file)  # Add other targets as necessary
```

### 6.1.3 Include hictk source using CMake add_subdirectory

Simply add a copy of hictk source code to your source tree (e.g., under folder `myproject/external/hictk`), then add `add_subdirectory("external/hictk")` to your `CMakeLists.txt`.

### 6.1.4 A quick tour of libhictk

libhictk is a C++17 header-only library that provides the building blocks required to build complex applications operating on .hic and .cool files.

libhictk public API is organized in 5 main sections:

1. Classes *cooler::File*, *cooler::MultiResFile* and *cooler::SingleCellFile*, which can be used to read and write .cool, .mcool and .scool files respectively.
2. Class *hic::File* which can be used to read .hic files
3. Class *File* which wraps *cooler::File* and *hic::File* and provides a uniform interface to read .cool and .hic files
4. Various other classes used e.g., to model tables of bins, reference genomes and much more
5. Classes and free-standing functions to perform common operations on files or pixel iterators, such as coarsening and balancing.

The quick tour showcases basic functionality of the generic *File* class. For more detailed examples refer to hictk examples and *C++ API Reference*.

```
#include <algorithm>
#include <cstdint>
#include <hictk/file.hpp>
#include <iostream>
#include <string>

int main() {
  // const std::string path = "interactions.cool";
  // const std::string path = "interactions.mcool::/resolutions/1000";
  const std::string path = "interactions.hic";
  const std::uint32_t resolution = 1000;
```

(continues on next page)

(continued from previous page)

```
  const hictk::File f(path, resolution);

  const auto selector = f.fetch("chr1", "chr2");

  std::for_each(selector.template begin<std::int32_t>(), selector.template end
↪<std::int32_t>(),
                [](const hictk::ThinPixel<std::int32_t>& p) {
                  std::cout << p.bin1_id << "\t";
                  std::cout << p.bin2_id << "\t";
                  std::cout << p.count << "\n";
                });
}
```

It is often the case that we need access to more information than just bin IDs and counts. Joining genomic coordinates to pixel counts can be done as follows:

```
#include <cstdint>
#include <hictk/file.hpp>
#include <hictk/transformers.hpp>
#include <iostream>
#include <string>

int main() {
  const std::string path = "interactions.hic";
  const std::uint32_t resolution = 1000;

  const hictk::File f(path, resolution);

  const auto selector = f.fetch("chr1", "chr2");
  const hictk::transformers::JoinGenomicCoords jselector(
      selector.template begin<std::int32_t>(), selector.template end<std::int32_t>(),␣
↪f.bins_ptr());

  for (const auto& p : jselector) {
    std::cout << p.coords.bin1.chrom().name() << "\t";
    std::cout << p.coords.bin1.start() << "\t";
    std::cout << p.coords.bin1.end() << "\t";
    std::cout << p.coords.bin2.chrom().name() << "\t";
    std::cout << p.coords.bin2.start() << "\t";
    std::cout << p.coords.bin2.end() << "\t";
    std::cout << p.count << "\n";
  }
}
```

The above examples work just fine. However, using iterators returned by generic *PixelSelector* is suboptimal. These iterators are implemented using std::variant and require checking the type of the underlying `PixelSelector` every iteration. The overhead of this check is quite low but still noticeable.

We can avoid paying this overhead by using the format-specific file handles instead of the generic one, or by using std::visit as follows:

```
#include <algorithm>
#include <cstdint>
#include <hictk/file.hpp>
#include <iostream>
#include <string>
#include <variant>
```

(continues on next page)

(continued from previous page)

```cpp
int main() {
  const std::string path = "interactions.hic";
  const std::uint32_t resolution = 1000;

  const hictk::File f(path, resolution);

  const auto selector = f.fetch("chr1", "chr2");

  // std::visit applies the lambda function provided as first argument
  // to the variant returned by selector.get().
  // In this way, the type held by the std::variant is checked once
  // and the underlying PixelSelector and iterators are used for all operations
  std::visit(
      [&](const auto& sel) {
        std::for_each(sel.template begin<std::int32_t>(), sel.template end<std::int32_
↪t>(),
                      [](const hictk::ThinPixel<std::int32_t>& p) {
                        std::cout << p.bin1_id << "\t";
                        std::cout << p.bin2_id << "\t";
                        std::cout << p.count << "\n";
                      });
      },
      selector.get());
}
```

CHAPTER

SEVEN

# DOWNLOADING TEST DATASETS

Test datasets for `hictk` are hosted on Zenodo: doi.org/10.5281/zenodo.8121686

After downloading the data, move to a folder with at least ~1 GB of free space and extract the test datasets:

```
user@dev:/tmp$ mkdir data/
user@dev:/tmp$ tar -xf hictk_test_data.tar.zst          \
                   -C data --strip-components=3         \
                   test/data/hic/4DNFIZ1ZVXC8.hic9      \
                   test/data/cooler/4DNFIZ1ZVXC8.mcool \
                   test/data/interactions/4DNFIKNWM36K.subset.pairs.xz

user@dev:/tmp$ ls -lah data
total 261M
drwx------  2 dev dev   80 Sep 29 17:00 .
drwxrwxrwt 26 dev dev  960 Sep 29 17:00 ..
-rw-------  1 dev dev 125M Jun 26  2023 4DNFIKNWM36K.subset.pairs.xz
-rw-------  1 dev dev 128M Jun  8  2023 4DNFIZ1ZVXC8.hic9
-rw-------  1 dev dev 133M Jul  7  2023 4DNFIZ1ZVXC8.mcool
```

CHAPTER

EIGHT

# FILE VALIDATION

## 8.1 Why is this needed?

`hictk validate` can detect several types of data corruption in .hic and .[ms]cool files, from simple file truncation due to e.g., failed downloads to subtle index corruption in .mcool files.

### 8.1.1 Cooler index corruption

In essence, older versions of cooler (including v0.8.3) had a bug in `cooler zoomify` that caused the generation of invalid file indexes. This results in duplicate pixels with different values being reported for the affected region.

Example:

Table 1: Output of cooler dump for corrupted file 4DNFI9GMP2J8.mcool

| chrom1 | start1 | end1 | chrom2 | start2 | end2 | count | balanced |
|---|---|---|---|---|---|---|---|
| chr1 | 10828000 | 10830000 | chr1 | 11002000 | 11004000 | 1 | 0.000208987 |
| chr1 | 10828000 | 10830000 | chr1 | 11002000 | 11004000 | 1 | 0.000208987 |
| chr1 | 10828000 | 10830000 | chr1 | 11006000 | 11008000 | 1 | 0.000199523 |
| chr1 | 10828000 | 10830000 | chr1 | 11006000 | 11008000 | 3 | 0.000598569 |
| chr1 | 10828000 | 10830000 | chr1 | 11010000 | 11012000 | 4 | 0.000695946 |
| chr1 | 10828000 | 10830000 | chr1 | 11010000 | 11012000 | 2 | 0.000347973 |
| chr1 | 10828000 | 10830000 | chr1 | 11020000 | 11022000 | 1 | 0.000219669 |
| chr1 | 10828000 | 10830000 | chr1 | 11020000 | 11022000 | 1 | 0.000219669 |
| chr1 | 10828000 | 10830000 | chr1 | 11030000 | 11032000 | 3 | 0.000499071 |
| chr1 | 10828000 | 10830000 | chr1 | 11030000 | 11032000 | 2 | 0.000332714 |
| … | … | … | … | … | … | … | … |

Unfortunately, this is not a rare issue, as the above bug currently affects most .mcool files released by 4D Nucleome:

This is an output file of the Hi-C processing pipeline

A February 25th, 2019 at 9:17pm

Note(s)

WARNING - Due to a bug in the version of cooler (0.8.3) used in the current 4DN standard Hi-C processing pipeline some pixels may occur mulitple times at a single resolution with different counts being reported for each occurence. This duplication does not affect the higlass display of these files, howevver, downstream analyses using this file may encounter issues due to this pixel duplication. The counts from the duplicate pixels can be aggregated to determine the correct count value at that location. If this issue is problematic for your needs you should consider regenerating the matrices from the merged pairs file of the associated dataset using a more recent version of cooler. We are working to update the pipeline but do not yet have a predicted date for when this issue will be resolved.

Properties

| File Format | File Type |
|---|---|
| mcool | Contact Matrix |
| General Classification | File Size |
| Processed File | 12.19 GB |

## 8.2 hictk validate

`hictk validate` was initially developed to detect files affected by the above issue and was later extended to also validate .cool, .scool, and .hic files as well.

Perform a quick check to detect truncated or otherwise invalid files:

```
# Validate a .hic file
user@dev:/tmp$ hictk validate data/4DNFIZ1ZVXC8.hic9
[2025-03-22 10:48:34.598] [info]: Running hictk v2.0.2-6a99e2c3
{
    "format": "hic",
    "is_valid_hic": true,
    "uri": "data/4DNFIZ1ZVXC8.hic9"
}
### SUCCESS: "data/4DNFIZ1ZVXC8.hic9" is a valid .hic file.

# Validate a .mcool file
user@dev:/tmp$ hictk validate data/4DNFIZ1ZVXC8.mcool
[2025-03-22 10:49:31.404] [info]: Running hictk v2.0.2-6a99e2c3-dirty
{
    "1000": {
        "bin_table_dtypes_ok": true,
        "bin_table_num_invalid_bins": 0,
        "bin_table_shape_ok": true,
        "file_was_properly_closed": true,
        "index_is_valid": "not_checked",
        "is_hdf5": true,
        "is_valid_cooler": true,
        "missing_groups": [],
        "missing_or_invalid_bin_type_attr": false,
        "missing_or_invalid_format_attr": false,
        "pixels_are_sorted": "not_checked",
        "pixels_are_symmetric_upper": "not_checked",
        "pixels_are_unique": "not_checked",
        "pixels_have_valid_counts": "not_checked",
        "unable_to_open_file": false
    },
    "100000": {
        ...
```

(continues on next page)

(continued from previous page)

```
    },
    "1000000": {
        ...
    },
    "25000": {
        ...
    },
    "250000": {
        ...
    },
    "2500000": {
        ...
    },
    "5000": {
        ...
    },
    "50000": {
        ...
    },
    "500000": {
        ...
    },
    "file_was_properly_closed": true,
    "format": "mcool",
    "is_hdf5": true,
    "is_valid_mcool": true,
    "missing_groups": [],
    "missing_or_invalid_bin_type_attr": false,
    "missing_or_invalid_format_attr": false,
    "unable_to_open_file": false,
    "uri": "data/4DNFIZ1ZVXC8.mcool"
}
### SUCCESS: "data/4DNFIZ1ZVXC8.mcool" is a valid .mcool file.
```

The quick check will not detect Cooler files with corrupted index, as this requires the `--validate-index` option (note, this step requires a corrupted .mcool file such as 4DNFI9GMP2J8.mcool):

```
user@dev:/tmp$ hictk validate --validate-index 4DNFI9GMP2J8.mcool::/resolutions/
↪1000000
[2024-09-26 16:26:32.671] [info]: Running hictk v1.0.0-fbdcb591
{
    "bin_table_dtypes_ok": true,
    "bin_table_num_invalid_bins": 0,
    "bin_table_shape_ok": true,
    "file_was_properly_closed": true,
    "format": "cool",
    "index_is_valid": "pixels between 0-2850 are not sorted in ascending order (and␣
↪very likely contain duplicate entries)",
    "is_hdf5": true,
    "is_valid_cooler": false,
    "missing_groups": [],
    "missing_or_invalid_bin_type_attr": false,
    "missing_or_invalid_format_attr": false,
    "pixels_are_symmetric_upper": "not_checked",
    "pixels_are_unique": "not_checked",
    "pixels_have_valid_counts": "not_checked",
```

(continues on next page)

(continued from previous page)

```
    "unable_to_open_file": false,
    "uri": "4DNFI9GMP2J8.mcool::/resolutions/100000"
}
### FAILURE: "4DNFI9GMP2J8.mcool::/resolutions/100000" does not point to valid Cooler.
```

In addition, when validating .[ms]cool files, the `--validate-pixels` flag can be used to detect malformed or invalid pixels such as:

- Unsorted pixels (this is usually a consequence of the file index corruption outlined above).
- File has `storage-mode="symmetric-upper"` but pixels overlap with the lower-triangular matrix.
- File contains duplicate pixels (note that this only checks consecutive values. If duplicate pixels are present but are not consecutive, they will be detected by the first check).
- Pixels have invalid count values (e.g., pixels have 0 interactions).

When launched with default settings, hictk validate outputs its report in .json format. The output format can be changed using the `--output-format` option. Output to stdout can be completely suppressed by providing the `--quiet` option (the outcome of file validation can still be determined based on hictk's exit code). When processing multi-resolution or single-cell files, hictk validate returns as soon as the first validation failure is encountered. This behavior can be changed by specifying the `--exhaustive` flag.

## 8.3 Restoring corrupted .mcool files

Luckily, the base resolution of .mcool files corrupted as described in *Cooler index corruption* is still valid, and so corrupted resolutions can be regenerated from the base resolution.

File restoration is automated with `hictk fix-mcool`:

```
hictk fix-mcool 4DNFI9GMP2J8.mcool 4DNFI9GMP2J8.fixed.mcool
```

`hictk fix-mcool` is basically a wrapper around `hictk zoomify` and `hictk balance`.

When balancing, `hictk fix-mcool` will try to use the same parameters used to balance the original .mcool file. When this is not possible, `hictk fix-mcool` will fall back to the default parameters used by `hictk balance`.

To improve performance, consider using the `--in-memory` and/or `--threads` CLI options when appropriate (see *Balancing Hi-C matrices* for more details).

CHAPTER

NINE

# FILE METADATA

`hictk metadata` can be used to read metadata from any Cooler file (that is, .cool, .mcool, and .scool files) as well as .hic files.

## 9.1 Fetching metadata from single-resolution files

The following shows how to use `hictk metadata` to access metadata information stored in single-resolution Cooler files (or URIs, as shown in the example below):

```
user@dev:/tmp$ hictk metadata data/4DNFIZ1ZVXC8.mcool::/resolutions/1000

{
    "assembly": "dm6",
    "bin-size": 1000,
    "bin-type": "fixed",
    "creation-date": "2023-07-07T14:15:30.759988",
    "format": "HDF5::Cooler",
    "format-url": "https://github.com/4dn-dcic/hic2cool",
    "format-version": 3,
    "generated-by": "hic2cool-0.8.3",
    "nbins": 137572,
    "nchroms": 8,
    "nnz": 26591454,
    "storage-mode": "symmetric-upper"
}
```

By default, `hictk metadata` outputs information in JSON format. However, the output format can be changed using the `--output-format` CLI options (currently, `json`, `toml`, and `yaml` formats are supported).

## 9.2 Fetching metadata from multi-resolution files

Next, we will look at how to fetch metadata from multi-resolution .hic and .mcool files.

```
user@dev:/tmp$ hictk metadata data/4DNFIZ1ZVXC8.hic9

{
    "assembly": "dm6",
    "format": "HIC",
    "format-url": "https://github.com/aidenlab/hic-format",
    "format-version": 9,
    "hicFileScalingFactor": 1.0,
    "nchroms": 8,
    "resolutions": [
        1000,
```

(continues on next page)

(continued from previous page)

```
        5000,
        10000,
        25000,
        50000,
        100000,
        250000,
        500000,
        1000000,
        2500000
    ],
    "software": "Juicer Tools Version 3.30.00"
}
```

When dealing with multi-resolution and single-cell files, it is possible to also view metadata information of individual resolutions/cells by using the `--recursive` CLI flag:

```
user@dev:/tmp$ hictk metadata data/4DNFIZ1ZVXC8.mcool --recursive

{
    "1000": {
        "assembly": "dm6",
        "bin-size": 1000,
        "bin-type": "fixed",
        "creation-date": "2023-07-07T14:15:30.759988",
        "format": "HDF5::Cooler",
        "format-url": "https://github.com/4dn-dcic/hic2cool",
        "format-version": 3,
        "generated-by": "hic2cool-0.8.3",
        "nbins": 137572,
        "nchroms": 8,
        "nnz": 26591454,
        "storage-mode": "symmetric-upper"
    },
    ...
    "bin-type": "fixed",
    "format": "HDF5::MCOOL",
    "format-version": 2,
    "resolutions": [
        1000,
        5000,
        10000,
        25000,
        50000,
        100000,
        250000,
        500000,
        1000000,
        2500000
    ]
}
```

CHAPTER

# TEN

# FORMAT CONVERSION

`hictk` supports conversion between .hic and .[m]cool file formats (including .hic v9 files).

## 10.1 Converting from .hic to .[m]cool

Converting from .hic to .cool or .mcool formats involves the following operations

1. Fetch the list of available resolutions
2. For each resolution to be converted:
    a. Copy all raw interactions present in the .hic file
    b. Copy all normalization vectors

Interactions are copied using streams of data, so memory requirements remain quite modest even when converting very high resolutions.

```
user@dev:/tmp$ hictk convert data/4DNFIZ1ZVXC8.hic9 4DNFIZ1ZVXC8.mcool

[2024-09-26 16:06:41.713] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-26 16:06:41.713] [info]: Converting data/4DNFIZ1ZVXC8.hic9 to 4DNFIZ1ZVXC8.
↪mcool (hic -> mcool)...
[2024-09-26 16:06:41.943] [info]: [1000] begin processing 1000bp matrix...
[2024-09-26 16:06:44.117] [info]: [1000] processing chr2R:11267000-11268000 at␣
↪4604052 pixels/s (cache hit rate 0.00%)...
[2024-09-26 16:06:46.026] [info]: [1000] processing chr3R:5812000-5813000 at 5238345␣
↪pixels/s (cache hit rate 0.10%)...
[2024-09-26 16:06:47.842] [info]: [1000] processing SCALE normalization vector...
[2024-09-26 16:06:47.873] [info]: [1000] processing VC normalization vector...
[2024-09-26 16:06:47.907] [info]: [1000] processing VC_SQRT normalization vector...
[2024-09-26 16:06:48.411] [info]: [1000] DONE! Processed 26682908 pixels across 8␣
↪chromosomes in 6.47s
...
[2024-09-26 16:06:58.265] [info]: DONE! Processed 10 resolution(s) in 16.55s!
[2024-09-26 16:06:58.265] [info]: data/4DNFIZ1ZVXC8.hic9 size: 133.68 MB
[2024-09-26 16:06:58.265] [info]: 4DNFIZ1ZVXC8.mcool size: 99.86 MB
```

It is also possible to convert only a subset of available resolutions by specifying resolutions to be converted with the `--resolutions` option.

When specifying a single resolution, the resulting file will be in .cool format.

```
user@dev:/tmp$ hictk convert data/4DNFIZ1ZVXC8.hic9 4DNFIZ1ZVXC8.1000.cool --
↪resolutions 1kbp

[2024-09-26 16:08:09.827] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-26 16:08:09.827] [info]: Converting data/4DNFIZ1ZVXC8.hic9 to 4DNFIZ1ZVXC8.
```

(continues on next page)

(continued from previous page)

```
↪cool (hic -> cool)...
[2024-09-26 16:08:10.043] [info]: [1000] begin processing 1000bp matrix...
[2024-09-26 16:08:11.216] [info]: [1000] processing chr2R:11267000-11268000 at␣
↪8539710 pixels/s (cache hit rate 93.05%)...
[2024-09-26 16:08:12.462] [info]: [1000] processing chr3R:5812000-5813000 at 8032129␣
↪pixels/s (cache hit rate 93.11%)...
[2024-09-26 16:08:13.423] [info]: [1000] processing SCALE normalization vector...
[2024-09-26 16:08:13.453] [info]: [1000] processing VC normalization vector...
[2024-09-26 16:08:13.485] [info]: [1000] processing VC_SQRT normalization vector...
[2024-09-26 16:08:13.968] [info]: [1000] DONE! Processed 26682908 pixels across 8␣
↪chromosomes in 3.92s
[2024-09-26 16:08:13.968] [info]: DONE! Processed 1 resolution(s) in 4.14s!
[2024-09-26 16:08:13.968] [info]: data/4DNFIZ1ZVXC8.hic9 size: 133.68 MB
[2024-09-26 16:08:13.968] [info]: 4DNFIZ1ZVXC8.cool size: 36.69 MB
```

## 10.2 Converting from .[m]cool to .hic

`hictk convert` can also be used to convert .[m]cool files to .hic format.

The conversion steps are similar to those carried out to convert .hic to .[m]cool. The main difference is that in this case hictk computes the raw and normalized expected values for each resolution.

```
user@dev:/tmp$ hictk convert data/4DNFIZ1ZVXC8.mcool 4DNFIZ1ZVXC8.hic

[2024-09-26 16:10:58.066] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-26 16:10:58.066] [info]: Converting data/4DNFIZ1ZVXC8.mcool to 4DNFIZ1ZVXC8.
↪hic (mcool -> hic)...
[2024-09-26 16:11:02.124] [info]: ingesting pixels at 2472799 pixels/s...
[2024-09-26 16:11:06.328] [info]: ingesting pixels at 2379253 pixels/s...
[2024-09-26 16:11:13.161] [info]: ingesting pixels at 2479544 pixels/s...
[2024-09-26 16:11:17.436] [info]: ingesting pixels at 2339729 pixels/s...
[2024-09-26 16:11:24.176] [info]: ingesting pixels at 2472188 pixels/s...
[2024-09-26 16:11:32.941] [info]: writing header at offset 0
[2024-09-26 16:11:32.941] [info]: begin writing interaction blocks to file
↪"4DNFIZ1ZVXC8.hic"...
[2024-09-26 16:11:32.941] [info]: [1000 bp] writing pixels for chr2L:chr2L matrix at␣
↪offset 249...
[2024-09-26 16:11:35.129] [info]: [1000 bp] written 2676654 pixels for chr2L:chr2L␣
↪matrix
[2024-09-26 16:11:35.159] [info]: [5000 bp] writing pixels for chr2L:chr2L matrix at␣
↪offset 4075891...
[2024-09-26 16:11:37.035] [info]: [5000 bp] written 2676654 pixels for chr2L:chr2L␣
↪matrix
[2024-09-26 16:11:37.096] [info]: [10000 bp] writing pixels for chr2L:chr2L matrix at␣
↪offset 8697885...
[2024-09-26 16:11:38.094] [info]: [10000 bp] written 1433133 pixels for chr2L:chr2L␣
↪matrix
...
[2024-09-26 16:13:20.981] [info]: [2500000 bp] initializing expected value vector
[2024-09-26 16:13:20.981] [info]: [2500000 bp] computing expected vector density
[2024-09-26 16:13:20.982] [info]: [500000 bp] computing expected vector density
[2024-09-26 16:13:20.982] [info]: writing 50 normalized expected value vectors at␣
↪offset 135622984...
[2024-09-26 16:13:20.983] [info]: writing 400 normalization vectors at offset␣
↪136510590...
[2024-09-26 16:13:21.027] [info]: DONE! Processed 10 resolution(s) in 142.96s!
```

(continues on next page)

(continued from previous page)

```
[2024-09-26 16:13:21.027] [info]: data/4DNFIZ1ZVXC8.mcool size: 139.37 MB
[2024-09-26 16:13:21.027] [info]: 4DNFIZ1ZVXC8.hic size: 140.32 MB
```

**Tips:**

- When converting large .[m]cool files to .hic, `hictk` may need to create large temporary files. When this is the case, use option `--tmpdir` to set the temporary folder to a path with sufficient space.
- When converting .[m]cool files to .hic certain conversion steps can be performed in parallel. To improve performance, ensure you increase the number of processing threads with option `--threads`.

CHAPTER

# ELEVEN

# READING INTERACTIONS

`hictk` supports reading interactions from .hic and .cool files using the `hictk dump` command.

By default, interactions are dumped to stdout in COO format (row, column, count):

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1kbp

7   7   1745
7   12  1766
7   17  1078
...
```

Use the option `--join` to instead dump interactions in bedgraph2 format:

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.mcool --resolution 1kbp --join | head -n 3

chr2L 7000    8000    chr2L   7000    8000    1745
chr2L 7000    8000    chr2L   12000   13000   1766
chr2L 7000    8000    chr2L   17000   18000   1078
```

All operations work on .hic as well as .[m]cool files:

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.mcool --resolution 1kbp | head -n 3

7   7   1745
7   12  1766
7   17  1078

user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.mcool::/resolutions/1000 | head -n 3

7   7   1745
7   12  1766
7   17  1078
```

Dump balanced or expected interactions:

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1kbp --join --balance␣
↪SCALE | head -n 3

chr2L 7000    8000    chr2L   7000    8000    1681.679565429688
chr2L 7000    8000    chr2L   12000   13000   1386.554565429688
chr2L 7000    8000    chr2L   17000   18000   878.9703979492188

user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1kbp --join --matrix-
↪type expected | head -n 3

chr2L 7000    8000    chr2L   7000    8000    88.33206176757812
```

(continues on next page)

(continued from previous page)

```
chr2L 7000    8000    chr2L   12000   13000   63.43805313110352
chr2L 7000    8000    chr2L   17000   18000   31.78345680236816
```

Dump interactions overlapping a region of interest:

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1kbp --join --range␣
↪chr3L:20,000,000-25,000,000 | head -n 3

chr3L 20002000        20003000        chr3L   20002000        20003000        2390
chr3L 20002000        20003000        chr3L   20007000        20008000        1285
chr3L 20002000        20003000        chr3L   20012000        20013000        490

user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1kbp --join --range␣
↪chr3L:20,000,000-25,000,000 --range2 chrX | head -n 3

chr3L 20002000        20003000        chrX    52000     53000   1
chr3L 20002000        20003000        chrX    157000  158000  1
chr3L 20002000        20003000        chrX    352000  353000  1
```

Dump tables other than pixels:

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --table chroms | head -n 3

chr2L 23513712
chr2R 25286936
chr3L 28110227

user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --table normalizations

SCALE
VC
VC_SQRT

user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --table resolutions | head -n 3

1000
5000
10000
```

See hictk dump help message for the complete list of supported tables.

Dump cis or trans interactions only:

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1kbp --cis-only --join␣
↪| head -n 3

chr2L 7000    8000    chr2L   7000    8000    1745
chr2L 7000    8000    chr2L   12000   13000   1766
chr2L 7000    8000    chr2L   17000   18000   1078

user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1kbp --trans-only --
↪join | head -n 3

chr2L 7000    8000    chr2R   27000   28000   1
chr2L 7000    8000    chr2R   322000  323000  1
chr2L 7000    8000    chr2R   397000  398000  1
```

CHAPTER

# TWELVE

# CREATING .COOL AND .HIC FILES

hictk supports creating .cool and .hic files from text files in the following formats:

- pairs (4DN-DCIC)
- validPairs (nf-core/hic)
- bedGraph2 (BG2)
- COO

The section below show some examples of what files in the above formats look like

**Tip**

**4DN-DCIC pairs**

```
## pairs format v1.0.0
#sorted: chr1-chr2-pos1-pos2
#shape: upper triangle
#genome_assembly: unknown
#chromsize: chr2L 23513712
#chromsize: chr2R 25286936
#chromsize: chr3L 28110227
#chromsize: chr3R 32079331
#chromsize: chr4 1348131
#chromsize: chrX 23542271
#chromsize: chrY 3667352
#columns: readID chrom1 pos1 chrom2 pos2 strand1 strand2 pair_type mapq1 mapq2
NS500537:79:HFYYWBGX2:3:13607:23885:10200        chr2L        5094        chr2L        13096
NS500537:79:HFYYWBGX2:4:11606:23060:8702        chr2L        5102        chr2L        7512        +
NB500947:283:HHKN3BGX2:4:12603:11442:19845        chr2L        5142        chr2L        5424
↪        UU        56        60
NS500537:79:HFYYWBGX2:3:21602:25370:1188        chr2L        5158        chr2L        5579343
NB500947:283:HHKN3BGX2:1:12310:9917:10061        chr2L        5161        chr2L        13625
↪        -        UU        54        60
NB500947:283:HHKN3BGX2:4:11510:19146:13622        chr2L        5164        chr2L        24518
↪        -        UU        60        60
NB500947:283:HHKN3BGX2:4:21609:3786:9179        chr2L        5169        chr2L        5304        +
↪        UU        33        60
NB500947:283:HHKN3BGX2:4:21512:6352:4317        chr2L        5181        chr2L        5836        -
↪        +        UU        27        60
```

**validPairs**

```
NS500537:79:HFYYWBGX2:3:13607:23885:10200        chr2L        5094        +        chr2L        130
↪info
NS500537:79:HFYYWBGX2:4:11606:23060:8702        chr2L        5102        +        chr2L        7512
↪info
NB500947:283:HHKN3BGX2:4:12603:11442:19845        chr2L        5142        +        chr2L        54
↪        1        frag1        frag2        1        1        allele-info
NS500537:79:HFYYWBGX2:3:21602:25370:1188        chr2L        5158        +        chr2L        5579
↪info
NB500947:283:HHKN3BGX2:1:12310:9917:10061        chr2L        5161        -
↪        chr2L        13625        -
↪        1        frag1        frag2        1        1        allele-info
NB500947:283:HHKN3BGX2:4:11510:19146:13622        chr2L        5164        -
↪        chr2L        24518        -
↪        1        frag1        frag2        1        1        allele-info
NB500947:283:HHKN3BGX2:4:21609:3786:9179        chr2L        5169        +        chr2L        5304
↪        1        frag1        frag2        1        1        allele-info
NB500947:283:HHKN3BGX2:4:21512:6352:4317        chr2L        5181        -
↪        chr2L        5836        +        1        frag1        frag2        1        1        al
↪info
```

**BG2**

```
chr2L        0        10000        chr2L        0        10000        155
chr2L        0        10000        chr2L        10000        20000        248
chr2L        0        10000        chr2L        20000        30000        43
chr2L        0        10000        chr2L        30000        40000        12
chr2L        0        10000        chr2L        40000        50000        20
chr2L        0        10000        chr2L        50000        60000        15
chr2L        0        10000        chr2L        60000        70000        9
chr2L        0        10000        chr2L        70000        80000        10
```

**COO**

```
0        0        155
0        1        248
0        2        43
0        3        12
0        4        20
0        5        15
0        6        9
0        7        10
```

**Important**

The following files are required to follow along with the examples below:

- `dm6.chrom.sizes` - download
- `4DNFIKNWM36K.pairs.gz` - download

## 12.1 Ingesting pairwise interactions into a 10kbp .cool file

Loading interactions in pairs (4DN-DCIC) format into a .cool/hic file is straightforward:

```
user@dev:/tmp$ hictk load --format 4dn --bin-size 10kbp 4DNFIKNWM36K.pairs.gz␣
↪4DNFIKNWM36K.10000.cool
```

(continues on next page)

(continued from previous page)

```
[2024-09-26 16:51:28.059] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-26 16:51:28.068] [info]: begin loading pairwise interactions into a .cool␣
↪file...
[2024-09-26 16:51:28.137] [info]: writing chunk #1 to intermediate file "/tmp/hictk-
↪tmp-XXXXQPdOSn/4DNFIKNWM36K.10000.cool.tmp"...
[2024-09-26 16:51:45.281] [info]: done writing chunk #1 to tmp file "/tmp/hictk-tmp-
↪XXXXQPdOSn/4DNFIKNWM36K.10000.cool.tmp".
[2024-09-26 16:51:45.281] [info]: writing chunk #2 to intermediate file "/tmp/hictk-
↪tmp-XXXXQPdOSn/4DNFIKNWM36K.10000.cool.tmp"...
[2024-09-26 16:52:04.969] [info]: done writing chunk #2 to tmp file "/tmp/hictk-tmp-
↪XXXXQPdOSn/4DNFIKNWM36K.10000.cool.tmp".
[2024-09-26 16:52:04.970] [info]: merging 2 chunks into "4DNFIKNWM36K.10000.cool"...
[2024-09-26 16:52:06.430] [info]: processing chr3L:1030000-1040000 chr3R:30240000-
↪30250000 at 6882312 pixels/s...
[2024-09-26 16:52:08.478] [info]: ingested 119208613 interactions (18122865 nnz) in␣
↪40.418916003s!
```

To ingest interactions in a .hic file, simply change the extension of the output file (or use the `--output-fmt` option).

hictk has native support for reading compressed interactions in the following formats: bzip2, lz4, lzo, gzip, xz, and zstd.

By default, the list of chromosomes is read from the file header. The reference genome used to build the .cool or .hic file can be provided explicitly using the `--chrom-sizes` option. Note that `--chrom-sizes` is a mandatory option when ingesting interactions in formats other than `--format=4dn`. In case the input file contains interactions mapping on chromosomes missing from the reference genome provided through `--chrom-sizes`, the `--drop-unknown-chroms` flag can be used to instruct hictk to ignore said interactions.

When loading interactions using `--format=pairs` or `--format=validPairs` into a .cool file, tables of variable bins are supported. To load interactions into a .cool with a variable bin size, provide the table of bins using the `--bin-table` option.

**Tips:**

- When creating large .cool/hic files, `hictk` needs to create potentially large temporary files. When this is the case, use option `--tmpdir` to set the temporary folder to a path with sufficient space.
- When loading interactions into .hic files, some of the steps can be run in parallel by increasing the number of processing threads using the `--threads` option.
- When loading pre-binned interactions into a .cool file, if the interactions are already sorted by genomic coordinates, the `--assume-sorted` option can be used to load interactions at once, without using temporary files.
- Interaction loading performance can be improved by processing interactions in larger chunks. This can be controlled using the `--chunk-size` option. In fact, when `--chunk-size` is greater than the number of interactions to be loaded, .hic and .cool files can be created without the use of temporary files.

## 12.2 Merging multiple files

Multiple .cool and .hic files using the same reference genome and resolution can be merged using `hictk merge`:

```
# Merge multiple cooler files

user@dev:/tmp$ hictk merge data/4DNFIZ1ZVXC8.mcool::/resolutions/10000 data/
↪4DNFIZ1ZVXC8.mcool::/resolutions/10000 -o 4DNFIZ1ZVXC8.merged.10000.cool

[2024-09-26 17:07:57.101] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-26 17:07:57.101] [info]: begin merging 2 files into one .cool file...
```

(continues on next page)

(continued from previous page)

```
[2024-09-26 17:07:58.978] [info]: processing chr3L:1030000-1040000 chr3R:29720000-
↪29730000 at 5571031 pixels/s...
[2024-09-26 17:08:01.224] [info]: DONE! Merging 2 files took 4.12s!
[2024-09-26 17:08:01.224] [info]: data/4DNFIZ1ZVXC8.merged.10000.cool size: 19.64 MB
```

Merging .hic files as well as a mix of .hic and .cool files is also supported (as long as all files have the same resolution and reference genome). When all input files contain data for multiple resolutions, the `--resolution` option is mandatory.

**Tips:**

Refer to the list of Tips from the previous section.

CHAPTER

THIRTEEN

# CREATING MULTI-RESOLUTION FILES (.HIC AND .MCOOL)

## 13.1 Converting .cool to .mcool

Interactions from a single-resolution Cooler file (.cool) can be used to generate a multi-resolution Cooler (.mcool) by iterative coarsening using `hictk zoomify`

```
user@dev:/tmp$ hictk zoomify data/4DNFIZ1ZVXC8.mcool::/resolutions/1000 out.mcool

[2024-09-26 17:21:21.792] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-26 17:21:21.795] [info]: coarsening cooler at data/4DNFIZ1ZVXC8.mcool::/
↪resolutions/1000 13 times (1000 -> 1000 -> 2000 -> 5000 -> 10000 -> 20000 -> 50000 -
↪> 100000 -> 200000 -> 500000 -> 1000000 -> 2000000 -> 5000000 -> 10000000)
[2024-09-26 17:21:21.795] [info]: copying 1000 resolution from data/4DNFIZ1ZVXC8.
↪mcool::/resolutions/1000
[2024-09-26 17:21:21.959] [info]: generating 2000 resolution from 1000 (2x)
[2024-09-26 17:21:22.134] [info]: [1000 -> 2000] processing chr2L:1996000-1998000 at␣
↪5747126 pixels/s...
[2024-09-26 17:21:22.355] [info]: [1000 -> 2000] processing chr2L:4932000-4934000 at␣
↪4545455 pixels/s...
[2024-09-26 17:21:22.563] [info]: [1000 -> 2000] processing chr2L:7986000-7988000 at␣
↪4830918 pixels/s...
 ...
[2024-09-26 17:21:42.886] [info]: generating 2000000 resolution from 1000000 (2x)
[2024-09-26 17:21:42.892] [info]: generating 5000000 resolution from 1000000 (5x)
[2024-09-26 17:21:42.898] [info]: generating 10000000 resolution from 5000000 (2x)
[2024-09-26 17:21:42.902] [info]: DONE! Processed 13 resolution(s) in 21.11s!

# Coarsen a single resolution
user@dev:/tmp$ hictk zoomify data/4DNFIZ1ZVXC8.mcool::/resolutions/1000 out.cool --
↪resolutions 50kbp

[2024-09-26 17:22:22.203] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-26 17:22:22.206] [info]: coarsening cooler at data/4DNFIZ1ZVXC8.mcool::/
↪resolutions/1000 2 times (1000 -> 1000 -> 50000)
[2024-09-26 17:22:22.206] [info]: copying 1000 resolution from data/4DNFIZ1ZVXC8.
↪mcool::/resolutions/1000
[2024-09-26 17:22:22.364] [info]: generating 50000 resolution from 1000 (50x)
[2024-09-26 17:22:23.165] [info]: [1000 -> 50000] processing chr2L:23000000-23050000␣
↪at 1253133 pixels/s...
[2024-09-26 17:22:23.939] [info]: [1000 -> 50000] processing chr3L:4600000-4650000 at␣
↪1293661 pixels/s...
[2024-09-26 17:22:24.878] [info]: [1000 -> 50000] processing chr3R:32050000-32079331␣
↪at 1064963 pixels/s...
[2024-09-26 17:22:25.151] [info]: DONE! Processed 2 resolution(s) in 2.95s!
```

## 13.2 Converting a single-resolution .hic to a multi-resolution .hic

Interactions from a .hic file (like the one generated by `hictk load`) can be used to generate a multi-resolution .hic file by iterative coarsening using `hictk zoomify`. hictk will copy interactions for resolutions that are available in the input file. Interactions at resolutions missing from the input file will be generated by iterative coarsening.

**Tips:**

For tips and tricks that also apply to hictk zoomify, please refer to the **Tips** section of the hictk load *documentation*.

CHAPTER

# FOURTEEN

# BALANCING HI-C MATRICES

`hictk` supports balancing .hic, .cool, and .mcool files using ICE (iterative correction and eigenvector decomposition), SCALE, and VC:

```
user@dev:/tmp$ hictk balance --help
Balance Hi-C matrices using ICE, SCALE, or VC.
Usage: hictk balance [OPTIONS] [SUBCOMMAND]

Options:
  -h,--help                   Print this help message and exit

Subcommands:
  ice                         Balance Hi-C matrices using ICE.
  scale                       Balance Hi-C matrices using SCALE.
  vc                          Balance Hi-C matrices using VC.
```

The following is an example showing how to balance a .cool file using ICE.

```
user@dev:/tmp$ hictk balance ice data/4DNFIZ1ZVXC8.mcool::/resolutions/1000

[2024-09-26 16:02:19.731] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-26 16:02:19.731] [info]: balancing using ICE (GW_ICE)
[2024-09-26 16:02:19.734] [info]: Writing interactions to temporary file /tmp/hictk-
↪tmp-XXXX1ZC9FF/4DNFIZ1ZVXC8.mcool.tmp...
[2024-09-26 16:02:22.480] [info]: Initializing bias vector...
[2024-09-26 16:02:22.482] [info]: Masking rows with fewer than 10 nnz entries...
[2024-09-26 16:02:23.392] [info]: Masking rows using mad_max=5...
[2024-09-26 16:02:23.860] [info]: Iteration 1: 36452362.243888594
[2024-09-26 16:02:24.327] [info]: Iteration 2: 21649057.88060747
[2024-09-26 16:02:24.792] [info]: Iteration 3: 7890065.688497526
...
[2024-09-26 16:03:12.285] [info]: Iteration 107: 2.0533518142916073e-05
[2024-09-26 16:03:12.752] [info]: Iteration 108: 1.601698258037195e-05
[2024-09-26 16:03:13.216] [info]: Iteration 109: 1.2493901433163442e-05
[2024-09-26 16:03:13.681] [info]: Iteration 110: 9.745791018854495e-06
[2024-09-26 16:03:13.707] [info]: Writing weights to data/4DNFIZ1ZVXC8.mcool::/
↪resolutions/1000/bins/GW_ICE...
[2024-09-26 16:03:13.708] [info]: Linking weights to data/4DNFIZ1ZVXC8.mcool::/
↪resolutions/1000/bins/weight...
```

When balancing files in .mcool or .hic formats, all resolutions are balanced.

By default, balancing coefficients are stored in the input file under the name "weight".

This can be changed by passing the desired name through the `--name` option.

`hictk` supports three balancing methods:

- Using all (genome-wide) interactions (default)

- Using trans interactions only
- Using cis interactions only

The balancing method can be changed through the `--mode` option (e.g., `--mode=gw` or `--mode=cis`).

When enough memory is available, `hictk` can be instructed to load all interactions into system memory by passing the `--in-memory` flag. This can dramatically speed up matrix balancing, at the cost of potentially much higher memory usage (approximately 1 GB of RAM for every 40M interactions).

Another way to improve performance is to increase the number of threads available for computation using the `--threads` option. It should be noted that when using a large number of threads (e.g., more than 16) without the `--in-memory` option, performance is likely limited by disk throughput. Thus, users are advised to use a large number of threads only when temporary data (`/tmp` by default on most UNIX-like systems) is stored on a fast SSD.

When the `--in-memory` option is not used, `hictk` will create a temporary file under the default temporary folder. This file stores interactions using a layout and compression that are optimized for the access pattern used by `hictk balance`. When balancing large matrices, this file can be quite large (sometimes tens of GBs). If this is the case, it may be appropriate to change the temporary folder using the `--tmpdir` option.

Finally, when balancing .hic files, only .hic v9 files and newer are supported.

CHAPTER

FIFTEEN

# REORDER CHROMOSOMES

## 15.1 TLDR

```
# Important! --bin-size should be the same resolution as matrix.cool
user@dev:/tmp hictk load <(hictk dump --join matrix.cool) \
                         output.cool \
                         --chrom-sizes=<(hictk dump --table=chroms matrix.cool | sort␣
↪-k2,2nr) \
                         --format=bg2 \
                         --bin-size=1kbp \
                         --transpose-lower-triangular-pixels
```

## 15.2 Why is this needed?

Sometimes we want to compare files using the same reference genome assembly, but with different chromosome orders (e.g. in one file chromosomes are sorted by size while in the other they are sorted by name). This can be a problem especially when trying to visually compare such files. This tutorial shows how to convert a .cool file with chromosomes sorted by name to a .cool file with chromosomes sorted by size. The same procedure can be applied to .hic files.

## 15.3 Walkthrough

For this tutorial, we will use file `4DNFIOTPSS3L.hic` as an example, which can be downloaded from here.

First, we extract the list of chromosomes from the input file:

```
user@dev:/tmp hictk dump 4DNFIOTPSS3L.hic --table=chroms | tee chrom.sizes

2L  23513712
2R  25286936
3L  28110227
3R  32079331
4   1348131
X   23542271
Y   3667352
```

Second, we re-order chromosomes:

```
user@dev:/tmp sort -k2,2nr chrom.sizes | tee chrom.sizes.sorted

3R  32079331
3L  28110227
2R  25286936
X   23542271
```

(continues on next page)

(continued from previous page)

```
2L  23513712
Y   3667352
4   1348131
```

Next, we dump pixels in bedGraph2 format (see below for how to make this step more efficient):

```
user@dev:/tmp hictk dump 4DNFIOTPSS3L.hic --join --resolution 1kbp > pixels.bg2

user@dev:/tmp head pixels.bg2

2L  5000    6000    2L      5000    6000    41
2L  5000    6000    2L      6000    7000    126
2L  5000    6000    2L      7000    8000    60
2L  5000    6000    2L      8000    9000    77
2L  5000    6000    2L      9000    10000   97
2L  5000    6000    2L      10000   11000   3
2L  5000    6000    2L      11000   12000   1
2L  5000    6000    2L      12000   13000   66
2L  5000    6000    2L      13000   14000   116
2L  5000    6000    2L      14000   15000   64
```

Finally, we load pixels into a new .hic file

```
user@dev:/tmp hictk load pixels.bg2 \
                         output.hic \
                         --chrom-sizes=chrom.sizes.sorted \
                         --transpose-lower-triangular-pixels \
                         --format=bg2 \
                         --bin-size=1kbp

[2024-09-27 19:00:40.344] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-27 19:00:40.353] [info]: begin loading pixels into a .hic file...
[2024-09-27 19:00:42.504] [info]: preprocessing chunk #1 at 4847310 pixels/s...
[2024-09-27 19:00:45.244] [info]: preprocessing chunk #2 at 3649635 pixels/s...
[2024-09-27 19:00:48.180] [info]: preprocessing chunk #3 at 3407155 pixels/s...
[2024-09-27 19:00:50.616] [info]: preprocessing chunk #4 at 4105090 pixels/s...
[2024-09-27 19:00:53.251] [info]: preprocessing chunk #5 at 3203434 pixels/s...
[2024-09-27 19:00:54.358] [info]: writing header at offset 0
[2024-09-27 19:00:54.358] [info]: begin writing interaction blocks to file "output.hic
↪"...
[2024-09-27 19:00:54.358] [info]: [1000 bp] writing pixels for 3R:3R matrix at offset␣
↪171...
[2024-09-27 19:01:01.039] [info]: [1000 bp] written 9571521 pixels for 3R:3R matrix
...
[2024-09-27 19:01:26.831] [info]: [1000 bp] initializing expected value vector
[2024-09-27 19:01:32.649] [info]: [1000 bp] computing expected vector density
[2024-09-27 19:01:32.649] [info]: writing 1 expected value vectors at offset 93720080.
↪..
[2024-09-27 19:01:32.649] [info]: writing 0 normalized expected value vectors at␣
↪offset 93848475...
[2024-09-27 19:01:32.682] [info]: ingested 114355295 interactions (48437845 nnz) in␣
↪52.337885908s!
```

Lastly, we check that chromosomes are properly sorted:

```
user@dev:/tmp hictk dump output.hic --table=chroms
```

(continues on next page)

(continued from previous page)

```
3R  32079331
3L  28110227
2R  25286936
X   23542271
2L  23513712
Y   3667352
4   1348131
```

## 15.4 Tips and tricks

There is one potential problem with the above solution, and that is the size of file `pixels.bg2` Luckily, we can completely avoid generating this file by using output redirection and process substitutions:

```
user@dev:/tmp hictk load <(hictk dump 4DNFIOTPSS3L.hic --join --resolution 1kbp) \
                         output.hic \
                         --chrom-sizes=chrom.sizes.sorted \
                         --transpose-lower-triangular-pixels \
                         --format=bg2 \
                         --bin-size=1kbp
```

Note that hictk still needs to generate some temporary file to load interactions into a new .cool or .hic file. When processing large files, it is a good idea to specify custom folder where to create temporary files through the `--tmpdir` flag:

```
user@dev:/tmp hictk load <(hictk dump 4DNFIOTPSS3L.hic --join --resolution 1kbp) \
                         output.hic \
                         --chrom-sizes=chrom.sizes.sorted \
                         --transpose-lower-triangular-pixels \
                         --format=bg2 \
                         --bin-size=1kbp \
                         --tmpdir=/var/tmp/
```

Another option you may want to consider when working with .hic files is the `--threads` option, which can significantly reduce the time required to load interactions into .hic files.

CHAPTER

SIXTEEN

# DUMP INTERACTIONS TO .COOL OR .HIC FILE

## 16.1 TLDR

```
# Important! --bin-size should be the same resolution as matrix.cool
user@dev:/tmp hictk load - \
                        output.cool \
                        --chrom-sizes=<(hictk dump --table=chroms matrix.cool) \
                        --format=bg2 \
                        --bin-size=1kbp \
                        < <(hictk dump --join
                                       --range=2L:0-10,000,000
                                       --range2=3R:0-10,000,000
                                       matrix.cool)
```

## 16.2 Why is this needed?

`hictk dump` can read interactions from .cool, .mcool, and .hic files and write them in text format to stdout. Additionally, `hictk dump` supports fetching interactions overlapping a pair of regions of interest through the `--range` and `--range2` CLI options. However, instead of writing interactions to stdout, we may want to write them to a new .cool or .hic file. This tutorial shows how this can be accomplished using `hictk dump` and `hictk load`.

## 16.3 Walkthrough

For this tutorial, we will use file `4DNFIOTPSS3L.hic` as an example, which can be downloaded from here.

First, we extract the list of chromosomes from the input file:

```
user@dev:/tmp hictk dump 4DNFIOTPSS3L.hic --table=chroms | tee chrom.sizes

2L  23513712
2R  25286936
3L  28110227
3R  32079331
4   1348131
X   23542271
Y   3667352
```

Second, we dump pixels in bedGraph2 format (see below for how to make this step more efficient):

```
user@dev:/tmp hictk dump 4DNFIOTPSS3L.hic \
                        --join \
                        --resolution=1kbp \
                        --range=2L:5,000,000-10,000,000 \
                        --range2=3R:7,500,000-10,000,000 > pixels.bg2
```

(continues on next page)

(continued from previous page)

```
user@dev:/tmp head pixels.bg2

2L  5000000 5001000 3R      7506000 7507000 1
2L  5000000 5001000 3R      7624000 7625000 1
2L  5000000 5001000 3R      7943000 7944000 1
2L  5000000 5001000 3R      8014000 8015000 1
2L  5000000 5001000 3R      8130000 8131000 1
2L  5000000 5001000 3R      8245000 8246000 1
2L  5000000 5001000 3R      8855000 8856000 1
2L  5000000 5001000 3R      9032000 9033000 1
2L  5000000 5001000 3R      9171000 9172000 1
2L  5000000 5001000 3R      9380000 9381000 1
```

Finally, we load pixels into a new .cool file

```
user@dev:/tmp hictk load pixels.bg2 \
                         output.cool \
                         --chrom-sizes=chrom.sizes \
                         --format=bg2 \
                         --bin-size=1kbp

[2024-09-27 18:54:58.532] [info]: Running hictk v1.0.0-fbdcb591
[2024-09-27 18:54:58.540] [info]: begin loading unsorted pixels into a .cool file...
[2024-09-27 18:54:58.629] [info]: writing chunk #1 to intermediate file "/tmp/hictk-
↪tmp-XXXXatmfuM/output.cool.tmp"...
[2024-09-27 18:54:58.641] [info]: done writing chunk #1 to tmp file "/tmp/hictk-tmp-
↪XXXXatmfuM/output.cool.tmp".
[2024-09-27 18:54:58.642] [info]: merging 1 chunks into "output.cool"...
[2024-09-27 18:54:58.672] [info]: ingested 26214 interactions (25085 nnz) in 0.
↪139864314s!
```

### 16.3.1 Removing empty chromosomes from the reference genome

This can be easily achieved by grepping `2L` and `3R` when generating the `chrom.sizes` file.

```
user@dev:/tmp hictk dump 4DNFIOTPSS3L.hic --table=chroms |
              grep -e '2L' -e '3R' |
              tee chrom.sizes

2L  23513712
3R  32079331
```

## 16.4 Tips and tricks

There is one potential problem with the above solution, and that is the size of file `pixels.bg2` Luckily, we can completely avoid generating this file by using output redirection and process substitutions:

```
user@dev:/tmp hictk load - \
                         output.cool \
                         --chrom-sizes=chrom.sizes \
                         --format=bg2 \
                         --bin-size=1kbp \
                         < <(hictk dump 4DNFIOTPSS3L.hic \
                                        --join \
```

(continues on next page)

(continued from previous page)

```
                                        --resolution=1kbp \
                                        --range=2L:0-10,000,000 \
                                        --range2=3R:0-10,000,000)
```

Note that hictk still needs to generate some temporary file to load interactions into a new .cool or .hic file. When processing large files, it is a good idea to specify custom folder where to create temporary files through the `--tmpdir` flag:

```
user@dev:/tmp hictk load - \
                          output.cool \
                          --chrom-sizes=chrom.sizes \
                          --format=bg2 \
                          --bin-size=1kbp \
                          --tmpdir=/var/tmp/ \
                          < <(hictk dump 4DNFIOTPSS3L.hic \
                                        --join \
                                        --resolution=1kbp \
                                        --range=2L:0-10,000,000 \
                                        --range2=3R:0-10,000,000)
```

Another option you may want to consider when working with .hic files, is the `--threads` option, which can significantly reduce the time required to load interactions into .hic files.

CHAPTER

# SEVENTEEN

# CLI REFERENCE

For an up-to-date list of subcommands and CLI options refer to `hictk --help`.

## 17.1 Subcommands

```
Blazing fast tools to work with .hic and .cool files.
hictk [OPTIONS] [SUBCOMMANDS]
OPTIONS:
  -h,     --help              Print this help message and exit
  -V,     --version           Display program version information and exit
[Option Group: help]
  [At most 1 of the following options are allowed]
OPTIONS:
          --help-cite         Print hictk's citation in Bibtex format and exit.
          --help-build-meta   Print information regarding hictk's build options and␣
↪third-party
                              dependencies, and exit.
          --help-docs         Print the URL to hictk's documentation and exit.
          --help-license      Print the hictk license and exit.
          --help-telemetry    Print information regarding telemetry collection and␣
↪exit.
SUBCOMMANDS:
  balance                     Balance Hi-C files using ICE, SCALE, or VC.
  convert                     Convert Hi-C files between different formats.
  dump                        Read interactions and other kinds of data from .hic and␣
↪Cooler
                              files and write them to stdout.
  fix-mcool                   Fix corrupted .mcool files.
  load                        Build .cool and .hic files from interactions in various␣
↪text
                              formats.
  merge                       Merge multiple Cooler or .hic files into a single file.
  metadata                    Print file metadata to stdout.
  rename-chromosomes, rename-chromsRename chromosomes found in Cooler files.
  validate                    Validate .hic and Cooler files.
  zoomify                     Convert single-resolution Cooler and .hic files to
                              multi-resolution by coarsening.
```

## 17.2 hictk balance

```
Balance Hi-C files using ICE, SCALE, or VC.
hictk balance [OPTIONS] SUBCOMMAND
OPTIONS:
```

(continues on next page)

(continued from previous page)

```
  -h,     --help              Print this help message and exit
SUBCOMMANDS:
  ice                         Balance Hi-C files using ICE.
  scale                       Balance Hi-C files using SCALE.
  vc                          Balance Hi-C matrices using VC.
```

## 17.3 hictk balance ice

```
Balance Hi-C files using ICE.
hictk balance ice [OPTIONS] input
POSITIONALS:
  input TEXT:((.[ms]cool) OR (.hic)) AND (NOT .scool) REQUIRED
                              Path to the .hic, .cool or .mcool file to be balanced.
OPTIONS:
  -h,     --help              Print this help message and exit
          --mode TEXT:{gw,trans,cis} [gw]
                              Balance matrix using:
                              - genome-wide interactions (gw)
                              - trans-only interactions (trans)
                              - cis-only interactions (cis)
          --tmpdir TEXT:DIR   Path to a folder where to store temporary data.
          --ignore-diags UINT [2]
                              Number of diagonals (including the main diagonal) to␣
↪mask before
                              balancing.
          --mad-max FLOAT:NONNEGATIVE [5]
                              Mask bins using the MAD-max filter.
                              Bins whose log marginal sum is less than --mad-max␣
↪median
                              absolute deviations below the median log marginal sum␣
↪of all the
                              bins in the same chromosome.
          --min-nnz UINT [10]
                              Mask rows with fewer than --min-nnz non-zero entries.
          --min-count UINT [0]
                              Mask rows with fewer than --min-count interactions.
          --tolerance FLOAT:NONNEGATIVE [1e-05]
                              Threshold of the variance of marginals used to␣
↪determine whether
                              the algorithm has converged.
          --max-iters UINT:POSITIVE [500]
                              Maximum number of iterations.
          --rescale-weights, --no-rescale-weights{false}
                              Rescale weights such that rows sum approximately to 2.
          --name TEXT         Name to use when writing weights to file.
                              Defaults to ICE, INTER_ICE and GW_ICE when --mode is␣
↪cis, trans
                              and gw, respectively.
          --create-weight-link, --no-create-weight-link{false}
                              Create a symbolic link to the balancing weights at
                              clr::/bins/weight.
                              Ignored when balancing .hic files
          --in-memory         Store all interactions in memory (greatly improves␣
↪performance).
          --stdout            Write balancing weights to stdout instead of writing␣
```

(continues on next page)

(continued from previous page)

```
↪them to the
                              input file.
          --chunk-size UINT:POSITIVE [10000000]
                              Number of interactions to process at once. Ignored when␣
↪using
                              --in-memory.
  -v,     --verbosity INT:INT in [1 - 4] [3]
                              Set verbosity of output to the console.
  -t,     --threads UINT:UINT in [1 - 32] [1]
                              Maximum number of parallel threads to spawn.
  -l,     --compression-lvl INT:INT in [0 - 19] [3]
                              Compression level used to compress temporary files␣
↪using ZSTD.
  -f,     --force             Overwrite existing files and datasets (if any).
```

## 17.4 hictk balance scale

```
Balance Hi-C files using SCALE.
hictk balance scale [OPTIONS] input
POSITIONALS:
  input TEXT:((.[ms]cool) OR (.hic)) AND (NOT .scool) REQUIRED
                              Path to the .hic, .cool or .mcool file to be balanced.
OPTIONS:
  -h,     --help              Print this help message and exit
          --mode TEXT:{gw,trans,cis} [gw]
                              Balance matrix using:
                              - genome-wide interactions (gw)
                              - trans-only interactions (trans)
                              - cis-only interactions (cis)
          --tmpdir TEXT       Path to a folder where to store temporary data.
          --max-percentile FLOAT [10]
                              Percentile used to compute the maximum number of nnz␣
↪values that
                              cause a row to be masked.
          --max-row-sum-err FLOAT:NONNEGATIVE [0.05]
                              Row sum threshold used to determine whether convergence␣
↪has been
                              achieved.
          --tolerance FLOAT:NONNEGATIVE [0.0001]
                              Threshold of the variance of marginals used to␣
↪determine whether
                              the algorithm has converged.
          --max-iters UINT:POSITIVE [500]
                              Maximum number of iterations.
          --rescale-weights, --no-rescale-weights{false}
                              Rescale weights such that the sum of the balanced␣
↪matrix is
                              similar to that of the input matrix.
          --name TEXT         Name to use when writing weights to file.
                              Defaults to SCALE, INTER_SCALE and GW_SCALE when --mode␣
↪is cis,
                              trans and gw, respectively.
          --create-weight-link, --no-create-weight-link{false}
                              Create a symbolic link to the balancing weights at
                              clr::/bins/weight.
```

(continues on next page)

(continued from previous page)

```
                              Ignored when balancing .hic files
          --in-memory         Store all interactions in memory (greatly improves␣
↪performance).
          --stdout            Write balancing weights to stdout instead of writing␣
↪them to the
                              input file.
          --chunk-size UINT:POSITIVE [10000000]
                              Number of interactions to process at once. Ignored when␣
↪using
                              --in-memory.
  -v,     --verbosity INT:INT in [1 - 4] [3]
                              Set verbosity of output to the console.
  -t,     --threads UINT:UINT in [1 - 32] [1]
                              Maximum number of parallel threads to spawn.
  -l,     --compression-lvl INT:INT in [0 - 19] [3]
                              Compression level used to compress temporary files␣
↪using ZSTD.
  -f,     --force             Overwrite existing files and datasets (if any).
```

## 17.5 hictk balance vc

```
Balance Hi-C matrices using VC.
hictk balance vc [OPTIONS] input
POSITIONALS:
  input TEXT:((.[ms]cool) OR (.hic)) AND (NOT .scool) REQUIRED
                              Path to the .hic, .cool or .mcool file to be balanced.
OPTIONS:
  -h,     --help              Print this help message and exit
          --mode TEXT:{gw,trans,cis} [gw]
                              Balance matrix using:
                              - genome-wide interactions (gw)
                              - trans-only interactions (trans)
                              - cis-only interactions (cis)
          --rescale-weights, --no-rescale-weights{false}
                              Rescale weights such that the sum of the balanced␣
↪matrix is
                              similar to that of the input matrix.
          --name TEXT         Name to use when writing weights to file.
                              Defaults to VC, INTER_VC and GW_VC when --mode is cis,␣
↪trans and
                              gw, respectively.
          --create-weight-link, --no-create-weight-link{false}
                              Create a symbolic link to the balancing weights at
                              clr::/bins/weight.
                              Ignored when balancing .hic files
          --stdout            Write balancing weights to stdout instead of writing␣
↪them to the
                              input file.
  -v,     --verbosity INT:INT in [1 - 4] [3]
                              Set verbosity of output to the console.
  -f,     --force             Overwrite existing files and datasets (if any).
```

## 17.6 hictk convert

```
Convert Hi-C files between different formats.
hictk convert [OPTIONS] input output
POSITIONALS:
  input TEXT:((.[ms]cool) OR (.hic)) AND (NOT .scool) REQUIRED
                              Path to the .hic, .cool or .mcool file to be converted.
  output TEXT REQUIRED        Output path. File extension is used to infer output␣
↪format.
OPTIONS:
  -h,     --help              Print this help message and exit
          --output-fmt TEXT:{cool,mcool,hic} [auto]
                              Output format (by default this is inferred from the␣
↪output file
                              extension).
                              Should be one of:
                              - cool
                              - mcool
                              - hic
  -r,     --resolutions UINT:POSITIVE ...
                              One or more resolutions to be converted. By default all
                              resolutions are converted.
          --normalization-methods TEXT [ALL]  ...
                              Name of one or more normalization methods to be copied.
                              By default, vectors for all known normalization methods␣
↪are
                              copied.
                              Pass NONE to avoid copying the normalization vectors.
          --fail-if-norm-not-found
                              Fail if any of the requested normalization vectors are␣
↪missing.
  -g,     --genome TEXT       Genome assembly name. By default this is copied from␣
↪the .hic
                              file metadata.
          --tmpdir TEXT:DIR   Path where to store temporary files.
          --chunk-size UINT:POSITIVE [10000000]
                              Batch size to use when converting .[m]cool to .hic.
  -v,     --verbosity INT:INT in [1 - 4] [3]
                              Set verbosity of output to the console.
  -t,     --threads UINT:UINT in [2 - 32] [2]
                              Maximum number of parallel threads to spawn.
                              When converting from hic to cool, only two threads will␣
↪be used.
  -l,     --compression-lvl UINT:INT in [1 - 12] [6]
                              Compression level used to compress interactions.
                              Defaults to 6 and 10 for .cool and .hic files,␣
↪respectively.
          --skip-all-vs-all, --no-skip-all-vs-all{false}
                              Do not generate All vs All matrix.
                              Has no effect when creating .[m]cool files.
          --count-type TEXT:{auto,int,float} [auto]
                              Specify the strategy used to infer count types when␣
↪converting
                              .hic files to .[m]cool format.
                              Can be one of: int, float, or auto.
  -f,     --force             Overwrite existing files (if any).
```

## 17.7 hictk dump

```
Read interactions and other kinds of data from .hic and Cooler files and write
them to stdout.
hictk dump [OPTIONS] uri
POSITIONALS:
  uri TEXT:(.[ms]cool) OR (.hic) REQUIRED
                              Path to a .hic, .cool or .mcool file (Cooler URI syntax
                              supported).
OPTIONS:
  -h,     --help              Print this help message and exit
          --resolution UINT:NONNEGATIVE
                              HiC matrix resolution (ignored when file is in .cool␣
↪format).
          --matrix-type ENUM:{observed,oe,expected} [observed]
                              Matrix type (ignored when file is not in .hic format).
          --matrix-unit ENUM:{BP,FRAG} [BP]
                              Matrix unit (ignored when file is not in .hic format).
  -t,     --table TEXT:{chroms,bins,pixels,normalizations,resolutions,cells,weights}␣
↪[pixels]
                              Name of the table to dump.
  -r,     --range TEXT [all]  Excludes: --query-file --cis-only --trans-only
                              Coordinates of the genomic regions to be dumped␣
↪following UCSC
                              style notation (chr1:0-1000).
          --range2 TEXT [all]  Needs: --range Excludes: --query-file --cis-only --
↪trans-only
                              Coordinates of the genomic regions to be dumped␣
↪following UCSC
                              style notation (chr1:0-1000).
          --query-file TEXT:(FILE) OR ({-}) Excludes: --range --range2 --cis-only --
↪trans-only
                              Path to a BEDPE file with the list of coordinates to be␣
↪fetched
                              (pass - to read queries from stdin).
          --cis-only Excludes: --range --range2 --query-file --trans-only
                              Dump intra-chromosomal interactions only.
          --trans-only Excludes: --range --range2 --query-file --cis-only
                              Dump inter-chromosomal interactions only.
  -b,     --balance TEXT [NONE]
                              Balance interactions using the given method.
          --sorted, --unsorted{false}
                              Return interactions in ascending order.
          --join, --no-join{false}
                              Output pixels in BG2 format.
```

## 17.8 hictk fix-mcool

```
Fix corrupted .mcool files.
hictk fix-mcool [OPTIONS] input output
POSITIONALS:
  input TEXT:.mcool REQUIRED  Path to a corrupted .mcool file.
  output TEXT REQUIRED        Path where to store the restored .mcool.
OPTIONS:
  -h,     --help              Print this help message and exit
          --tmpdir TEXT:DIR   Path to a folder where to store temporary data.
```

(continues on next page)

(continued from previous page)

```
          --skip-balancing    Do not recompute or copy balancing weights.
          --check-base-resolution
                              Check whether the base resolution is corrupted.
          --in-memory         Store all interactions in memory while balancing␣
↪(greatly
                              improves performance).
          --chunk-size UINT:POSITIVE [10000000]
                              Number of interactions to process at once during␣
↪balancing.
                              Ignored when using --in-memory.
  -v,     --verbosity INT:INT in [1 - 4] [3]
                              Set verbosity of output to the console.
  -t,     --threads UINT:UINT in [1 - 32] [1]
                              Maximum number of parallel threads to spawn (only␣
↪applies to the
                              balancing stage).
  -l,     --compression-lvl INT:INT in [0 - 19] [3]
                              Compression level used to compress temporary files␣
↪using ZSTD
                              (only applies to the balancing stage).
  -f,     --force             Overwrite existing files (if any).
```

## 17.9 hictk load

```
Build .cool and .hic files from interactions in various text formats.
hictk load [OPTIONS] interactions output-path
POSITIONALS:
  interactions TEXT:(FILE) OR ({-}) REQUIRED
                              Path to a file with the interactions to be loaded.
                              Common compression formats are supported (namely, bzip2,
↪ gzip,
                              lz4, lzo, xz, and zstd).
                              Pass "-" to indicate that interactions should be read␣
↪from stdin.
  output-path TEXT REQUIRED   Path to output file.
                              File extension will be used to infer the output format.
                              This behavior can be overridden by explicitly␣
↪specifying an
                              output format through option --output-fmt.
OPTIONS:
  -h,     --help              Print this help message and exit
  -c,     --chrom-sizes TEXT:FILE Excludes: --bin-table
                              Path to .chrom.sizes file.
                              Required when interactions are not in 4DN pairs format.
  -b,     --bin-size UINT:POSITIVE Excludes: --bin-table
                              Bin size (bp).
                              Required when --bin-table is not used.
          --bin-table TEXT:FILE Excludes: --chrom-sizes --bin-size
                              Path to a BED3+ file with the bin table.
  -f,     --format TEXT:{4dn,validpairs,bg2,coo} REQUIRED
                              Input format.
          --output-fmt TEXT:{auto,cool,hic} [auto]
                              Output format (by default this is inferred from the␣
↪output file
                              extension).
```

(continues on next page)

(continued from previous page)

```
                              Should be one of:
                              - auto
                              - cool
                              - hic
          --force             Force overwrite existing output file(s).
          --assembly TEXT [unknown]
                              Assembly name.
          --drop-unknown-chroms
                              Ignore records referencing unknown chromosomes.
          --one-based, --zero-based{false}
                              Interpret genomic coordinates or bins as one/zero based.
                              By default coordinates are assumed to be one-based for
                              interactions in 4dn and validpairs formats and zero-
↪based
                              otherwise.
          --count-as-float    Interactions are floats.
          --skip-all-vs-all, --no-skip-all-vs-all{false}
                              Do not generate All vs All matrix.
                              Has no effect when creating .cool files.
          --assume-sorted, --assume-unsorted{false}
                              Assume input files are already sorted.
          --validate-pixels, --no-validate-pixels{false}
                              Toggle pixel validation on or off.
                              When --no-validate-pixels is used and invalid pixels are
                              encountered, hictk will either crash or produce invalid␣
↪files.
          --transpose-lower-triangular-pixels, --no-transpose-lower-triangular-pixels
↪{false}
                              Transpose pixels overlapping the lower-triangular␣
↪matrix.
                              When --no-transpose-lower-triangular-pixels is used and␣
↪one or
                              more pixels overlapping with the lower triangular␣
↪matrix are
                              encountered an exception will be raised.
          --chunk-size UINT [10000000]
                              Number of pixels to buffer in memory.
  -l,     --compression-lvl UINT:INT bounded to [1 - 12]
                              Compression level used to compress interactions.
                              Defaults to 6 and 10 for .cool and .hic files,␣
↪respectively.
  -t,     --threads UINT:UINT in [2 - 32] [2]
                              Maximum number of parallel threads to spawn.
                              When loading interactions in a .cool file, only up to␣
↪two threads
                              will be used.
          --tmpdir TEXT:DIR   Path to a folder where to store temporary data.
  -v,     --verbosity INT:INT in [1 - 4] [3]
                              Set verbosity of output to the console.
```

## 17.10 hictk merge

```
Merge multiple Cooler or .hic files into a single file.
hictk merge [OPTIONS] input-files...
POSITIONALS:
```

(continues on next page)

(continued from previous page)

```
  input-files TEXT:((.[ms]cool) OR (.hic)) AND (NOT .scool) x 2 REQUIRED
                              Path to two or more Cooler or .hic files to be merged␣
↪(Cooler URI
                              syntax supported).
OPTIONS:
  -h,     --help              Print this help message and exit
  -o,     --output-file TEXT REQUIRED
                              Output Cooler or .hic file (Cooler URI syntax␣
↪supported).
          --output-fmt TEXT:{cool,hic} [auto]
                              Output format (by default this is inferred from the␣
↪output file
                              extension).
                              Should be one of:
                              - cool
                              - hic
          --resolution UINT:NONNEGATIVE
                              Hi-C matrix resolution (ignored when input files are in␣
↪.cool
                              format).
  -f,     --force             Force overwrite output file.
          --chunk-size UINT [10000000]
                              Number of pixels to store in memory before writing to␣
↪disk.
  -l,     --compression-lvl UINT:INT bounded to [1 - 12]
                              Compression level used to compress interactions.
                              Defaults to 6 and 10 for .cool and .hic files,␣
↪respectively.
  -t,     --threads UINT:UINT in [1 - 32] [1]
                              Maximum number of parallel threads to spawn.
                              When merging interactions in Cooler format, only a␣
↪single thread
                              will be used.
          --tmpdir TEXT:DIR   Path to a folder where to store temporary data.
          --skip-all-vs-all, --no-skip-all-vs-all{false}
                              Do not generate All vs All matrix.
                              Has no effect when merging .cool files.
          --count-type TEXT:{int,float} [int]
                              Specify the count type to be used when merging files.
                              Ignored when the output file is in .hic format.
  -v,     --verbosity INT:INT in [1 - 4] [3]
                              Set verbosity of output to the console.
```

## 17.11 hictk metadata

```
Print file metadata to stdout.
hictk metadata [OPTIONS] uri
POSITIONALS:
  uri TEXT:(.[ms]cool) OR (.hic) REQUIRED
                              Path to a .hic or .[ms]cool file (Cooler URI syntax␣
↪supported).
OPTIONS:
  -h,     --help              Print this help message and exit
  -f,     --output-format TEXT:{json,toml,yaml} [json]
                              Format used to return file metadata.
```

(continues on next page)

(continued from previous page)

```
                              Should be one of: json, toml, or yaml.
          --include-file-path, --exclude-file-path{false}
                              Output the given input path using attribute "uri".
          --recursive         Print metadata for each resolution or cell contained in␣
↪a
                              multi-resolution or single-cell file.
```

## 17.12 hictk rename-chromosomes

```
Rename chromosomes found in Cooler files.
hictk rename-chromosomes [OPTIONS] uri
POSITIONALS:
  uri TEXT:.[ms]cool REQUIRED Path to a .[ms]cool file (Cooler URI syntax supported).
OPTIONS:
  -h,     --help              Print this help message and exit
          --name-mappings TEXT Excludes: --add-chr-prefix --remove-chr-prefix
                              Path to a two column TSV with pairs of chromosomes to␣
↪be renamed.
                              The first column should contain the original chromosome␣
↪name,
                              while the second column should contain the destination␣
↪name to
                              use when renaming.
          --add-chr-prefix Excludes: --name-mappings --remove-chr-prefix
                              Prefix chromosome names with "chr".
          --remove-chr-prefix Excludes: --name-mappings --add-chr-prefix
                              Remove prefix "chr" from chromosome names.
  -v,     --verbosity INT:INT in [1 - 4] [3]
                              Set verbosity of output to the console.
```

## 17.13 hictk validate

```
Validate .hic and Cooler files.
hictk validate [OPTIONS] uri
POSITIONALS:
  uri TEXT REQUIRED           Path to a .hic or .[ms]cool file (Cooler URI syntax␣
↪supported).
OPTIONS:
  -h,     --help              Print this help message and exit
          --validate-index    Validate Cooler index (may take a long time).
          --validate-pixels   Validate pixels found in Cooler files (may take a long␣
↪time).
  -f,     --output-format TEXT:{json,toml,yaml} [json]
                              Format used to report the outcome of file validation.
                              Should be one of: json, toml, or yaml.
          --include-file-path, --exclude-file-path{false}
                              Output the given input path using attribute "uri".
          --exhaustive, --fail-fast{false}
                              When processing multi-resolution or single-cell files,␣
↪do not
                              fail as soon as the first error is detected.
          --quiet             Don't print anything to stdout. Success/failure is␣
↪reported
                              through exit codes.
```

## 17.14 hictk zoomify

```
Convert single-resolution Cooler and .hic files to multi-resolution by
coarsening.
hictk zoomify [OPTIONS] cooler/hic [m]cool/hic
POSITIONALS:
  cooler/hic TEXT:((.[ms]cool) OR (.hic)) AND (NOT .scool) REQUIRED
                              Path to a .cool or .hic file (Cooler URI syntax␣
↪supported).
  [m]cool/hic TEXT REQUIRED   Output path.
                              When zoomifying Cooler files, providing a single␣
↪resolution
                              through --resolutions and specifying --no-copy-base-
↪resolution,
                              the output file will be in .cool format.
OPTIONS:
  -h,     --help              Print this help message and exit
          --force             Force overwrite existing output file(s).
          --resolutions UINT:POSITIVE ...
                              One or more resolutions to be used for coarsening.
          --copy-base-resolution, --no-copy-base-resolution{false}
                              Copy the base resolution to the output file.
          --nice-steps, --pow2-steps{false} [--nice-steps]
                              Use nice or power of two steps to automatically␣
↪generate the list
                              of resolutions.
                              Example:
                              Base resolution: 1000
                              Pow2: 1000, 2000, 4000, 8000...
                              Nice: 1000, 2000, 5000, 10000...
  -l,     --compression-lvl UINT:INT bounded to [1 - 12] [6]
                              Compression level used to compress interactions.
                              Defaults to 6 and 10 for .mcool and .hic files,␣
↪respectively.
  -t,     --threads UINT:UINT in [1 - 32] [1]
                              Maximum number of parallel threads to spawn.
                              When zoomifying interactions from a .cool file, only a␣
↪single
                              thread will be used.
          --chunk-size UINT [10000000]
                              Number of pixels to buffer in memory.
                              Only used when zoomifying .hic files.
          --skip-all-vs-all, --no-skip-all-vs-all{false}
                              Do not generate All vs All matrix.
                              Has no effect when zoomifying .cool files.
          --tmpdir TEXT:DIR   Path to a folder where to store temporary data.
  -v,     --verbosity INT:INT in [1 - 4] [3]
                              Set verbosity of output to the console.
```

CHAPTER

EIGHTEEN

# C++ API REFERENCE

hictk C++ API is structured as follows:

## 18.1 Generic API

hictk generic API allows users to transparently operate on .hic .cool files. There is virtually no runtime overhead when using the `File` and `PixelSelector` classes. However iterating over `Pixel`s using this API is slightly slower than using the format-specific APIs.

Refer to examples in the *Quickstart (API)* section for how to use the generic API without incurring into any overhead when iterating over `Pixel`s overlapping queries.

### 18.1.1 Common

enum class **QUERY_TYPE**

enumerator **BED**

enumerator **UCSC**

### 18.1.2 File handle

class **File**

This class implements a generic file handle capable of transparently operating on .cool and .hic files.

**Constructors**

**File**(cooler::*File* clr);

**File**(hic::*File* hf);

**File**(
std::string_view uri, std::optional<std::uint32_t> resolution = {}, hic::*MatrixType* type = hic::*MatrixType*::*observed*, hic::*MatrixUnit* unit = hic::*MatrixUnit*::*BP*
);

Constructors for `File` class. `resolution` is a mandatory argument when opening .hic files. Matrix `type` and `unit` are ignored when operating on .cool files.

**Accessors**

[[nodiscard]] std::string **uri**() const;

Returns the URI of the open file. Always returns the file path when file is .hic.

[[nodiscard]] std::string **path**() const;

Returns the path to the open file.

[[nodiscard]] constexpr bool **is_hic**() const noexcept;

[[nodiscard]] constexpr bool **is_cooler**() const noexcept;

Test whether the open file is in .hic or .cool format.

[[nodiscard]] auto **chromosomes**() const -> const *Reference*&;

[[nodiscard]] auto **bins**() const -> const *BinTable*&;

[[nodiscard]] std::shared_ptr<const *BinTable*> **bins_ptr**() const;

Accessors to the chromosomes and bin table of the open file.

[[nodiscard]] std::uint32_t **resolution**() const;

[[nodiscard]] std::uint64_t **nbins**() const;

[[nodiscard]] std::uint64_t **nchroms**(bool include_ALL = false) const;

Accessors for common attributes. Calling any of these accessors does not involve any computation.

[[nodiscard]] bool **has_normalization**(std::string_view normalization) const;

[[nodiscard]] const balancing::Weights &**normalization**(
    std::string_view normalization_
) const;

[[nodiscard]] std::shared_ptr<const balancing::Weights> **normalization_ptr**(
    std::string_view normalization_
) const;

Accessors for normalization methods/vectors.

**Fetch methods (1D queries)**

[[nodiscard]] *PixelSelector* **fetch**(
    const balancing::Method &normalization = balancing::Method::NONE()
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view range, const balancing::Method &normalization = balancing::Method::NONE(), *QUERY_TYPE* query_type = *QUERY_TYPE*::*UCSC*
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view chrom_name, std::uint32_t start, std::uint32_t end, const balancing::Method &normalization = balancing::Method::NONE()
) const;

Return a *PixelSelector* object that can be used to fetch pixels overlapping 1D (symmetric) queries.

**Example usage:**

```
hictk::File f{"myfile.hic", 1'000};

// Fetch all pixels
const auto sel1 = f.fetch();

// Fetch all pixels (normalized with VC);
const auto sel2 = f.fetch(balancing::Method::VC());

// Fetch pixels overlapping chr1
const auto sel3 = f.fetch("chr1");

// Fetch pixels overlapping a region of interest
```

(continues on next page)

(continued from previous page)

```
const auto sel4 = f.fetch("chr1:10,000,000-20,000,000");
const auto sel5 = f.fetch("chr1", 10'000'000, 20'000'000");

// Fetch pixels using a BED query
const auto sel6 = f.fetch("chr1\t10000000\t20000000",
                          balancing::Method::NONE(),
                          QUERY_TYPE::BED);
```

**Fetch methods (2D queries)**

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view range1, std::string_view range2, const balancing::Method &normalization = balancing::Method::NONE(), *QUERY_TYPE* query_type = *QUERY_TYPE*::*UCSC*
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view chrom1_name, std::uint32_t start1, std::uint32_t end1, std::string_view chrom2_name, std::uint32_t start2, std::uint32_t end2, const balancing::Method &normalization = balancing::Method::NONE()
) const;

Return a *PixelSelector* object that can be used to fetch pixels overlapping 2D (asymmetric) queries.

**Example usage:**

```
hictk::File f{"myfile.hic", 1'000};

// Fetch pixels overlapping chr1:chr2
const auto sel1 = f.fetch("chr1", "chr2");

// Fetch pixels overlapping a region of interest
const auto sel2 = f.fetch("chr1:10,000,000-20,000,000",
                          "chr2:10,000,000-20,000,000");
const auto sel3 = f.fetch("chr1", 10'000'000, 20'000'000,
                          "chr2", 10'000'000, 20'000'000);
```

**Advanced**

template<typename **FileT**>
[[nodiscard]] constexpr const *FileT* &**get**() const;

template<typename **FileT**>
[[nodiscard]] constexpr *FileT* &**get**();

[[nodiscard]] constexpr auto **get**() const noexcept -> const FileVar&;

[[nodiscard]] constexpr auto **get**() noexcept -> FileVar&;

Methods to get the underlying *hic::File* or *cooler::File* file handle or a `std::variant` of thereof.

**Example usage:**

```
hictk::File f{"myfile.hic", 1'000};

assert(f.get<hic::File>().path() == "myfile.hic");
assert(f.get<cooler::File>().path() == "myfile.hic");  // Throws an exception

const auto fvar = f.get();
std::visit([](const auto& f) {
```

(continues on next page)

(continued from previous page)

```
  assert(f.path() == "myfile.hic");
}, fvar);
```

### 18.1.3 Pixel selector

class **PixelSelector**

This class implements a generic, lightweight pixel selector object.

*PixelSelector* objects are constructed and returned by *File::fetch()* methods. Users are **not** supposed to construct *PixelSelector* objects themselves.

**Iteration**

template<typename **N**>
[[nodiscard]] auto **begin**(
    bool sorted = true
) const -> iterator<*N*>;

template<typename **N**>
[[nodiscard]] auto **end**() const -> iterator<*N*>;

template<typename **N**>
[[nodiscard]] auto **cbegin**(
    bool sorted = true
) const -> iterator<*N*>;

template<typename **N**>
[[nodiscard]] auto **cend**() const -> iterator<*N*>;

Return an InputIterator to traverse pixels overlapping the genomic coordinates used to create the *PixelSelector*.

Specifying `sorted = false` will improve throughput for queries over .hic files.

When operating on .cool files, pixels are always returned sorted by genomic coordinates.

**Example usage:**

```
hictk::File f{"myfile.hic", 1'000};
const auto sel = f.fetch();

std::for_each(sel.begin<std::int32_t>(), sel.end<std::int32_t>(),
              [&](const auto& pixel) { fmt::print("{}\n", pixel); });

// STDOUT
// 0  0 12
// 0  2 7
// 0  4 1
// ...
```

**Fetch at once**

template<typename **N**>
[[nodiscard]] std::vector<*Pixel*<*N*>> **read_all**() const;

Read and return all *Pixel*s at once using a `std::vector`.

**Accessors**

[[nodiscard]] const *PixelCoordinates* &**coord1**() const noexcept;

[[nodiscard]] const *PixelCoordinates* &**coord2**() const noexcept;

[[nodiscard]] std::uint64_t **size**(bool upper_triangular = true) const;

Return the genomic coordinates used to construct the *PixelSelector*.

[[nodiscard]] const *BinTable* &**bins**() const noexcept;

[[nodiscard]] std::shared_ptr<const *BinTable*> **bins_ptr**() const noexcept;

Return the *BinTable* used to map *Pixel*s to genomic *Bin*s.

[[nodiscard]] const balancing::Weights &**weights**() const noexcept;

Return the balancing weights associated with the *PixelSelector* instance.

**Advanced**

template<typename **PixelSelectorT**>
[[nodiscard]] constexpr const *PixelSelectorT* &**get**(
) const;

template<typename **PixelSelectorT**>
[[nodiscard]] constexpr *PixelSelectorT* &**get**(
);

[[nodiscard]] constexpr auto **get**() const noexcept -> const PixelSelectorVar&;

[[nodiscard]] constexpr auto **get**() noexcept -> PixelSelectorVar&;

**Example usage:**

```
hictk::File f{"myfile.hic", 1'000};

const auto sel = f.fetch();

assert(f.get<hic::PixelSelector>().matrix_type() == hic::MatrixType::observed
↪");
f.get<cooler::PixelSelector>();  // Throws an exception

const auto selvar = sel.get();
std::visit([](const auto& s) { assert(s.bins().resolution() == 1'000); },␣
↪selvar);
```

## 18.2 Cooler API

API to operate on .cool files. Compared to the generic API, this API provides:

- more control over how files are opened
- direct access to HDF5 group and datasets
- lower overhead
- support for creating .cool files
- support for opening collections of Coolers (e.g. .mcool and .scool files)

### 18.2.1 Single-resolution Cooler (.cool)

class **File**

**Constructors**

**File**(const *File* &other) = delete;

**File**(*File* &&other) noexcept = default;

[[nodiscard]] explicit **File**(
    std::string_view uri, std::size_t cache_size_bytes = DEFAULT_HDF5_CACHE_SIZE * 4, bool validate = true
);

[[nodiscard]] explicit **File**(
    RootGroup entrypoint, std::size_t cache_size_bytes = DEFAULT_HDF5_CACHE_SIZE * 4, bool validate = true
);

**Factory functions**

[[nodiscard]] static *File* **open_random_access**(
    RootGroup entrypoint, std::size_t cache_size_bytes = DEFAULT_HDF5_CACHE_SIZE * 4, bool validate = true
);

[[nodiscard]] static *File* **open_read_once**(
    RootGroup entrypoint, std::size_t cache_size_bytes = DEFAULT_HDF5_CACHE_SIZE * 4, bool validate = true
);

template<typename **PixelT** = DefaultPixelT>
[[nodiscard]] static *File* **create**(
    RootGroup entrypoint, const *Reference* &chroms, std::uint32_t bin_size, Attributes attributes = Attributes::init<*PixelT*>(0), std::size_t cache_size_bytes = DEFAULT_HDF5_CACHE_SIZE * 4, std::uint32_t compression_lvl = DEFAULT_COMPRESSION_LEVEL
);

template<typename **PixelT** = DefaultPixelT>
[[nodiscard]] static *File* **create**(
    std::string_view uri, const *Reference* &chroms, std::uint32_t bin_size, bool overwrite_if_exists = false, Attributes attributes = Attributes::init<*PixelT*>(0), std::size_t cache_size_bytes = DEFAULT_HDF5_CACHE_SIZE * 4, std::uint32_t compression_lvl = DEFAULT_COMPRESSION_LEVEL
);

**Open/close methods**

[[nodiscard]] static *File* **open_random_access**(
    std::string_view uri, std::size_t cache_size_bytes = DEFAULT_HDF5_CACHE_SIZE * 4, bool validate = true
);

[[nodiscard]] static *File* **open_read_once**(
    std::string_view uri, std::size_t cache_size_bytes = DEFAULT_HDF5_CACHE_SIZE * 4, bool validate = true
);

void **close**();

Note that *File*s are automatically closed upon destruction.

**Operators**

*File* &**operator=**(const *File* &other) = delete;

*File* &**operator=**(*File* &&other) noexcept = default;

[[nodiscard]] explicit constexpr **operator bool**() const noexcept;

[[nodiscard]] constexpr bool **operator!**() const noexcept;

Return whether the *File* is in a valid state and other member functions can be safely called.

**Accessors**

[[nodiscard]] std::string **uri**() const;

[[nodiscard]] std::string **hdf5_path**() const;

[[nodiscard]] std::string **path**() const;

[[nodiscard]] auto **chromosomes**() const noexcept -> const *Reference*&;

[[nodiscard]] auto **bins**() const noexcept -> const *BinTable*&;

[[nodiscard]] auto **bins_ptr**() const noexcept -> std::shared_ptr<const *BinTable*>;

[[nodiscard]] std::uint32_t **resolution**() const noexcept;

[[nodiscard]] std::uint64_t **nbins**() const;

[[nodiscard]] std::uint64_t **nchroms**() const;

[[nodiscard]] std::uint64_t **nnz**() const;

[[nodiscard]] auto **attributes**() const noexcept -> const Attributes&;

[[nodiscard]] auto **group**(std::string_view group_name) -> Group&;

[[nodiscard]] auto **dataset**(std::string_view dataset_name) -> Dataset&;

[[nodiscard]] auto **group**(std::string_view group_name) const -> const Group&;

[[nodiscard]] auto **dataset**(
    std::string_view dataset_name
) const -> const Dataset&;

[[nodiscard]] const NumericVariant &**pixel_variant**() const noexcept;

template<typename **T**>
[[nodiscard]] bool **has_pixel_of_type**() const noexcept;

[[nodiscard]] bool **has_signed_pixels**() const noexcept;

[[nodiscard]] bool **has_unsigned_pixels**() const noexcept;

[[nodiscard]] bool **has_integral_pixels**() const noexcept;

[[nodiscard]] bool **has_float_pixels**() const noexcept;

**Iteration**

template<typename **N**>
[[nodiscard]] typename *PixelSelector*::iterator<*N*> **begin**(
    std::string_view weight_name = "NONE"
) const;

template<typename **N**>
[[nodiscard]] typename *PixelSelector*::iterator<*N*> **end**(
    std::string_view weight_name = "NONE"
) const;

template<typename **N**>
[[nodiscard]] typename *PixelSelector*::iterator<*N*> **cbegin**(
    std::string_view weight_name = "NONE"
) const;

template<typename **N**>
[[nodiscard]] typename *PixelSelector*::iterator<*N*> **cend**(
    std::string_view weight_name = "NONE"
) const;

**Fetch methods (1D queries)**

[[nodiscard]] *PixelSelector* **fetch**(
    const balancing::Method &normalization = balancing::Method::NONE(), bool load_index = false
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::shared_ptr<const balancing::Weights> weights, bool load_index = false
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view range, std::shared_ptr<const balancing::Weights> weights, *QUERY_TYPE* query_type = *QUERY_TYPE*::*UCSC*
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view chrom_name, std::uint32_t start, std::uint32_t end, std::shared_ptr<const balancing::Weights> weights
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view range, const balancing::Method &normalization = balancing::Method::NONE(), *QUERY_TYPE* query_type = *QUERY_TYPE*::*UCSC*
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view chrom_name, std::uint32_t start, std::uint32_t end, const balancing::Method &normalization = balancing::Method::NONE()
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::uint64_t first_bin, std::uint64_t last_bin, std::shared_ptr<const balancing::Weights> weights = nullptr
) const;

**Fetch methods (2D queries)**

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view range1, std::string_view range2, std::shared_ptr<const balancing::Weights> weights, *QUERY_TYPE* query_type = *QUERY_TYPE*::*UCSC*
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view chrom1_name, std::uint32_t start1, std::uint32_t end1, std::string_view chrom2_name, std::uint32_t start2, std::uint32_t end2, std::shared_ptr<const balancing::Weights> weights
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view range1, std::string_view range2, const balancing::Method &normalization = balancing::Method::NONE(), *QUERY_TYPE* query_type = *QUERY_TYPE*::*UCSC*
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view chrom1_name, std::uint32_t start1, std::uint32_t end1, std::string_view chrom2_name, std::uint32_t start2, std::uint32_t end2, const balancing::Method &normalization = balancing::Method::NONE()
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::uint64_t first_bin1, std::uint64_t last_bin1, std::uint64_t first_bin2, std::uint64_t last_bin2, std::shared_ptr<const balancing::Weights> weights = nullptr
) const;

**Write pixels**

template<typename **PixelIt**, typename = std::enable_if_t<is_iterable_v<*PixelIt*>>>
void **append_pixels**(
    *PixelIt* first_pixel, *PixelIt* last_pixel, bool validate = false
);

**Normalization**

[[nodiscard]] bool **has_normalization**(std::string_view normalization) const;

[[nodiscard]] std::shared_ptr<const balancing::Weights> **normalization_ptr**(
    std::string_view normalization_, bool rescale = false
) const;

[[nodiscard]] std::shared_ptr<const balancing::Weights> **normalization_ptr**(
    std::string_view normalization_, balancing::Weights::Type type, bool rescale = false
) const;

[[nodiscard]] const balancing::Weights &**normalization**(
    std::string_view normalization_, bool rescale = false
) const;

[[nodiscard]] const balancing::Weights &**normalization**(
    std::string_view normalization_, balancing::Weights::Type type, bool rescale = false
) const;

[[nodiscard]] bool **has_normalization**(
    const balancing::Method &normalization
) const;

[[nodiscard]] std::shared_ptr<const balancing::Weights> **normalization_ptr**(
    const balancing::Method &normalization_, bool rescale = false
) const;

[[nodiscard]] std::shared_ptr<const balancing::Weights> **normalization_ptr**(
    const balancing::Method &normalization_, balancing::Weights::Type type, bool rescale = false
) const;

[[nodiscard]] const balancing::Weights &**normalization**(
    const balancing::Method &normalization_, bool rescale = false
) const;

[[nodiscard]] const balancing::Weights &**normalization**(
    const balancing::Method &normalization_, balancing::Weights::Type type, bool rescale = false
) const;

[[nodiscard]] std::vector<balancing::Method> **avail_normalizations**() const;

bool **purge_weights**(std::string_view name = "");

template<typename **It**>
static void **write_weights**(
    std::string_view uri, std::string_view name, *It* first_weight, *It* last_weight, bool overwrite_if_exists = false, bool divisive = false
);

template<typename **It**>
void **write_weights**(
    std::string_view name, *It* first_weight, *It* last_weight, bool overwrite_if_exists = false, bool divisive = false
);

**Others**

void **flush**();

void **validate_bins**(bool full = false) const;

### 18.2.2 Multi-resolution Cooler (.mcool)

class **MultiResFile**

**Constructors**

explicit **MultiResFile**(
    const std::filesystem::path &path, HighFiveAccessMode mode = HighFive::File::ReadOnly
);

**Factory functions**

[[nodiscard]] static *MultiResFile* **create**(
    const std::filesystem::path &path, const *Reference* &chroms, bool force_overwrite = false
);

template<typename **ResolutionIt**>
[[nodiscard]] static *MultiResFile* **create**(
    const std::filesystem::path &path, const *File* &base, *ResolutionIt* first_res, *ResolutionIt* last_res, bool force_overwrite = false
);

**Open/close methods**

[[nodiscard]] *File* **open**(std::uint32_t resolution) const;

**Operators**

[[nodiscard]] explicit **operator bool**() const noexcept;

**Accessors**

[[nodiscard]] std::string **path**() const;

[[nodiscard]] auto **chromosomes**() const noexcept -> const *Reference*&;

[[nodiscard]] constexpr const std::vector<std::uint32_t> &**resolutions**(
) const noexcept;

[[nodiscard]] constexpr const MultiResAttributes &**attributes**() const noexcept;

**Modifiers**

*File* **copy_resolution**(const cooler::*File* &clr);

template<typename **N** = DefaultPixelT>
*File* **create_resolution**(
    std::uint32_t resolution, Attributes attributes = Attributes::init<*N*>(0)
);

RootGroup **init_resolution**(std::uint32_t resolution);

**Others**

[[nodiscard]] static std::uint32_t **compute_base_resolution**(
    const std::vector<std::uint32_t> &resolutions, std::uint32_t target_res
);

template<typename **N** = std::int32_t>
static void **coarsen**(
    const *File* &clr1, *File* &clr2, std::vector<*ThinPixel*<*N*>> &buffer
);

### 18.2.3 Single-cell Cooler (.scool)

class **SingleCellFile**

**Constructors**

explicit **SingleCellFile**(
    const std::filesystem::path &path, HighFiveAccessMode mode = HighFive::File::ReadOnly
);

**Factory functions**

[[nodiscard]] static *SingleCellFile* **create**(
    const std::filesystem::path &path, const *Reference* &chroms, std::uint32_t bin_size, bool force_overwrite = false
);

**Open/close functions**

[[nodiscard]] *File* **open**(std::string_view cell) const;

**Operators**

[[nodiscard]] explicit **operator bool**() const noexcept;

**Accessors**

[[nodiscard]] std::string **path**() const;

[[nodiscard]] auto **chromosomes**() const noexcept -> const *Reference*&;

[[nodiscard]] auto **bins**() const noexcept -> const *BinTable*&;

[[nodiscard]] auto **bins_ptr**() const noexcept -> std::shared_ptr<const *BinTable*>;

[[nodiscard]] std::uint32_t **resolution**() const noexcept;

[[nodiscard]] constexpr const phmap::btree_set<std::string> &**cells**(
) const noexcept;

[[nodiscard]] constexpr const SingleCellAttributes &**attributes**() const noexcept;

**Modifiers**

template<typename **N**>

*File* **create_cell**(
    std::string_view cell, Attributes attrs = Attributes::init<*N*>(0)
);

**Others**

template<typename **N**>
*File* **aggregate**(
    std::string_view uri, bool overwrite_if_exists = false, std::size_t chunk_size = 500'000, std::size_t update_frequency = 10'000'000
) const;

### 18.2.4 Pixel selector

class **PixelSelector**

**Operators**

[[nodiscard]] bool **operator==**(const *PixelSelector* &other) const noexcept;

[[nodiscard]] bool **operator!=**(const *PixelSelector* &other) const noexcept;

**Iteration**

template<typename **N**>
[[nodiscard]] auto **begin**() const -> iterator<*N*>;

template<typename **N**>
[[nodiscard]] auto **end**() const -> iterator<*N*>;

template<typename **N**>
[[nodiscard]] auto **cbegin**() const -> iterator<*N*>;

template<typename **N**>
[[nodiscard]] auto **cend**() const -> iterator<*N*>;

**Fetch at once**

template<typename **N**>
[[nodiscard]] std::vector<*Pixel*<*N*>> **read_all**() const;

**Accessors**

[[nodiscard]] const *PixelCoordinates* &**coord1**() const noexcept;

[[nodiscard]] const *PixelCoordinates* &**coord2**() const noexcept;

[[nodiscard]] std::uint64_t **size**(bool upper_triangular = true) const;

[[nodiscard]] const *BinTable* &**bins**() const noexcept;

[[nodiscard]] std::shared_ptr<const *BinTable*> **bins_ptr**() const noexcept;

## 18.3 Hi-C API

API to operate on .hic files. Compared to the generic API, this API provides:

- more control over how files are opened
- access to .hic-specific metadata
- control over the interaction block cache

### 18.3.1 Common

enum class **MatrixType**

enumerator **observed**

enumerator **oe**

enumerator **expected**

enum class **MatrixUnit**

enumerator **BP**

enumerator **FRAG**

enum class **QUERY_TYPE**

enumerator **BED**

enumerator **UCSC**

### 18.3.2 File handle

class **File**

**Constructors**

explicit **File**(
std::string url_, std::optional<std::uint32_t> resolution_, *MatrixType* type_ = *MatrixType*::*observed*, *MatrixUnit* unit_ = *MatrixUnit*::*BP*, std::uint64_t block_cache_capacity = 0
);

**Open/close methods**

*File* &**open**(
std::string url_, std::optional<std::uint32_t> resolution_, *MatrixType* type_ = *MatrixType*::*observed*, *MatrixUnit* unit_ = *MatrixUnit*::*BP*, std::uint64_t block_cache_capacity = 0
);

*File* &**open**(
std::uint32_t resolution_, *MatrixType* type_ = *MatrixType*::*observed*, *MatrixUnit* unit_ = *MatrixUnit*::*BP*, std::uint64_t block_cache_capacity = 0
);

**Accessors**

[[nodiscard]] bool **has_resolution**(std::uint32_t resolution) const;

[[nodiscard]] const std::string &**path**() const noexcept;

[[nodiscard]] const std::string &**name**() const noexcept;

[[nodiscard]] std::int32_t **version**() const noexcept;

[[nodiscard]] const *Reference* &**chromosomes**() const noexcept;

[[nodiscard]] const *BinTable* &**bins**() const noexcept;

[[nodiscard]] std::shared_ptr<const *BinTable*> **bins_ptr**() const noexcept;

[[nodiscard]] std::uint32_t **resolution**() const noexcept;

[[nodiscard]] std::uint64_t **nbins**() const;

[[nodiscard]] std::uint64_t **nchroms**(bool include_ALL = false) const;

[[nodiscard]] const std::string &**assembly**() const noexcept;

[[nodiscard]] const std::vector<std::uint32_t> &**avail_resolutions**(
) const noexcept;

[[nodiscard]] bool **has_normalization**(std::string_view normalization) const;

[[nodiscard]] std::vector<balancing::Method> **avail_normalizations**() const;

[[nodiscard]] const balancing::Weights &**normalization**(
    const balancing::Method &norm, const *Chromosome* &chrom
) const;

[[nodiscard]] const balancing::Weights &**normalization**(
    std::string_view norm, const *Chromosome* &chrom
) const;

[[nodiscard]] const balancing::Weights &**normalization**(
    const balancing::Method &norm
) const;

[[nodiscard]] const balancing::Weights &**normalization**(
    std::string_view norm
) const;

[[nodiscard]] std::shared_ptr<const balancing::Weights> **normalization_ptr**(
    const balancing::Method &norm, const *Chromosome* &chrom
) const;

[[nodiscard]] std::shared_ptr<const balancing::Weights> **normalization_ptr**(
    std::string_view norm, const *Chromosome* &chrom
) const;

[[nodiscard]] std::shared_ptr<const balancing::Weights> **normalization_ptr**(
    const balancing::Method &norm
) const;

[[nodiscard]] std::shared_ptr<const balancing::Weights> **normalization_ptr**(
    std::string_view norm
) const;

**Fetch methods (1D queries)**

[[nodiscard]] PixelSelectorAll **fetch**(
    const balancing::Method &norm = balancing::Method::NONE(), std::optional<std::uint64_t> diagonal_band_width = {}
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view range, const balancing::Method &norm = balancing::Method::NONE(), *QUERY_TYPE* query_type = *QUERY_TYPE*::*UCSC*, std::optional<std::uint64_t> diagonal_band_width = {}
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view chrom_name, std::uint32_t start, std::uint32_t end, const balancing::Method &norm = balancing::Method::NONE(), std::optional<std::uint64_t> diagonal_band_width = {}
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::uint64_t first_bin, std::uint64_t last_bin, const balancing::Method &norm = balancing::Method::NONE(), std::optional<std::uint64_t> diagonal_band_width = {}
) const;

**Fetch methods (2D queries)**

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view range1, std::string_view range2, const balancing::Method &norm = balancing::Method::NONE(), *QUERY_TYPE* query_type = *QUERY_TYPE*::*UCSC*, std::optional<std::uint64_t> diagonal_band_width = {}
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::string_view chrom1_name, std::uint32_t start1, std::uint32_t end1, std::string_view chrom2_name, std::uint32_t start2, std::uint32_t end2, const balancing::Method &norm = balancing::Method::NONE(), std::optional<std::uint64_t> diagonal_band_width = {}
) const;

[[nodiscard]] *PixelSelector* **fetch**(
    std::uint64_t first_bin1, std::uint64_t last_bin1, std::uint64_t first_bin2, std::uint64_t last_bin2, const balancing::Method &norm = balancing::Method::NONE(), std::optional<std::uint64_t> diagonal_band_width = {}
) const;

**Caching**

[[nodiscard]] std::size_t **num_cached_footers**() const noexcept;

void **purge_footer_cache**();

[[nodiscard]] double **block_cache_hit_rate**() const noexcept;

void **reset_cache_stats**() const noexcept;

void **clear_cache**() noexcept;

void **optimize_cache_size**(
    std::size_t upper_bound = (std::numeric_limits<std::size_t>::max)()
);

void **optimize_cache_size_for_iteration**(
    std::size_t upper_bound = (std::numeric_limits<std::size_t>::max)()
);

void **optimize_cache_size_for_random_access**(
    std::size_t upper_bound = (std::numeric_limits<std::size_t>::max)()
);

[[nodiscard]] std::size_t **cache_capacity**() const noexcept;

### 18.3.3 Pixel selector

class **PixelSelector**

**Operators**

[[nodiscard]] bool **operator==**(const *PixelSelector* &other) const noexcept;

[[nodiscard]] bool **operator!=**(const *PixelSelector* &other) const noexcept;

**Iteration**

template<typename **N**>

[[nodiscard]] auto **begin**(
    bool sorted = true
) const -> iterator<*N*>;

template<typename **N**>
[[nodiscard]] auto **end**() const -> iterator<*N*>;

template<typename **N**>
[[nodiscard]] auto **cbegin**(
    bool sorted = true
) const -> iterator<*N*>;

template<typename **N**>
[[nodiscard]] auto **cend**() const -> iterator<*N*>;

**Fetch at once**

template<typename **N**>
[[nodiscard]] std::vector<*Pixel*<*N*>> **read_all**() const;

**Accessors**

[[nodiscard]] const *PixelCoordinates* &**coord1**() const noexcept;

[[nodiscard]] const *PixelCoordinates* &**coord2**() const noexcept;

[[nodiscard]] std::uint64_t **size**(bool upper_triangular = true) const;

[[nodiscard]] *MatrixType* **matrix_type**() const noexcept;

[[nodiscard]] const balancing::Method &**normalization**() const noexcept;

[[nodiscard]] *MatrixUnit* **unit**() const noexcept;

[[nodiscard]] std::uint32_t **resolution**() const noexcept;

[[nodiscard]] const *Chromosome* &**chrom1**() const noexcept;

[[nodiscard]] const *Chromosome* &**chrom2**() const noexcept;

[[nodiscard]] const balancing::Weights &**weights1**() const noexcept;

[[nodiscard]] const balancing::Weights &**weights2**() const noexcept;

[[nodiscard]] const *BinTable* &**bins**() const noexcept;

[[nodiscard]] std::shared_ptr<const *BinTable*> **bins_ptr**() const noexcept;

[[nodiscard]] const internal::HiCFooterMetadata &**metadata**() const noexcept;

[[nodiscard]] bool **is_inter**() const noexcept;

[[nodiscard]] bool **is_intra**() const noexcept;

[[nodiscard]] bool **empty**() const noexcept;

**Caching**

[[nodiscard]] std::size_t **estimate_optimal_cache_size**(
    std::size_t num_samples = 500
) const;

void **clear_cache**() const;

## 18.4 Shared Types

Types documented in this page are used throughout hictk code-base to model various concepts such as genomic intervals, reference genomes, bins and pixels.

### 18.4.1 Chromosome

class **Chromosome**

This class models chromosomes as triplets consisting of:

- A numeric identifier
- The chromosome name
- The chromosome size

*Chromosome*s are compared by ID.

**Constructors**

**Chromosome**() = default;

**Chromosome**(std::uint32_t id_, std::string name_, std::uint32_t size_) noexcept;

**Operators**

[[nodiscard]] explicit constexpr **operator bool**() const noexcept;

**Accessors**

[[nodiscard]] constexpr std::uint32_t **id**() const noexcept;

[[nodiscard]] std::string_view **name**() const noexcept;

[[nodiscard]] constexpr std::uint32_t **size**() const noexcept;

[[nodiscard]] bool **is_all**() const noexcept;

**Comparison operators**

[[nodiscard]] constexpr bool **operator<**(const *Chromosome* &other) const noexcept;

[[nodiscard]] constexpr bool **operator>**(const *Chromosome* &other) const noexcept;

[[nodiscard]] constexpr bool **operator<=**(const *Chromosome* &other) const noexcept;

[[nodiscard]] constexpr bool **operator>=**(const *Chromosome* &other) const noexcept;

[[nodiscard]] bool **operator==**(const *Chromosome* &other) const noexcept;

[[nodiscard]] bool **operator!=**(const *Chromosome* &other) const noexcept;

friend bool **operator==**(const *Chromosome* &a, std::string_view b_name) noexcept;

friend bool **operator!=**(const *Chromosome* &a, std::string_view b_name) noexcept;

friend bool **operator==**(std::string_view a_name, const *Chromosome* &b) noexcept;

friend bool **operator!=**(std::string_view a_name, const *Chromosome* &b) noexcept;

friend constexpr bool **operator<**(
    const *Chromosome* &a, std::uint32_t b_id
) noexcept;

friend constexpr bool **operator>**(
    const *Chromosome* &a, std::uint32_t b_id
) noexcept;

friend constexpr bool **operator<=**(
const *Chromosome* &a, std::uint32_t b_id
) noexcept;

friend constexpr bool **operator>=**(
const *Chromosome* &a, std::uint32_t b_id
) noexcept;

friend constexpr bool **operator==**(
const *Chromosome* &a, std::uint32_t b_id
) noexcept;

friend constexpr bool **operator!=**(
const *Chromosome* &a, std::uint32_t b_id
) noexcept;

friend constexpr bool **operator<**(
std::uint32_t a_id, const *Chromosome* &b
) noexcept;

friend constexpr bool **operator>**(
std::uint32_t a_id, const *Chromosome* &b
) noexcept;

friend constexpr bool **operator<=**(
std::uint32_t a_id, const *Chromosome* &b
) noexcept;

friend constexpr bool **operator>=**(
std::uint32_t a_id, const *Chromosome* &b
) noexcept;

friend constexpr bool **operator==**(
std::uint32_t a_id, const *Chromosome* &b
) noexcept;

friend constexpr bool **operator!=**(
std::uint32_t a_id, const *Chromosome* &b
) noexcept;

### 18.4.2 Genomic intervals

class **GenomicInterval**

Class to represent 1D genomic intervals.

This class has two main purposes:

- Storing information regarding genomic intervals
- Simplifying comparison of genomic intervals (e.g. is interval A upstream of interval B)

enum class **QUERY_TYPE**

enumerator **BED**

enumerator **UCSC**

**Constructors**

constexpr **GenomicInterval**() = default;

explicit **GenomicInterval**(const *Chromosome* &chrom_) noexcept;

**GenomicInterval**(
    const *Chromosome* &chrom_, std::uint32_t start_, std::uint32_t end
) noexcept;

**Factory methods**

[[nodiscard]] static *GenomicInterval* **parse**(
    const *Reference* &chroms, const std::string &query, Type type = Type::UCSC
);

[[nodiscard]] static *GenomicInterval* **parse_ucsc**(
    const *Reference* &chroms, std::string query
);

[[nodiscard]] static *GenomicInterval* **parse_bed**(
    const *Reference* &chroms, std::string_view query, char sep = '\t'
);

[[nodiscard]] static std::tuple<std::string, std::uint32_t, std::uint32_t> **parse**(
    const std::string &query, Type type = Type::UCSC
);

[[nodiscard]] static std::tuple<std::string, std::uint32_t, std::uint32_t> **parse_ucsc**(
    std::string buffer
);

[[nodiscard]] static std::tuple<std::string, std::uint32_t, std::uint32_t> **parse_bed**(
    std::string_view buffer, char sep = '\t'
);

**Operators**

[[nodiscard]] explicit **operator bool**() const noexcept;

[[nodiscard]] bool **operator==**(const *GenomicInterval* &other) const noexcept;

[[nodiscard]] bool **operator!=**(const *GenomicInterval* &other) const noexcept;

[[nodiscard]] bool **operator<**(const *GenomicInterval* &other) const noexcept;

[[nodiscard]] bool **operator<=**(const *GenomicInterval* &other) const noexcept;

[[nodiscard]] bool **operator>**(const *GenomicInterval* &other) const noexcept;

[[nodiscard]] bool **operator>=**(const *GenomicInterval* &other) const noexcept;

**Accessors**

[[nodiscard]] const *Chromosome* &**chrom**() const noexcept;

[[nodiscard]] constexpr std::uint32_t **start**() const noexcept;

[[nodiscard]] constexpr std::uint32_t **end**() const noexcept;

[[nodiscard]] constexpr std::uint32_t **size**() const noexcept;

### 18.4.3 Genomic bins

class **Bin**

Class modeling genomic bins.

The class is implemented as a thin wrapper around *GenomicInterval*s. The main difference between *Bin* and *GenomicInterval* objects is that in addition to genomic coordinates, the *Bin* object also store two identifiers:

- A unique identifier that can be used to refer *Bin*s in a *Reference*.
- A relative identifier that can be used to refer to *Bin*s in a *Chromosome*.

constexpr **Bin**() = default;

**Bin**(const *Chromosome* &chrom_, std::uint32_t start_, std::uint32_t end) noexcept;

**Bin**(
    std::uint64_t id_, std::uint32_t rel_id_, const *Chromosome* &chrom_, std::uint32_t start_, std::uint32_t end_
) noexcept;

explicit **Bin**(*GenomicInterval* interval) noexcept;

**Bin**(
    std::uint64_t id_, std::uint32_t rel_id_, *GenomicInterval* interval
) noexcept;

[[nodiscard]] explicit **operator bool**() const noexcept;

[[nodiscard]] bool **operator==**(const *Bin* &other) const noexcept;

[[nodiscard]] bool **operator!=**(const *Bin* &other) const noexcept;

[[nodiscard]] bool **operator<**(const *Bin* &other) const noexcept;

[[nodiscard]] bool **operator<=**(const *Bin* &other) const noexcept;

[[nodiscard]] bool **operator>**(const *Bin* &other) const noexcept;

[[nodiscard]] bool **operator>=**(const *Bin* &other) const noexcept;

[[nodiscard]] constexpr std::uint64_t **id**() const noexcept;

[[nodiscard]] constexpr std::uint32_t **rel_id**() const noexcept;

[[nodiscard]] const *GenomicInterval* &**interval**() const noexcept;

[[nodiscard]] const *Chromosome* &**chrom**() const noexcept;

[[nodiscard]] constexpr std::uint32_t **start**() const noexcept;

[[nodiscard]] constexpr std::uint32_t **end**() const noexcept;

[[nodiscard]] constexpr bool **has_null_id**() const noexcept;

### 18.4.4 Reference genome

class **Reference**

This class models the reference genome used as coordinate systems in Hi-C matrices.

*Reference* objects consist of collections of *Chromosome*s with unique IDs.

*Chromosome*s can be queried by ID or by name.

As a general rule, queries by *Chromosome* ID are more efficient than queries by name.

**Constructors**

**Reference**() = default;

template<typename **ChromosomeNameIt**, typename **ChromosomeSizeIt**>
**Reference**(
    *ChromosomeNameIt* first_chrom_name, *ChromosomeNameIt* last_chrom_name, *ChromosomeSizeIt* first_chrom_size
);

template<typename **ChromosomeIt**>
**Reference**(
    *ChromosomeIt* first_chrom, *ChromosomeIt* last_chrom
);

**Reference**(std::initializer_list<*Chromosome*> chromosomes);

#### Factory methods

[[nodiscard]] static *Reference* **from_chrom_sizes**(
    const std::filesystem::path &path_to_chrom_sizes
);

#### Operators

[[nodiscard]] bool **operator==**(const *Reference* &other) const;

[[nodiscard]] bool **operator!=**(const *Reference* &other) const;

#### Iteration

[[nodiscard]] auto **begin**() const -> const_iterator;

[[nodiscard]] auto **end**() const -> const_iterator;

[[nodiscard]] auto **cbegin**() const -> const_iterator;

[[nodiscard]] auto **cend**() const -> const_iterator;

[[nodiscard]] auto **rbegin**() const -> const_reverse_iterator;

[[nodiscard]] auto **rend**() const -> const_reverse_iterator;

[[nodiscard]] auto **rcbegin**() const -> const_reverse_iterator;

[[nodiscard]] auto **rcend**() const -> const_reverse_iterator;

#### Accessors

[[nodiscard]] bool **empty**() const noexcept;

[[nodiscard]] std::size_t **size**() const noexcept;

#### Lookup

[[nodiscard]] auto **find**(std::uint32_t id) const -> const_iterator;

[[nodiscard]] auto **find**(std::string_view chrom_name) const -> const_iterator;

[[nodiscard]] auto **find**(const *Chromosome* &chrom) const -> const_iterator;

[[nodiscard]] const *Chromosome* &**at**(std::uint32_t id) const;

[[nodiscard]] const *Chromosome* &**at**(std::string_view chrom_name) const;

[[nodiscard]] const *Chromosome* &**operator[]**(std::uint32_t id) const noexcept;

[[nodiscard]] const *Chromosome* &**operator[]**(
    std::string_view chrom_name
) const noexcept;

[[nodiscard]] bool **contains**(std::uint32_t id) const;

[[nodiscard]] bool **contains**(const *Chromosome* &chrom) const;

[[nodiscard]] bool **contains**(std::string_view chrom_name) const;

[[nodiscard]] std::uint32_t **get_id**(std::string_view chrom_name) const;

[[nodiscard]] const *Chromosome* &**longest_chromosome**() const;

[[nodiscard]] const *Chromosome* &**chromosome_with_longest_name**() const;

**Other** .. cpp:function:: [[nodiscard]] Reference remove_ALL() const; .. cpp:function:: [[nodiscard]] Reference add_ALL(std::uint32_t scaling_factor = 1) const;

### 18.4.5 Bin Table

class **BinTable**

This class models the bin table used as coordinate systems in Hi-C matrices.

The class API gives the illusion of operating over a collection of *Bin*s. In reality *BinTable*s do not store any *Bin*s. All queries are satisfied through simple arithmetic operations on the prefix sum of *Chromosome* sizes and *Bin*s are generated on the fly as needed.

This implementation has two main benefits:

- Decoupling of *BinTable* resolution and memory requirements
- Lookups in constant or linear time complexity with performance independent of resolution.

**Bin Type enum**

enum class **Type**

enumerator **fixed**

enumerator **variable**

**Constructors**

**BinTable**() = default;

**BinTable**(*Reference* chroms, std::uint32_t bin_size, std::size_t bin_offset = 0);

template<typename **ChromIt**>
**BinTable**(
    *ChromIt* first_chrom, *ChromIt* last_chrom, std::uint32_t bin_size, std::size_t bin_offset = 0
);

template<typename **ChromNameIt**, typename **ChromSizeIt**>
**BinTable**(
    *ChromNameIt* first_chrom_name, *ChromNameIt* last_chrom_name, *ChromSizeIt*
    first_chrom_size, std::uint32_t bin_size, std::size_t bin_offset = 0
);

**Operators**

[[nodiscard]] bool **operator==**(const *BinTable* &other) const;

[[nodiscard]] bool **operator!=**(const *BinTable* &other) const;

**Accessors**

[[nodiscard]] std::size_t **size**() const noexcept;

[[nodiscard]] bool **empty**() const noexcept;

[[nodiscard]] std::size_t **num_chromosomes**() const noexcept;

[[nodiscard]] constexpr std::uint32_t **resolution**() const noexcept;

[[nodiscard]] constexpr const *Reference* &**chromosomes**() const noexcept;

[[nodiscard]] constexpr auto **type**() const noexcept -> *Type*;

[[nodiscard]] constexpr const std::vector<std::uint64_t> &**num_bin_prefix_sum**(
) const noexcept;

**Iteration**

[[nodiscard]] auto **begin**() const -> iterator;

[[nodiscard]] auto **end**() const -> iterator;

[[nodiscard]] auto **cbegin**() const -> iterator;

[[nodiscard]] auto **cend**() const -> iterator;

**Slicing**

[[nodiscard]] *BinTable* **subset**(const *Chromosome* &chrom) const;

[[nodiscard]] *BinTable* **subset**(std::string_view chrom_name) const;

[[nodiscard]] *BinTable* **subset**(std::uint32_t chrom_id) const;

**Lookup**

[[nodiscard]] auto **find_overlap**(
    const *GenomicInterval* &query
) const -> std::pair<*BinTable*::iterator, *BinTable*::iterator>;

[[nodiscard]] auto **find_overlap**(
    const *Chromosome* &chrom, std::uint32_t start, std::uint32_t end
) const -> std::pair<*BinTable*::iterator, *BinTable*::iterator>;

[[nodiscard]] auto **find_overlap**(
    std::string_view chrom_name, std::uint32_t start, std::uint32_t end
) const -> std::pair<*BinTable*::iterator, *BinTable*::iterator>;

[[nodiscard]] auto **find_overlap**(
    std::uint32_t chrom_id, std::uint32_t start, std::uint32_t end
) const -> std::pair<*BinTable*::iterator, *BinTable*::iterator>;

[[nodiscard]] std::pair<*Bin*, *Bin*> **at**(const *GenomicInterval* &gi) const;

[[nodiscard]] std::pair<std::uint64_t, std::uint64_t> **map_to_bin_ids**(
    const *GenomicInterval* &gi
) const;

Query bins by genomic interval.

[[nodiscard]] *Bin* **at**(std::uint64_t bin_id) const;

[[nodiscard]] *Bin* **at**(const *Chromosome* &chrom, std::uint32_t pos = 0) const;

[[nodiscard]] *Bin* **at**(std::string_view chrom_name, std::uint32_t pos = 0) const;

[[nodiscard]] *Bin* **at**(std::uint32_t chrom_id, std::uint32_t pos) const;

[[nodiscard]] *Bin* **at_hint**(std::uint64_t bin_id, const *Chromosome* &chrom) const;

Query by bin identifier.

[[nodiscard]] std::uint64_t **map_to_bin_id**(
    const *Chromosome* &chrom, std::uint32_t pos
) const;

[[nodiscard]] std::uint64_t **map_to_bin_id**(
    std::string_view chrom_name, std::uint32_t pos
) const;

[[nodiscard]] std::uint64_t **map_to_bin_id**(
    std::uint32_t chrom_id, std::uint32_t pos
) const;

Query by genomic coordinates

**Others**

[[nodiscard]] BinTableConcrete **concretize**() const;

### 18.4.6 Pixels

template<typename **N**>
class **ThinPixel**

Struct to model a genomic pixel using as little memory as possible.

**Member variables**

static constexpr auto **null_id** = std::numeric_limits<std::uint64_t>::max();

std::uint64_t **bin1_id**{*null_id*};

std::uint64_t **bin2_id**{*null_id*};

*N* **count**{};

**Factory methods**

static auto **from_coo**(std::string_view line) -> *ThinPixel*;

static auto **from_coo**(const *BinTable* &bins, std::string_view line) -> *ThinPixel*;

**Operators**

[[nodiscard]] explicit **operator bool**() const noexcept;

[[nodiscard]] bool **empty**() const noexcept;

[[nodiscard]] bool **operator==**(const *ThinPixel* &other) const noexcept;

[[nodiscard]] bool **operator!=**(const *ThinPixel* &other) const noexcept;

[[nodiscard]] bool **operator<**(const *ThinPixel* &other) const noexcept;

[[nodiscard]] bool **operator<=**(const *ThinPixel* &other) const noexcept;

[[nodiscard]] bool **operator>**(const *ThinPixel* &other) const noexcept;

[[nodiscard]] bool **operator>=**(const *ThinPixel* &other) const noexcept;

class **PixelCoordinates**;

Struct to model 2D genomic coordinates using a pair of *Bin*s.

**Member variables**

*Bin* **bin1**

*Bin* **bin2**

**Constructors**

**PixelCoordinates**() = default;

**PixelCoordinates**(*Bin* bin1_, *Bin* bin2_) noexcept;

explicit **PixelCoordinates**(std::pair<*Bin*, *Bin*> bins) noexcept;

explicit **PixelCoordinates**(*Bin* bin) noexcept;

**Operators**

[[nodiscard]] explicit **operator bool**() const noexcept;

[[nodiscard]] bool **empty**() const noexcept;

[[nodiscard]] bool **operator==**(const *PixelCoordinates* &other) const noexcept;

[[nodiscard]] bool **operator!=**(const *PixelCoordinates* &other) const noexcept;

[[nodiscard]] bool **operator<**(const *PixelCoordinates* &other) const noexcept;

[[nodiscard]] bool **operator<=**(const *PixelCoordinates* &other) const noexcept;

[[nodiscard]] bool **operator>**(const *PixelCoordinates* &other) const noexcept;

[[nodiscard]] bool **operator>=**(const *PixelCoordinates* &other) const noexcept;

**Accessors**

[[nodiscard]] bool **is_intra**() const noexcept;

template<typename **N**>
class **Pixel**

Struct to model genomic pixels as interaction counts associated to a pair of genomic *Bin*s.

The main difference between *ThinPixel* and *Pixel* objects, is that the latter possessesall the knowledge required to map interactions to genomic coordinates, not just bin IDs.

**Member variables**

*PixelCoordinates* **coords**{};

*N* **count**{};

**Constructors**

**Pixel**() = default;

explicit **Pixel**(*Bin* bin, *N* count_ = 0) noexcept;

**Pixel**(*Bin* bin1_, *Bin* bin2_, *N* count_ = 0) noexcept;

explicit **Pixel**(*PixelCoordinates* coords_, *N* count_ = 0) noexcept;

**Pixel**(
    const *Chromosome* &chrom, std::uint32_t start, std::uint32_t end, *N* count_ = 0
) noexcept;

**Pixel**(
    const *Chromosome* &chrom1, std::uint32_t start1, std::uint32_t end1, const *Chromosome* &chrom2, std::uint32_t start2, std::uint32_t end2, *N* count_ = 0
) noexcept;

**Pixel**(
    const *BinTable* &bins, std::uint64_t bin1_id, std::uint64_t bin2_id, *N* count_ = 0
);

**Pixel**(const *BinTable* &bins, std::uint64_t bin_id, *N* count_ = 0);

**Pixel**(const *BinTable* &bins, const *ThinPixel*<*N*> &p);

**Factory methods**

static auto **from_coo**(const *BinTable* &bins, std::string_view line) -> *Pixel*;

static auto **from_bg2**(const *BinTable* &bins, std::string_view line) -> *Pixel*;

static auto **from_validpair**(
    const *BinTable* &bins, std::string_view line
) -> *Pixel*;

static auto **from_4dn_pairs**(
    const *BinTable* &bins, std::string_view line
) -> *Pixel*;

**Operators**

[[nodiscard]] explicit **operator bool**() const noexcept;

[[nodiscard]] bool **operator==**(const *Pixel*<*N*> &other) const noexcept;

[[nodiscard]] bool **operator!=**(const *Pixel*<*N*> &other) const noexcept;

[[nodiscard]] bool **operator<**(const *Pixel*<*N*> &other) const noexcept;

[[nodiscard]] bool **operator<=**(const *Pixel*<*N*> &other) const noexcept;

[[nodiscard]] bool **operator>**(const *Pixel*<*N*> &other) const noexcept;

[[nodiscard]] bool **operator>=**(const *Pixel*<*N*> &other) const noexcept;

**Conversion**

[[nodiscard]] *ThinPixel*<*N*> **to_thin**() const noexcept;

## 18.5 Pixel transformers

The transformer library provides a set of common algorithms used to manipulate streams of pixels. Classes in defined in this library take a pair of pixel iterators or *PixelSelector*s directly and transform and/or aggregate them in different ways.

### 18.5.1 Common types

enum class **QuerySpan**

enumerator **lower_triangle**

enumerator **upper_triangle**

enumerator **full**

### 18.5.2 Coarsening pixels

template<typename **PixelIt**>
class **CoarsenPixels**

Class used to coarsen pixels read from a pair of pixel iterators. The underlying sequence of pixels are expected to be sorted by their genomic coordinates. Coarsening is performed in a streaming fashion, minimizing the number of pixels that are kept into memory at any given time.

**CoarsenPixels**(
*PixelIt* first_pixel, *PixelIt* last_pixel, std::shared_ptr<const *BinTable*> source_bins, std::size_t factor
);

Constructor for *CoarsenPixels* class. `first_pixel` and `last_pixels` should be a pair of iterators pointing to the stream of pixels to be coarsened. `source_bins` is a shared pointer to the bin table to which `first_pixel` and `last_pixel` refer to. `factor` should be an integer value greater than 1, and is used to determine the properties of the `target_bins` *BinTable* used for coarsening.

**Accessors**

[[nodiscard]] const *BinTable* &**src_bins**() const noexcept;

[[nodiscard]] const *BinTable* &**dest_bins**() const noexcept;

[[nodiscard]] std::shared_ptr<const *BinTable*> **src_bins_ptr**() const noexcept;

[[nodiscard]] std::shared_ptr<const *BinTable*> **dest_bins_ptr**() const noexcept;

*BinTable* accessors.

**Iteration**

[[nodiscard]] **begin**() const -> iterator;

[[nodiscard]] **end**() const -> iterator;

[[nodiscard]] **cbegin**() const -> iterator;

[[nodiscard]] **cend**() const -> iterator;

Return an InputIterator to traverse the coarsened pixels.

**Others**

[[nodiscard]] auto **read_all**() const -> std::vector<*ThinPixel*<N>>;

### 18.5.3 Selecting pixels overlapping with a band around the matrix diagonal

template<typename **PixelIt**>
class **DiagonalBand**

Class used to select pixels overlapping with a band around the matrix diagonal.

**DiagonalBand**(*PixelIt* first_pixel, *PixelIt* last_pixel, std::uint64_t num_bins);

Constructor for *DiagonalBand* class. `first_pixel` and `last_pixels` should be a pair of iterators pointing to the stream of pixels to be processed. `num_bins` should correspond to the width of the band around the matrix diagonal. As all filtering operations are performed based on bin IDs, this transformer is unsuitable for processing pixels originating from files with bin tables of non-uniform size.

**Iteration**

[[nodiscard]] **begin**() const -> iterator;

[[nodiscard]] **end**() const -> iterator;

[[nodiscard]] **cbegin**() const -> iterator;

[[nodiscard]] **cend**() const -> iterator;

Return an InputIterator to traverse the pixels after filtering.

**Others***

[[nodiscard]] auto **read_all**() const -> std::vector<*Pixel*<N>>;

### 18.5.4 Transforming COO pixels to BG2 pixels

template<typename **PixelIt**>
class **JoinGenomicCoords**

Class used to join genomic coordinates onto COO pixels, effectively transforming *ThinPixel*s into *Pixel*s.

**JoinGenomicCoords**(
*PixelIt* first_pixel, *PixelIt* last_pixel, std::shared_ptr<const *BinTable*> bins
);

Constructor for *JoinGenomicCoords* class. `first_pixel` and `last_pixels` should be a pair of iterators pointing to the stream of pixels to be processed. `bins` is a shared pointer to the bin table to which `first_pixel` and `last_pixel` refer to.

**Iteration**

[[nodiscard]] **begin**() const -> iterator;

[[nodiscard]] **end**() const -> iterator;

[[nodiscard]] **cbegin**() const -> iterator;

[[nodiscard]] **cend**() const -> iterator;

Return an InputIterator to traverse the *Pixel*s.

**Others***

[[nodiscard]] auto **read_all**() const -> std::vector<*Pixel*<N>>;

### 18.5.5 Merging streams of pre-sorted pixels

template<typename **PixelIt**>
class **PixelMerger**

Class used to merge streams of pre-sorted pixels, yielding a sequence of unique pixels sorted by their genomic coordinates. Merging is performed in a streaming fashion, minimizing the number of pixels that are kept into memory at any given time.

Duplicate pixels are aggregated by summing their corresponding interactions. Pixel merging also affects duplicate pixels coming from the same stream.

**PixelMerger**(std::vector<*PixelIt*> head, std::vector<*PixelIt*> tail);

template<typename **ItOfPixelIt**>
**PixelMerger**(
    *ItOfPixelIt* first_head, *ItOfPixelIt* last_head, *ItOfPixelIt* first_tail
);

Constructors taking either two vectors of InputIterators or pairs of iterators to InputIterators.

The `head` and `tail` vectors should contain the iterators pointing to the beginning and end of *ThinPixel* streams, respectively.

**Iteration**

[[nodiscard]] auto **begin**() const -> iterator;

[[nodiscard]] auto **end**() const noexcept -> iterator;

Return an InputIterator to traverse the stream *ThinPixel*s after merging.

**Others**

[[nodiscard]] auto **read_all**() const -> std::vector<PixelT>;

### 18.5.6 Computing common statistics

template<typename **PixelIt**>
[[nodiscard]] double **avg**(
    *PixelIt* first, *PixelIt* last
);

template<typename **PixelIt**, typename **N**>
[[nodiscard]] *N* **max**(
    *PixelIt* first, *PixelIt* last
);

template<typename **PixelIt**>
[[nodiscard]] std::size_t **nnz**(
    *PixelIt* first, *PixelIt* last
);

template<typename **PixelIt**, typename **N**>
[[nodiscard]] *N* **sum**(
    *PixelIt* first, *PixelIt* last
);

### 18.5.7 Converting streams of pixels to Arrow Tables

enum class **DataFrameFormat**

enumerator **COO**

enumerator **BG2**

template<typename **PixelIt**>
class **ToDataFrame**

**ToDataFrame**(
    *PixelIt* first_pixel, *PixelIt* last_pixel, *DataFrameFormat* format = *DataFrameFormat*::*COO*, std::shared_ptr<const *BinTable*> bins = nullptr, *QuerySpan* span = *QuerySpan*::*upper_triangle*, bool include_bin_ids = false, bool mirror_pixels = true, std::size_t chunk_size = 256'000, std::optional<std::uint64_t> diagonal_band_width = {}
);

**ToDataFrame**(
*PixelIt* first_pixel, *PixelIt* last_pixel, std::optional<*PixelCoordinates*> coord1_, std::optional<*PixelCoordinates*> coord2_ = {}, *DataFrameFormat* format = *DataFrameFormat*::*COO*, std::shared_ptr<const *BinTable*> bins = nullptr, *QuerySpan* span = *QuerySpan*::*upper_triangle*, bool include_bin_ids = false, bool mirror_pixels = true, std::size_t chunk_size = 256'000, std::optional<std::uint64_t> diagonal_band_width = {}
);

template<typename **PixelSelector**>
**ToDataFrame**(
const *PixelSelector* &sel, *PixelIt* it, *DataFrameFormat* format = *DataFrameFormat*::*COO*, std::shared_ptr<const *BinTable*> bins = nullptr, *QuerySpan* span = *QuerySpan*::*upper_triangle*, bool include_bin_ids = false, std::size_t chunk_size = 256'000, std::optional<std::uint64_t> diagonal_band_width = {}
);

Construct an instance of a *ToDataFrame* converter given a stream of pixels delimited by `first_pixel` and `last_pixel`, a DataFrame `format` and a *BinTable*. The underlying sequence of pixels are expected to be sorted by their genomic coordinates.

The optional argument `span` determines whether the resulting `arrow::Table` should contain interactions spanning the upper/lower-triangle or all interactions (regardless of whether they are located above or below the genome-wide matrix diagonal). It should be noted that queries spanning the the full-matrix or the lower-triangle are always more expensive because they involve an additional step where pixels are sorted by their genomic coordinates.

When provided, the `diagonal_band_width` argument has the same semantics as the `num_bins` argument from the *DiagonalBand* constructor.

When fetching interactions with `span=full` from the cis portion of interaction maps using one of the overloads taking a pair of pixel iterators, users should provide a pair of genomic coordinates to ensure that, if necessary, interactions are correctly mirrored.

[[nodiscard]] std::shared_ptr<arrow::Table> **operator()**();

Convert the stream of pixels into an `arrow::Table`.

### 18.5.8 Converting streams of pixels to Eigen Dense Matrices

template<typename **N**, typename **PixelSelector**>
class **ToDenseMatrix**

using **MatrixT** = Eigen::Matrix<*N*, Eigen::Dynamic, Eigen::Dynamic, Eigen::RowMajor>;

**ToDenseMatrix**(
*PixelSelector* sel, *N* n, *QuerySpan* span = *QuerySpan*::*full*, std::optional<std::uint64_t> diagonal_band_width = {}
);

**ToDenseMatrix**(
std::shared_ptr<const *PixelSelector*> sel, *N* n, *QuerySpan* span = *QuerySpan*::*full*, std::optional<std::uint64_t> diagonal_band_width = {}
);

Construct an instance of a *ToDenseMatrix* converter given a *PixelSelector* object and a count type `n`.

The optional argument `span` determines whether the resulting matrix should contain interactions spanning the upper/lower-triangle or all interactions (regardless of whether they are located above or below the genome-wide matrix diagonal). Note that attempting to fetch trans-interactions with `span=QuerySpan::lower_triangle` will result in an exception being thrown. If you need to fetch trans-interactions from the lower-triangle, consider exchanging the range arguments used to fetch interactions, then transpose the resulting matrix.

When provided, the `diagonal_band_width` argument has the same semantics as the `num_bins` argument from the *DiagonalBand* constructor.

[[nodiscard]] auto **operator()**() -> *MatrixT*;

Convert the stream of pixels into an *MatrixT*.

### 18.5.9 Converting streams of pixels to Eigen Sparse Matrices

template<typename **N**, typename **PixelSelector**>
class **ToSparseMatrix**

using **MatrixT** = Eigen::SparseMatrix<*N*, Eigen::RowMajor>;

**ToSparseMatrix**(
*PixelSelector* sel, *N* n, *QuerySpan* span = *QuerySpan*::*upper_triangle*, bool minimize_memory_usage = false, std::optional<std::uint64_t> diagonal_band_width = {}
);

**ToSparseMatrix**(
std::shared_ptr<const *PixelSelector*> sel, *N* n, *QuerySpan* span = *QuerySpan*::*full*, bool minimize_memory_usage = false, std::optional<std::uint64_t> diagonal_band_width = {}
);

Construct an instance of a *ToSparseMatrix* converter given a *PixelSelector* object and a count type `n`.

The optional argument `span` determines whether the resulting matrix should contain interactions spanning the upper/lower-triangle or all interactions (regardless of whether they are located above or below the genome-wide matrix diagonal). Note that attempting to fetch trans-interactions with `span=QuerySpan::lower_triangle` will result in an exception being thrown. If you need to fetch trans-interactions from the lower-triangle, consider exchanging the range arguments used to fetch interactions, then transpose the resulting matrix.

When `minimize_memory_usage=true`, hictk will minimize memory usage by doing two passes over the queried pixels: one to calculate the exact number of entries to allocate for each row in the matrix, and the second pass to fill values in the matrix. This is usually slower than the default strategy, which traverses the data only once (but may overall require more memory than what is strictly needed). It should be noted that matrices are always compressed before being returned. Thus, the memory footprint of the matrices returned by *ToSparseMatrix::operator()()* will be the same regardless of the fill strategy.

When provided, the `diagonal_band_width` argument has the same semantics as the `num_bins` argument from the *DiagonalBand* constructor.

[[nodiscard]] auto **operator()**() -> *MatrixT*;

Convert the stream of pixels into an *MatrixT*.

CHAPTER

# NINETEEN

# TELEMETRY

Starting with version v2.1.0 of hictk, we introduced support for telemetry collection.

This only applies when hictk is invoked from the CLI (i.e., not to libhictk).

This page outlines what information we are collecting and why. Furthermore, we provide instructions on how telemetry collection can be disabled at execution and compile time.

## 19.1 What information is being collected

`hictk` is instrumented to collect general information about `hictk` itself and the system where it is being run.

We do not collect any sensitive information that could be used to identify our users, the machine or environment where `hictk` is being run, the datasets processed by `hictk`, or the parameters used to run `hictk`.

This is the data we are collecting:

- Information on how `hictk` was compiled (i.e., compiler name, version, and build type).
- Information on the system where `hictk` is being run (i.e., operating system and processor architecture).
- Information about `hictk` itself (i.e., version of `hictk` and its third-party version of dependencies).
- The continent, country, and region names, as well as the time zone where `hictk` was launched. This information is inferred from the IP address used to submit the telemetry (the IP address itself is not part of the telemetry data we collect and it never stored by our servers).
- How `hictk` is being invoked (i.e., the subcommand, the hash of the command line arguments used to invoke `hictk`, and the input/output format(s) where applicable).
- Information about `hictk` execution (i.e., when it was launched, how long the command took to finish, and whether the command terminated with an error).
- For the `hictk dump` subcommand, we are also collecting the name of the table that is being dumped (e.g., pixels or chroms).

This is an example of the telemetry collected when running `hictk dump`:

```
name          : subcommand.dump
trace_id      : a51ce70f8aff91281eb70332c5eb775b
span_id       : 869ec9f57d2e170e
tracestate    :
parent_span_id: 0000000000000000
start         : 1758032386754347017
duration      : 151590958
description   :
span kind     : Internal
status        : Ok
attributes    :
      param.table: pixels
      meta.input-format: mcool
```

(continues on next page)

(continued from previous page)

```
      meta.argv-sha3-256:␣
↪6840f26c9293a323369ea6d571a48b8a49934e76e6e1a645f224caf14663c
      schema: 1
events        :
links         :
resources     :
      build.compiler.name: Clang
      build.compiler.version: 20.1.8
      build.dependencies.boost.version: 1.88.0
      build.dependencies.bshoshany-thread-pool.version: 5.0.0
      build.dependencies.cli11.version: 2.5.0
      build.dependencies.concurrentqueue.version: 1.0.4
      build.dependencies.fast_float.version: 8.0.2
      build.dependencies.fmt.version: 11.2.0
      build.dependencies.hdf5.version: 1.14.6
      build.dependencies.highfive.version: 2.10.0
      build.dependencies.libarchive.version: 3.8.1
      build.dependencies.libdeflate.version: 1.23
      build.dependencies.nlohmann_json.version: 3.12.0
      build.dependencies.opentelemetry-cpp.version: 1.21.0
      build.dependencies.parallel-hashmap.version: 2.0.0
      build.dependencies.readerwriterqueue.version: 1.0.6
      build.dependencies.span-lite.version: 0.11.0
      build.dependencies.spdlog.version: 1.15.3
      build.dependencies.tomlplusplus.version: 3.4.0
      build.dependencies.zstd.version: 1.5.7
      build.type: Release
geo.continent_name: Europe
geo.country_name: Norway
geo.region_name: Oslo County
geo.timezone: Europe/Oslo
      host.arch: x86_64
      os.type: Linux
      os.version: 6.16.3-200.fc42.x86_64
      service.name: hictk
      service.version: 2.1.5
      telemetry.sdk.language: cpp
      telemetry.sdk.name: opentelemetry
      telemetry.sdk.version: 1.21.0
instr-lib     : hictk
```

## 19.2 Why are we collecting this information?

There are two main motivations behind our decision to start collecting telemetry data:

1. To get an idea of how big our user base is: this will help us, among other things, to secure funding to maintain `hictk` in the future.
2. To better understand which of the functionalities offered by `hictk` are most used by our users: we intend to use this information to help us decide which features we should focus our development efforts on.

## 19.3 How is telemetry information processed and stored

Telemetry is sent to an OpenTelemetry collector running on a virtual server hosted on the Norwegian Research and Education Cloud (NREC).

The virtual server and collector are managed by us, and traffic between `hictk` and the collector is encrypted.

The collector processes incoming data continuously and forwards it to a dashboard for data analytics and a backup solution (both services are hosted in Europe). Communication between the collector, dashboard, and backup site is also encrypted. Data stored by the dashboard and backup site is encrypted at rest.

The analytics dashboard keeps telemetry data for up to 60 days, while the backup site is currently set up to store telemetry data indefinitely (although this may change in the future).

## 19.4 How to disable telemetry collection

We provide two mechanisms to disable telemetry.

1. Disabling telemetry at runtime: simply define the `HICTK_NO_TELEMETRY` environment variable before launching `hictk` (e.g., `HICTK_NO_TELEMETRY=1 hictk dump matrix.cool`)
2. Disabling telemetry at compile time: this only applies if you are building hictk from source as outlined in *Installation (source)*.

   To completely disable telemetry support at compile time pass `-DHICTK_ENABLE_TELEMETRY=OFF` when configuring the project with CMake.

   When `HICTK_ENABLE_TELEMETRY` is set to `OFF`, classes and functions used to collect information using OpenTelemetry are replaced with alternative implementations that do nothing. Furthermore, the OpenTelemetry library is not linked to the `hictk` binary, meaning that no code involved in the collection of telemetry information is contained in or loaded by the `hictk` binary.

## 19.5 Where can I find the code used for telemetry collection?

All code concerning telemetry collection is defined in the library under src/hictk/telemetry.

The link flags and pre-processor macros toggling telemetry support at compile time are defined in files src/hictk/CMakeLists.txt and src/hictk/telemetry/CMakeLists.txt.

# INDEX

## H