hictk Documentation

Roberto Rossini

May 13, 2024

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).

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 = {Roberto Rossini and Jonas Paulsen},
        title = {hictk: blazing fast toolkit to work with .hic and .cool files},
        elocation-id = {2023.11.26.568707},
        year = {2023},
        doi = {10.1101/2023.11.26.568707},
        publisher = {Cold Spring Harbor Laboratory},
        URL = {https://www.biorxiv.org/content/early/2023/11/27/2023.11.26.568707},
        eprint = {https://www.biorxiv.org/content/early/2023/11/27/2023.11.26.568707.
↪full.pdf},
        journal = {bioRxiv}
}
```

If you use `hictk convert` to convert .[m]cool files to .hic format you should also cite JuicerTools or HiCTools.

CHAPTER

ONE

INSTALLATION

1.1 Conda (bioconda)

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-v0.0.12-bioconda
```

If you are trying to install hictk on a Mac with an M chip, the above command may fail due to conda not being able to find a package for hictk. You can workaround the above issue by prefixing conda commands with `CONDA_SUBDIR=osx-64`. Note that this will make hictk quite a bit slower, as the installed binary will be executed through Rosetta. If performance is important, please consider *compiling hictk from source* or using containers (see below).

1.2 Containers (Docker or Singularity/Apptainer)

First, make sure you follow 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 just 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.

1.2.1 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:0.0.12 --help

# Using Singularity/Apptainer
user@dev:/tmp$ singularity run ghcr.io/paulsengroup/hictk:0.0.12 --help

Blazing fast tools to work with .hic and .cool files.
Usage: /usr/local/bin/hictk [OPTIONS] SUBCOMMAND

Options:
  -h,--help                   Print this help message and exit
  -V,--version                Display program version information and exit

Subcommands:
  convert                     Convert HiC matrices to a different format.
  dump                        Dump data from .hic and Cooler files to stdout.
  load                        Build .cool files from interactions in various text␣
 ↪formats.
  merge                       Merge coolers.
  validate                    Validate .hic and Cooler files.
  zoomify                     Convert single-resolution Cooler file to multi-
 ↪resolution by coarsening.
```

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

1.3 Installing from source

Please refer to hictk's *build instructions*.

CHAPTER

TWO

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.

2.1 Build instructions

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

2.1.1 Build requirements

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

- GCC 8+
- Clang 8+
- Apple-Clang 10.0+

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 of course free to install build dependencies in any way you want.

```
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's 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 compiler toolchain. It is usually a good idea to explicitly set CC and CXX
CC=gcc CXX=g++ conan profile detect --force
```

2.1.2 Getting the source code

Download from the Release page (recommended).

```
mkdir /tmp/hictk
curl -L 'https://github.com/paulsengroup/hictk/archive/refs/tags/v0.0.6.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 v0.0.12  # Skip this step if you want to build the latest commit from␣
↪main
```

2.1.3 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

# Install/build dependencies with Conan
conan install --build=missing \
              -pr default \
              -s build_type=Release \
              -s compiler.cppstd=17 \
              --output-folder=./build/ \
              .

# This may take a while, as CMake will run Conan to build hictk dependencies.
# Do not pass -G Ninja if you want CMake to use make instead of ninja
cmake -DCMAKE_BUILD_TYPE=Release \
      -DCMAKE_PREFIX_PATH="$PWD/build" \
      -DHICTK_ENABLE_TESTING=ON \
      -DHICTK_BUILD_TOOLS=ON \
      -G Ninja \
      -S /tmp/hictk \
      -B /tmp/hictk/build

cmake --build /tmp/hictk/build
```

To override the default compiler used by CMake, pass the following arguments to the first CMake command: `-DCMAKE_C_COMPILER=path/to/cc -DCMAKE_CXX_COMPILER=path/to/c++`

We highly recommend using the same compiler when running Conan and CMake.

2.2 Running automated tests

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

2.2.1 Unit tests

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

cd /tmp/hictk
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/ ...
...
63/70 Test #21: Cooler: init files - SHORT ....................................  ␣
 ↪Passed    0.02 sec
64/70 Test #57: HiC: pixel selector fetch (observed NONE BP 10000) - LONG ........  ␣
 ↪Passed    1.53 sec
65/70 Test  #5: Cooler: index validation - SHORT ..............................  ␣
 ↪Passed    3.83 sec
66/70 Test #17: Cooler: index validation - SHORT ..............................  ␣
 ↪Passed    3.62 sec
67/70 Test #37: Cooler: utils merge - LONG .....................................  ␣
 ↪Passed    4.35 sec
68/70 Test #67: Transformers (cooler) - SHORT ..................................  ␣
 ↪Passed    4.11 sec
69/70 Test #36: Cooler: dataset random iteration - MEDIUM ......................  ␣
 ↪Passed    5.50 sec
70/70 Test #40: Cooler: dataset large read/write - LONG ........................  ␣
 ↪Passed   11.47 sec

100% tests passed, 0 tests failed out of 70

Total Test time (real) =  12.03 sec
```

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

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

1. Make sure you are running `ctest` from the root of the source tree (`/tmp/hictk` if you are following the instructions).
2. Make sure 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.xz` 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/10522583/files/hictk_test_data.tar.xz?download=1'␣
↪| tar -xJf -

# 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, feel free to start discussion asking for help.

2.2.2 Integration tests

The integration test scripts depend on the following tools:

- cooler>=0.9
- java
- juicer_tools or hic_tools
- xz
- common UNIX shell commands

cooler can be installed using pip:

```
/tmp/venv/bin/pip3 install 'cooler>=0.9'
```

juicer_tools and hic_tools do not need to be installed, downloading the JAR file is enough:

```
curl -L 'https://github.com/aidenlab/HiCTools/releases/download/v3.30.00/hic_tools.3.
↪30.00.jar' -o /tmp/hictk/hic_tools.jar
```

If not already installed, `xz` can usually be installed with your system package manager (on some Linux distributions the relevant package is called `xz-utils`).

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

cd /tmp/hictk
```

(continues on next page)

(continued from previous page)

```
# hictk convert
test/scripts/hictk_convert_cool2hic.sh build/src/hictk/hictk juicer_tools.jar
test/scripts/hictk_convert_hic2cool.sh build/src/hictk/hictk

# hictk dump
test/scripts/hictk_dump_balanced.sh build/src/hictk/hictk
test/scripts/hictk_dump_bins.sh build/src/hictk/hictk
test/scripts/hictk_dump_chroms.sh build/src/hictk/hictk
test/scripts/hictk_dump_cis.sh build/src/hictk/hictk
test/scripts/hictk_dump_gw.sh build/src/hictk/hictk
test/scripts/hictk_dump_trans.sh build/src/hictk/hictk

# hictk load (sorted)
test/scripts/hictk_load_4dn.sh build/src/hictk/hictk sorted
test/scripts/hictk_load_bg2.sh build/src/hictk/hictk sorted
test/scripts/hictk_load_coo.sh build/src/hictk/hictk sorted

# hictk load (unsorted)
test/scripts/hictk_load_4dn.sh build/src/hictk/hictk unsorted
test/scripts/hictk_load_bg2.sh build/src/hictk/hictk unsorted
test/scripts/hictk_load_coo.sh build/src/hictk/hictk unsorted

# hictk merge
test/scripts/hictk_merge.sh build/src/hictk/hictk

# hictk validate
test/scripts/hictk_validate.sh build/src/hictk/hictk

# hictk zoomify
test/scripts/hictk_zoomify.sh build/src/hictk/hictk
```

2.3 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 runtime path of "/usr/local/bin/hictk" to ""
-- Up-to-date: /usr/local/share/licenses/hictk/LICENSE
...

# 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 runtime path of "/home/user/.local/bin/hictk" to ""
-- Up-to-date: /home/user/.local/share/licenses/hictk/LICENSE
...
```

2.4 Cleaning build artifacts

After successfully compiling hictk the following folders 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 also delete Conan's folder `~/.conan2/`

CHAPTER

THREE

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-v0.0.12
```

3.1 Command line interface

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

3.1.1 Verifying file integrity

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

hictk validate interactions.hic
```

For more detailed examples refer to *File validation*

3.1.2 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*

3.1.3 Other operations

- *Format conversion*
- *Creating .cool and .hic files*
- *Converting single-resolution files to multi-resolution*
- *Balancing Hi-C matrices*

3.2 API

Refer to *Quickstart (API)*.

CHAPTER

FOUR

QUICKSTART (API)

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

4.1 Installing libhictk

4.1.1 Installing using Conan

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

```
[requires]
hictk/0.0.12

[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.

4.1.2 Installing using CMake FetchContent

Before beginning, make sure all of hictk dependencies have been installed. Refer to conanfile.txt 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)
cmake_policy(VERSION 3.25...3.27)

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         v0.0.12
  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
```

4.1.3 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`.

4.1.4 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>
```

(continues on next page)

(continued from previous page)

```cpp
#include <variant>

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

FIVE

DOWNLOADING TEST DATASETS

Test dataset 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.xz                    \
                   -C data --strip-components=3                  \
                   test/data/hic/4DNFIZ1ZVXC8.hic9               \
                   test/data/integration_tests/4DNFIZ1ZVXC8.mcool \
                   test/data/integration_tests/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 128M Jun  8 19:42 4DNFIZ1ZVXC8.hic9
-rw-------  1 dev dev 133M Jul  7 16:29 4DNFIZ1ZVXC8.mcool
```

CHAPTER

SIX

FILE VALIDATION

6.1 Why is this needed?

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

6.1.1 Cooler index corruption

To make a long story short, 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 (possibly all) .mcool files released by 4DNucleome:

6.2 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.

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

```
# Validate a .hic file
user@dev:/tmp$ hictk validate test/data/hic/4DNFIZ1ZVXC8.hic8
### SUCCESS: "test/data/hic/4DNFIZ1ZVXC8.hic8" is a valid .hic file.

# Validate a .cool file
user@dev:/tmp$ hictk validate test/data/integration_tests/4DNFIZ1ZVXC8.mcool
uri="test/data/integration_tests/4DNFIZ1ZVXC8.mcool::/resolutions/2500000"
is_hdf5=true
unable_to_open_file=false
file_was_properly_closed=true
missing_or_invalid_format_attr=false
missing_or_invalid_bin_type_attr=false
missing_groups=[]
is_valid_cooler=true
index_is_valid=not_checked
### SUCCESS: "test/data/integration_tests/4DNFIZ1ZVXC8.mcool::/resolutions/2500000"␣
 ↪is a valid Cooler.
uri="test/data/integration_tests/4DNFIZ1ZVXC8.mcool::/resolutions/1000000"
is_hdf5=true
unable_to_open_file=false
file_was_properly_closed=true
missing_or_invalid_format_attr=false
missing_or_invalid_bin_type_attr=false
missing_groups=[]
is_valid_cooler=true
index_is_valid=not_checked
### SUCCESS: "test/data/integration_tests/4DNFIZ1ZVXC8.mcool::/resolutions/1000000"␣
 ↪is a valid Cooler.
...
uri="test/data/integration_tests/4DNFIZ1ZVXC8.mcool::/resolutions/1000"
is_hdf5=true
unable_to_open_file=false
file_was_properly_closed=true
missing_or_invalid_format_attr=false
missing_or_invalid_bin_type_attr=false
missing_groups=[]
is_valid_cooler=true
index_is_valid=not_checked
### SUCCESS: "test/data/integration_tests/4DNFIZ1ZVXC8.mcool::/resolutions/1000" is a␣
 ↪valid Cooler.
```

The quick check will not detect Cooler files with corrupted index, as this requires the `--validate-index` option:

```
user@dev:/tmp$ hictk validate --validate-index 4DNFI9GMP2J8.mcool::/resolutions/
 ↪1000000
uri="4DNFI9GMP2J8.mcool::/resolutions/1000000"
is_hdf5=true
unable_to_open_file=false
file_was_properly_closed=true
missing_or_invalid_format_attr=false
missing_or_invalid_bin_type_attr=false
```

(continues on next page)

(continued from previous page)

```
missing_groups=[]
is_valid_cooler=true
index_is_valid=false
### FAILURE: "4DNFI9GMP2J8.mcool::/resolutions/1000000" is not a valid Cooler.
```

6.3 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`.

CHAPTER

SEVEN

FORMAT CONVERSION

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

7.1 Converting from .hic to .[m]cool

Converting from .hic to .cool or .mcool formats consists of 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 known normalization vectors (currently these are VC, VC_SQRT, KR, and SCALE)

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

[2023-09-29 17:12:08.983] [info]: Running hictk v0.0.2-f83f93e
[2023-09-29 17:12:08.983] [info]: Converting data/4DNFIZ1ZVXC8.hic9 to 4DNFIZ1ZVXC8.
 ↪mcool (hic -> mcool)...
[2023-09-29 17:12:09.052] [info]: [1000] begin processing 1000bp matrix...
[2023-09-29 17:12:12.212] [info]: [1000] processing chr2R:11267000-11268000 at␣
 ↪3167564 pixels/s (cache hit rate 0.00%)...
[2023-09-29 17:12:15.346] [info]: [1000] processing chr3R:5672000-5673000 at 3190810␣
 ↪pixels/s (cache hit rate 0.00%)...
[2023-09-29 17:12:18.204] [info]: [1000] processing SCALE normalization vector...
[2023-09-29 17:12:18.241] [info]: [1000] processing VC normalization vector...
[2023-09-29 17:12:18.285] [info]: [1000] processing VC_SQRT normalization vector...
[2023-09-29 17:12:19.123] [info]: [1000] DONE! Processed 26658348 pixels across 8␣
 ↪chromosomes in 10.07s
...
[2023-09-29 17:12:37.412] [info]: DONE! Processed 10 resolution(s) in 28.43s!
[2023-09-29 17:12:37.412] [info]: data/4DNFIZ1ZVXC8.hic9 size: 133.68 MB
[2023-09-29 17:12:37.412] [info]: 4DNFIZ1ZVXC8.mcool size: 100.00 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 1000

[2023-09-29 17:42:47.917] [info]: Running hictk v0.0.2-f83f93e
```

(continues on next page)

(continued from previous page)

```
[2023-09-29 17:42:47.917] [info]: Converting data/4DNFIZ1ZVXC8.hic9 to 4DNFIZ1ZVXC8.
 ↪cool (hic -> cool)...
[2023-09-29 17:42:47.982] [info]: [1000] begin processing 1000bp matrix...
[2023-09-29 17:42:49.982] [info]: [1000] processing chr2R:11267000-11268000 at␣
 ↪5005005 pixels/s (cache hit rate 93.05%)...
[2023-09-29 17:42:52.339] [info]: [1000] processing chr3R:5672000-5673000 at 4242681␣
 ↪pixels/s (cache hit rate 92.66%)...
[2023-09-29 17:42:54.071] [info]: [1000] processing SCALE normalization vector...
[2023-09-29 17:42:54.109] [info]: [1000] processing VC normalization vector...
[2023-09-29 17:42:54.150] [info]: [1000] processing VC_SQRT normalization vector...
[2023-09-29 17:42:54.931] [info]: [1000] DONE! Processed 26658348 pixels across 8␣
 ↪chromosomes in 6.95s
[2023-09-29 17:42:54.931] [info]: DONE! Processed 1 resolution(s) in 7.01s!
[2023-09-29 17:42:54.931] [info]: data/4DNFIZ1ZVXC8.hic9 size: 133.68 MB
[2023-09-29 17:42:54.931] [info]: 4DNFIZ1ZVXC8.cool size: 36.74 MB
```

7.2 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

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

[2024-01-23 17:19:34.045] [info]: Running hictk v0.0.6-570037c-dirty
[2024-01-23 17:19:34.045] [info]: Converting 4DNFIZ1ZVXC8.mcool to 4DNFIZ1ZVXC8.hic␣
 ↪(mcool -> hic)...
[2024-01-23 17:19:37.808] [info]: ingesting pixels at 2700513 pixels/s...
[2024-01-23 17:19:41.916] [info]: ingesting pixels at 2434275 pixels/s...
[2024-01-23 17:19:48.685] [info]: ingesting pixels at 2500000 pixels/s...
[2024-01-23 17:19:52.753] [info]: ingesting pixels at 2458815 pixels/s...
[2024-01-23 17:19:59.034] [info]: ingesting pixels at 2805049 pixels/s...
[2024-01-23 17:20:07.190] [info]: writing header at offset 0
[2024-01-23 17:20:07.190] [info]: begin writing interaction blocks to file
 ↪"4DNFIZ1ZVXC8.hic"...
[2024-01-23 17:20:07.190] [info]: [1000 bp] writing pixels for chr2L:chr2L matrix at␣
 ↪offset 248...
[2024-01-23 17:20:07.595] [info]: [1000 bp] written 2676654 pixels for chr2L:chr2L␣
 ↪matrix
[2024-01-23 17:20:07.651] [info]: [5000 bp] writing pixels for chr2L:chr2L matrix at␣
 ↪offset 4303035...
[2024-01-23 17:20:08.257] [info]: [5000 bp] written 2676654 pixels for chr2L:chr2L␣
 ↪matrix
[2024-01-23 17:20:08.366] [info]: [10000 bp] writing pixels for chr2L:chr2L matrix at␣
 ↪offset 9144982...
[2024-01-23 17:20:08.821] [info]: [10000 bp] written 1433133 pixels for chr2L:chr2L␣
 ↪matrix
...
[2024-01-23 17:21:30.092] [info]: [5000 bp] computing expected vector density
[2024-01-23 17:21:30.240] [info]: [5000 bp] computing expected vector density
[2024-01-23 17:21:30.297] [info]: [1000 bp] computing expected vector density
[2024-01-23 17:21:30.784] [info]: [5000 bp] computing expected vector density
[2024-01-23 17:21:30.784] [info]: writing 50 normalized expected value vectors at␣
 ↪offset 142822186...
[2024-01-23 17:21:30.785] [info]: writing 400 normalization vectors at offset␣
```

(continues on next page)

(continued from previous page)

```
↪143709792...
[2024-01-23 17:21:30.839] [info]: DONE! Processed 10 resolution(s) in 116.79s!
[2024-01-23 17:21:30.839] [info]: 4DNFIZ1ZVXC8.mcool size: 139.38 MB
[2024-01-23 17:21:30.839] [info]: 4DNFIZ1ZVXC8.hic size: 147.52 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, please make sure to increase the number of processing threads with option `--thread`.

CHAPTER

EIGHT

READING INTERACTIONS

`hictk` supports reading interactions from .hic and .cool files through 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 1000

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

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

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.mcool --resolution 1000 --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 1000

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

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

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

Dump balanced or expected interactions:

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1000 --join --balance␣
 ↪SCALE

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

(continues on next page)

(continued from previous page)

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1000 --join --matrix-
↪type expected

chr2L 7000    8000    chr2L   7000    8000    88.33206176757812
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 1000 --join --range␣
↪chr3L:20,000,000-25,000,000

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 1000 --join --range␣
↪chr3L:20,000,000-25,000,000 --range2 chrX

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

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

1000
5000
10000
...
```

Dump cis or trans interactions only:

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1000 --cis-only --join

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

(continues on next page)

(continued from previous page)

```
user@dev:/tmp$ hictk dump data/4DNFIZ1ZVXC8.hic9 --resolution 1000 --trans-only --join

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

CHAPTER

NINE

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
- COO

File requirements:

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

```
# Create a 10kbp .cool file using dm6 as reference
user@dev:/tmp$ zcat 4DNFIKNWM36K.pairs.gz | hictk load --format 4dn --assembly dm6 --
 ↪bin-size 10000 dm6.chrom.sizes 4DNFIKNWM36K.10000.cool

[2024-01-23 15:15:00.520] [info]: Running hictk v0.0.6-45c36af-dirty
[2024-01-23 15:15:00.531] [info]: writing chunk #1 to intermediate file "/tmp/
 ↪4DNFIKNWM36K.10000.cool.tmp/4DNFIKNWM36K.10000.cool.tmp"...
[2024-01-23 15:15:23.762] [info]: done writing chunk #1 to tmp file "/tmp/
 ↪4DNFIKNWM36K.10000.cool.tmp/4DNFIKNWM36K.10000.cool.tmp".
[2024-01-23 15:15:23.762] [info]: writing chunk #2 to intermediate file "/tmp/
 ↪4DNFIKNWM36K.10000.cool.tmp/4DNFIKNWM36K.10000.cool.tmp"...
[2024-01-23 15:15:49.042] [info]: done writing chunk #2 to tmp file "/tmp/
 ↪4DNFIKNWM36K.10000.cool.tmp/4DNFIKNWM36K.10000.cool.tmp".
[2024-01-23 15:15:49.042] [info]: writing chunk #3 to intermediate file "/tmp/
 ↪4DNFIKNWM36K.10000.cool.tmp/4DNFIKNWM36K.10000.cool.tmp"...
[2024-01-23 15:15:49.834] [info]: done writing chunk #3 to tmp file "/tmp/
 ↪4DNFIKNWM36K.10000.cool.tmp/4DNFIKNWM36K.10000.cool.tmp".
[2024-01-23 15:15:49.836] [info]: merging 3 chunks into "4DNFIKNWM36K.10000.cool"...
[2024-01-23 15:15:55.118] [info]: processing chr3L:15100000-15110000 chr3L:16230000-
 ↪16240000 at 4789272 pixels/s...
[2024-01-23 15:15:59.718] [info]: ingested 119208613 interactions (18122865 nnz) in␣
 ↪59.197723453s!

# Create a 10kbp .hic file using dm6 as reference
user@dev:/tmp$ zcat 4DNFIKNWM36K.pairs.gz | hictk load --format 4dn --assembly dm6 --
 ↪bin-size 10000 dm6.chrom.sizes 4DNFIKNWM36K.10000.hic

[2024-01-23 15:45:19.969] [info]: Running hictk v0.0.6-570037c-dirty
[2024-01-23 15:45:42.439] [info]: preprocessing chunk #1 at 452919 pixels/s...
[2024-01-23 15:46:09.182] [info]: preprocessing chunk #2 at 303750 pixels/s...
[2024-01-23 15:46:11.184] [info]: writing header at offset 0
[2024-01-23 15:46:11.184] [info]: begin writing interaction blocks to file
```

(continues on next page)

(continued from previous page)

```
↪"4DNFIKNWM36K.10000.hic"...
[2024-01-23 15:46:11.184] [info]: [10000 bp] writing pixels for chr3R:chr3R matrix at␣
↪offset 50632...
[2024-01-23 15:46:13.295] [info]: [10000 bp] written 2264963 pixels for chr3R:chr3R␣
↪matrix
[2024-01-23 15:46:13.295] [info]: [10000 bp] writing pixels for chr3R:chr3L matrix at␣
↪offset 4235718...
[2024-01-23 15:46:14.611] [info]: [10000 bp] written 1610264 pixels for chr3R:chr3L␣
↪matrix
...
[2024-01-23 15:46:44.065] [info]: [10000 bp] initializing expected value vector
[2024-01-23 15:46:50.531] [info]: [10000 bp] computing expected vector density
[2024-01-23 15:46:51.157] [info]: writing 1 expected value vectors at offset 32065110.
↪..
[2024-01-23 15:46:51.158] [info]: writing 0 normalized expected value vectors at␣
↪offset 32078017...
[2024-01-23 15:46:51.194] [info]: ingested 119208613 interactions (18122865 nnz) in␣
↪91.225341628s!
```

Tips:

- When creating large .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.

9.1 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/1000 data/
↪4DNFIZ1ZVXC8.mcool::/resolutions/1000 -o 4DNFIZ1ZVXC8.merged.cool

[2023-09-29 19:24:49.479] [info]: Running hictk v0.0.2
[2023-09-29 19:24:49.479] [info]: begin merging 2 coolers...
[2023-09-29 19:24:52.032] [info]: processing chr2R:11267000-11268000 chr4:1052000-
↪1053000 at 3976143 pixels/s...
[2023-09-29 19:24:55.157] [info]: processing chr3R:5812000-5813000 chr3R:23422000-
↪23423000 at 3201024 pixels/s...
[2023-09-29 19:24:57.992] [info]: DONE! Merging 2 coolers took 8.51s!
[2023-09-29 19:24:57.992] [info]: 4DNFIZ1ZVXC8.merged.cool size: 36.23 MB

# Merge multiple .hic files

user@dev:/tmp$ hictk merge data/4DNFIZ1ZVXC8.hic9 data/4DNFIZ1ZVXC8.hic9 -o␣
↪4DNFIZ1ZVXC8.10000.merged.hic --resolution 10000

[2024-01-23 15:49:23.248] [info]: Running hictk v0.0.6-570037c-dirty
[2024-01-23 15:49:23.248] [info]: begin merging 2 .hic files...
[2024-01-23 15:49:31.101] [info]: ingesting pixels at 1352814 pixels/s...
[2024-01-23 15:49:37.777] [info]: writing header at offset 0
[2024-01-23 15:49:37.777] [info]: begin writing interaction blocks to file
↪"4DNFIZ1ZVXC8.10000.merged.hic"...
[2024-01-23 15:49:37.777] [info]: [10000 bp] writing pixels for chr2L:chr2L matrix at␣
↪offset 212...
```

(continues on next page)

(continued from previous page)

```
[2024-01-23 15:49:39.060] [info]: [10000 bp] written 1433133 pixels for chr2L:chr2L␣
 ↪matrix
[2024-01-23 15:49:39.060] [info]: [10000 bp] writing pixels for chr2L:chr2R matrix at␣
 ↪offset 2619165...
...
[2024-01-23 15:49:58.624] [info]: [10000 bp] initializing expected value vector
[2024-01-23 15:50:05.276] [info]: [10000 bp] computing expected vector density
[2024-01-23 15:50:05.276] [info]: writing 1 expected value vectors at offset 31936601.
 ↪..
[2024-01-23 15:50:05.276] [info]: writing 0 normalized expected value vectors at␣
 ↪offset 31949508...
[2024-01-23 15:50:05.299] [info]: DONE! Merging 2 files took 42.05s!
[2024-01-23 15:50:05.299] [info]: 4DNFIZ1ZVXC8.10000.merged.hic size: 31.95 MB
```

Tips:

- When merging many, large .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.

CHAPTER

TEN

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

10.1 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

[2023-09-29 19:28:39.926] [info]: Running hictk v0.0.2
[2023-09-29 19:28:39.929] [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)
[2023-09-29 19:28:39.929] [info]: copying 1000 resolution from data/4DNFIZ1ZVXC8.
↪mcool::/resolutions/1000
[2023-09-29 19:28:40.119] [info]: generating 2000 resolution from 1000 (2x)
[2023-09-29 19:28:40.343] [info]: [1000 -> 2000] processing chr2L:1996000-1998000 at␣
↪4484305 pixels/s...
[2023-09-29 19:28:40.663] [info]: [1000 -> 2000] processing chr2L:4932000-4934000 at␣
↪3125000 pixels/s...
[2023-09-29 19:28:40.973] [info]: [1000 -> 2000] processing chr2L:7986000-7988000 at␣
↪3236246 pixels/s...
...
[2023-09-29 19:29:12.513] [info]: generating 10000000 resolution from 5000000 (2x)
[2023-09-29 19:29:12.519] [info]: DONE! Processed 13 resolution(s) in 32.59s!

# Coarsen a single resolution
user@dev:/tmp$ hictk zoomify data/4DNFIZ1ZVXC8.mcool::/resolutions/1000 out.cool --
↪resolutions 50000

[2023-09-29 19:30:52.476] [info]: Running hictk v0.0.2
[2023-09-29 19:30:52.482] [info]: coarsening cooler at data/4DNFIZ1ZVXC8.mcool::/
↪resolutions/1000 2 times (1000 -> 1000 -> 50000)
[2023-09-29 19:30:52.482] [info]: copying 1000 resolution from data/4DNFIZ1ZVXC8.
↪mcool::/resolutions/1000
[2023-09-29 19:30:52.668] [info]: generating 50000 resolution from 1000 (50x)
[2023-09-29 19:30:53.789] [info]: [1000 -> 50000] processing chr2L:23000000-23050000␣
↪at 896057 pixels/s...
[2023-09-29 19:30:55.005] [info]: [1000 -> 50000] processing chr3L:4600000-4650000 at␣
↪822368 pixels/s...
[2023-09-29 19:30:56.440] [info]: [1000 -> 50000] processing chr3R:32050000-32079331␣
↪at 696864 pixels/s...
[2023-09-29 19:30:56.863] [info]: DONE! Processed 2 resolution(s) in 4.39s!
```

10.2 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.

```
user@dev:/tmp$ hictk zoomify 4DNFIZ1ZVXC8.hic9 4DNFIZ1ZVXC8.zoomified.hic --threads 8

[2024-01-23 16:59:57.369] [info]: Running hictk v0.0.6-570037c-dirty
[2024-01-23 16:59:57.369] [info]: copying resolution 1000 from "4DNFIZ1ZVXC8.hic9"
[2024-01-23 16:59:57.369] [info]: generating 2000 resolution from 1000 (2x)
[2024-01-23 16:59:57.369] [info]: copying resolution 5000 from "4DNFIZ1ZVXC8.hic9"
[2024-01-23 16:59:57.369] [info]: copying resolution 10000 from "4DNFIZ1ZVXC8.hic9"
[2024-01-23 16:59:57.369] [info]: generating 20000 resolution from 10000 (2x)
[2024-01-23 16:59:57.369] [info]: copying resolution 50000 from "4DNFIZ1ZVXC8.hic9"
[2024-01-23 16:59:57.369] [info]: copying resolution 100000 from "4DNFIZ1ZVXC8.hic9"
[2024-01-23 16:59:57.369] [info]: generating 200000 resolution from 100000 (2x)
[2024-01-23 16:59:57.369] [info]: copying resolution 500000 from "4DNFIZ1ZVXC8.hic9"
[2024-01-23 16:59:57.369] [info]: copying resolution 1000000 from "4DNFIZ1ZVXC8.hic9"
[2024-01-23 16:59:57.369] [info]: generating 2000000 resolution from 1000000 (2x)
[2024-01-23 16:59:57.369] [info]: generating 5000000 resolution from 1000000 (5x)
[2024-01-23 16:59:57.369] [info]: generating 10000000 resolution from 5000000 (2x)
[2024-01-23 16:59:57.379] [info]: [1000 bp] ingesting interactions...
[2024-01-23 17:00:02.183] [info]: ingesting pixels at 2157032 pixels/s...
[2024-01-23 17:00:07.271] [info]: ingesting pixels at 1965795 pixels/s...
...
[2024-01-23 17:02:04.842] [info]: [1000 bp] computing expected vector density
[2024-01-23 17:02:05.325] [info]: [2000 bp] computing expected vector density
[2024-01-23 17:02:06.291] [info]: [5000 bp] computing expected vector density
[2024-01-23 17:02:06.292] [info]: writing 13 expected value vectors at offset␣
↪193918320...
[2024-01-23 17:02:06.293] [info]: writing 0 normalized expected value vectors at␣
↪offset 194161639...
[2024-01-23 17:02:06.318] [info]: DONE! Processed 13 resolution(s) in 128.95s!
```

Tips:

- When zoomifying large .hic files, `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.

CHAPTER

ELEVEN

BALANCING HI-C MATRICES

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

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

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

[2023-10-01 13:18:02.119] [info]: Running hictk v0.0.2-f83f93e
[2023-10-01 13:18:02.130] [info]: Writing interactions to temporary file /tmp/
↪4DNFIZ1ZVXC8.tmp0...
[2023-10-01 13:18:05.098] [info]: Initializing bias vector...
[2023-10-01 13:18:05.099] [info]: Masking rows with fewer than 10 nnz entries...
[2023-10-01 13:18:06.298] [info]: Masking rows using mad_max=5...
[2023-10-01 13:18:06.971] [info]: Iteration 1: 36874560.192587376
[2023-10-01 13:18:07.634] [info]: Iteration 2: 21347543.04950776
[2023-10-01 13:18:08.307] [info]: Iteration 3: 7819314.542541969
...
[2023-10-01 13:19:20.365] [info]: Iteration 105: 2.1397932757529552e-05
[2023-10-01 13:19:21.146] [info]: Iteration 106: 1.6604770462001875e-05
[2023-10-01 13:19:21.870] [info]: Iteration 107: 1.2885285040054778e-05
[2023-10-01 13:19:22.608] [info]: Iteration 108: 9.99900768769869e-06
[2023-10-01 13:19:22.619] [info]: Writing weights to 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 of "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

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 `--thread` 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

TWELVE

REORDERING CHROMOSOMES

12.1 TLDR

```
# Important! --bin-size should be the same resolution as matrix.cool
user@dev:/tmp hictk load --format=bg2 \
                         --bin-size=1000 \
                         <(hictk dump --table=chroms matrix.cool |
                           sort -k2,2nr) \
                         output.cool \
                         < <(hictk dump --join matrix.cool)
```

12.2 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.

12.3 Walkthrough

For this tutorial, we will use file `4DNFIZ1ZVXC8.mcool` 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 4DNFIZ1ZVXC8.mcool --table=chroms | tee chrom.sizes

chr2L       23513712
chr2R       25286936
chr3L       28110227
chr3R       32079331
chr4        1348131
chrX        23542271
chrY        3667352
```

Second, we re-order chromosomes:

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

chr3R       32079331
chr3L       28110227
```

(continues on next page)

(continued from previous page)

```
chr2R       25286936
chrX        23542271
chr2L       23513712
chrY        3667352
chr4        1348131
```

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

```
user@dev:/tmp hictk dump 4DNFIZ1ZVXC8.mcool --join --resolution=1000 > pixels.bg2

user@dev:/tmp head pixels.bg2

chr2L       5000    6000    chr2L   5000    6000    127
chr2L       5000    6000    chr2L   6000    7000    129
chr2L       5000    6000    chr2L   7000    8000    60
chr2L       5000    6000    chr2L   8000    9000    77
chr2L       5000    6000    chr2L   9000    10000   97
chr2L       5000    6000    chr2L   10000   11000   3
chr2L       5000    6000    chr2L   11000   12000   1
chr2L       5000    6000    chr2L   12000   13000   66
chr2L       5000    6000    chr2L   13000   14000   116
chr2L       5000    6000    chr2L   14000   15000   64
```

Finally, we load pixels into a new .cool file

```
user@dev:/tmp hictk load --format=bg2 \
                         --bin-size=1000 \
                         chrom.sizes.sorted \
                         output.cool < pixels.bg2

[2024-03-21 12:27:16.998] [info]: Running hictk v0.0.10-1c2bafd
[2024-03-21 12:27:16.998] [info]: begin loading unsorted pixels into a .cool file...
[2024-03-21 12:27:17.077] [info]: writing chunk #1 to intermediate file "/tmp/output.
 ↪cool.tmp/output.cool.tmp"...
[2024-03-21 12:27:20.945] [info]: done writing chunk #1 to tmp file "/tmp/output.cool.
 ↪tmp/output.cool.tmp".
[2024-03-21 12:27:20.945] [info]: writing chunk #2 to intermediate file "/tmp/output.
 ↪cool.tmp/output.cool.tmp"...
[2024-03-21 12:27:24.890] [info]: done writing chunk #2 to tmp file "/tmp/output.cool.
 ↪tmp/output.cool.tmp".
[2024-03-21 12:27:24.890] [info]: writing chunk #3 to intermediate file "/tmp/output.
 ↪cool.tmp/output.cool.tmp"...
[2024-03-21 12:27:28.823] [info]: done writing chunk #3 to tmp file "/tmp/output.cool.
 ↪tmp/output.cool.tmp".
[2024-03-21 12:27:28.823] [info]: writing chunk #4 to intermediate file "/tmp/output.
 ↪cool.tmp/output.cool.tmp"...
[2024-03-21 12:27:32.668] [info]: done writing chunk #4 to tmp file "/tmp/output.cool.
 ↪tmp/output.cool.tmp".
[2024-03-21 12:27:32.668] [info]: writing chunk #5 to intermediate file "/tmp/output.
 ↪cool.tmp/output.cool.tmp"...
[2024-03-21 12:27:36.070] [info]: done writing chunk #5 to tmp file "/tmp/output.cool.
 ↪tmp/output.cool.tmp".
[2024-03-21 12:27:36.070] [info]: writing chunk #6 to intermediate file "/tmp/output.
 ↪cool.tmp/output.cool.tmp"...
[2024-03-21 12:27:36.079] [info]: done writing chunk #6 to tmp file "/tmp/output.cool.
 ↪tmp/output.cool.tmp".
[2024-03-21 12:27:36.080] [info]: merging 6 chunks into "output.cool"...
```

(continues on next page)

(continued from previous page)

```
[2024-03-21 12:27:38.572] [info]: processing chr3R:20786000-20787000 chr3R:20808000-
 ↪20809000 at 4091653 pixels/s...
[2024-03-21 12:27:41.443] [info]: processing chr3L:7391000-7392000 chr3L:7417000-
 ↪7418000 at 3484321 pixels/s...
[2024-03-21 12:27:44.292] [info]: processing chr2R:9278000-9279000 chrX:5993000-
 ↪5994000 at 3510004 pixels/s...
[2024-03-21 12:27:47.062] [info]: processing chrX:14217000-14218000 chrX:17476000-
 ↪17477000 at 3611412 pixels/s...
[2024-03-21 12:27:49.901] [info]: ingested 119208613 interactions (48469783 nnz) in␣
 ↪32.902465965s!
```

Lastly, we check that chromosomes are properly sorted:

```
user@dev:/tmp hictk dump 4DNFIZ1ZVXC8.mcool --table=chroms

chr3R       32079331
chr3L       28110227
chr2R       25286936
chrX        23542271
chr2L       23513712
chrY        3667352
chr4        1348131
```

12.4 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 --format=bg2 \
                         --bin-size=1000 \
                         chrom.sizes.sorted \
                         output.cool \
                         < <(hictk dump 4DNFIZ1ZVXC8.mcool --join --resolution=1000)
```

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 --format=bg2 \
                         --bin-size=1000 \
                         --tmpdir=/var/tmp/ \
                         chrom.sizes.sorted \
                         output.cool \
                         < <(hictk dump 4DNFIZ1ZVXC8.mcool --join --resolution=1000)
```

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

THIRTEEN

DUMP INTERACTIONS TO .COOL OR .HIC FILE

13.1 TLDR

```
# Important! --bin-size should be the same resolution as matrix.cool
user@dev:/tmp hictk load --format=bg2 \
                         --bin-size=1000 \
                         <(hictk dump --table=chroms matrix.cool)
                         output.cool \
                         < <(hictk dump --join
                                        --range=chr2L:0-10,000,000
                                        --range2=chr3R:0-10,000,000
                                        matrix.cool)
```

13.2 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`.

13.3 Walkthrough

For this tutorial, we will use file `4DNFIZ1ZVXC8.mcool` 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 4DNFIZ1ZVXC8.mcool --table=chroms | tee chrom.sizes

chr2L       23513712
chr2R       25286936
chr3L       28110227
chr3R       32079331
chr4        1348131
chrX        23542271
chrY        3667352
```

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

```
user@dev:/tmp hictk dump 4DNFIZ1ZVXC8.mcool \
                         --join \
                         --resolution=1000 \
```

(continues on next page)

(continued from previous page)

```
                        --range=chr2L:5,000,000-10,000,000 \
                        --range2=chr3R:7,500,000-10,000,000 > pixels.bg2

user@dev:/tmp head pixels.bg2

chr2L       5000000 5001000 chr3R   7506000 7507000 1
chr2L       5000000 5001000 chr3R   7624000 7625000 1
chr2L       5000000 5001000 chr3R   7943000 7944000 1
chr2L       5000000 5001000 chr3R   8014000 8015000 1
chr2L       5000000 5001000 chr3R   8130000 8131000 1
chr2L       5000000 5001000 chr3R   8245000 8246000 1
chr2L       5000000 5001000 chr3R   8855000 8856000 1
chr2L       5000000 5001000 chr3R   9032000 9033000 1
chr2L       5000000 5001000 chr3R   9171000 9172000 1
chr2L       5000000 5001000 chr3R   9380000 9381000 1
```

Finally, we load pixels into a new .cool file

```
user@dev:/tmp hictk load --format=bg2 \
                        --bin-size=1000 \
                        chrom.sizes \
                        output.cool < pixels.bg2

[2024-03-21 13:22:57.542] [info]: Running hictk v0.0.10-1c2bafd
[2024-03-21 13:22:57.542] [info]: begin loading unsorted pixels into a .cool file...
[2024-03-21 13:22:57.613] [info]: writing chunk #1 to intermediate file "/tmp/output.
↪cool.tmp/output.cool.tmp"...
[2024-03-21 13:22:57.630] [info]: done writing chunk #1 to tmp file "/tmp/output.cool.
↪tmp/output.cool.tmp".
[2024-03-21 13:22:57.630] [info]: writing chunk #2 to intermediate file "/tmp/output.
↪cool.tmp/output.cool.tmp"...
[2024-03-21 13:22:57.634] [info]: done writing chunk #2 to tmp file "/tmp/output.cool.
↪tmp/output.cool.tmp".
[2024-03-21 13:22:57.634] [info]: merging 2 chunks into "output.cool"...
[2024-03-21 13:22:57.676] [info]: ingested 26214 interactions (25085 nnz) in 0.
↪133955616s!
```

13.3.1 Removing empty chromosomes from the reference genome

This can be easily achieved by grepping `chr2L` and `chr3R` when generating the `chrom.sizes` file.

```
user@dev:/tmp hictk dump 4DNFIZ1ZVXC8.mcool --table=chroms |
              grep -e 'chr2L' -e 'chr3R' |
              tee chrom.sizes

chr2L       23513712
chr3R       32079331
```

13.4 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 --format=bg2 \
                         --bin-size=1000 \
                         chrom.sizes \
                         output.cool \
                         < <(hictk dump 4DNFIZ1ZVXC8.mcool \
                                        --join \
                                        --resolution=1000 \
                                        --range=chr2L:0-10,000,000 \
                                        --range2=chr3R: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 --format=bg2 \
                         --bin-size=1000 \
                         --tmpdir=/var/tmp/ \
                         chrom.sizes.sorted \
                         output.cool \
                         < <(hictk dump 4DNFIZ1ZVXC8.mcool \
                                        --join \
                                        --resolution=1000 \
                                        --range=chr2L:0-10,000,000 \
                                        --range2=chr3R: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

FOURTEEN

CLI REFERENCE

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

14.1 Subcommands

```
Blazing fast tools to work with .hic and .cool files.
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 matrices using ICE, SCALE, or VC.
  convert                     Convert HiC matrices to a different format.
  dump                        Dump data from .hic and Cooler files 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.
  rename-chromosomes, rename-chroms
                              Rename chromosomes found in a Cooler file.
  validate                    Validate .hic and Cooler files.
  zoomify                     Convert single-resolution Cooler and .hic files to␣
↪multi-resolution by coarsening.
```

14.2 hictk balance

```
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.
```

14.3 hictk balance ice

```
Balance Hi-C matrices using ICE.
Usage: hictk balance ice [OPTIONS] input
Positionals:
  input TEXT:((HiC) OR (Cooler)) OR (Multires-cooler) 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 [/tmp]        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        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␣
↪them to the input file.
  --chunk-size UINT:POSITIVE [10000000]
                              Number of interactions to process at once. Ignored when␣
↪using --in-memory.
  -v,--verbosity UINT:INT in [1 - 4] []
                              Set verbosity of output to the console.
  -t,--threads UINT:UINT in [1 - 16] [1]
                              Maximum number of parallel threads to spawn.
  -l,--compression-lvl UINT:INT in [0 - 19] []
                              Compression level used to compress temporary files␣
↪using ZSTD.
  -f,--force                  Overwrite existing files and datasets (if any).
```

14.4 hictk balance scale

```
Balance Hi-C matrices using SCALE.
Usage: hictk balance scale [OPTIONS] input
Positionals:
  input TEXT:((HiC) OR (Cooler)) OR (Multires-cooler) 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 [/tmp]        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 [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 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        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␣
↪them to the input file.
  --chunk-size UINT:POSITIVE [10000000]
                              Number of interactions to process at once. Ignored when␣
↪using --in-memory.
  -v,--verbosity UINT:INT in [1 - 4] []
                              Set verbosity of output to the console.
  -t,--threads UINT:UINT in [1 - 16] [1]
                              Maximum number of parallel threads to spawn.
  -l,--compression-lvl UINT:INT in [0 - 19] []
                              Compression level used to compress temporary files␣
↪using ZSTD.
  -f,--force                  Overwrite existing files and datasets (if any).
```

14.5 hictk balance vc

```
Balance Hi-C matrices using VC.
Usage: hictk balance vc [OPTIONS] input
Positionals:
  input TEXT:((HiC) OR (Cooler)) OR (Multires-cooler) 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        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 UINT:INT in [1 - 4] []
                              Set verbosity of output to the console.
  -f,--force                  Overwrite existing files and datasets (if any).
```

14.6 hictk convert

```
Convert HiC matrices to a different format.
Usage: hictk convert [OPTIONS] input output
Positionals:
  input TEXT:((HiC) OR (Cooler)) OR (Multires-cooler) 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␣
```

(continues on next page)

(continued from previous page)

```
↪are copied.
                              Pass NONE to avoid copying 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                Path where to store temporary files.
 --chunk-size UINT:POSITIVE [10000000]
                              Batch size to use when converting .[m]cool to .hic.
 -v,--verbosity UINT:INT in [1 - 4] []
                              Set verbosity of output to the console.
 -t,--threads UINT:UINT in [2 - 16] [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.
 -f,--force                   Overwrite existing files (if any).
```

14.7 hictk dump

```
Dump data from .hic and Cooler files to stdout.
Usage: hictk dump [OPTIONS] uri
Positionals:
  uri TEXT:(((HiC) OR (Cooler)) OR (Multires-cooler)) OR (Single-cell-cooler) 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:value in {expected->2,observed->0,oe->1} OR {2,0,1} [observed]
                              Matrix type (ignored when file is not in .hic format).
  --matrix-unit ENUM:value in {BP->0,FRAG->1} OR {0,1} [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
```

(continues on next page)

(continued from previous page)

```
                              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.
```

14.8 hictk fix-mcool

```
Fix corrupted .mcool files.
Usage: hictk fix-mcool [OPTIONS] input output
Positionals:
  input TEXT:Multires-cooler 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 [/tmp]        Path to a folder where to store temporary data.
  --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 UINT:INT in [1 - 4] []
                              Set verbosity of output to the console.
  -t,--threads UINT:UINT in [1 - 16] [1]
                              Maximum number of parallel threads to spawn (only␣
↪applies to the balancing stage).
  -l,--compression-lvl UINT:INT in [0 - 19] []
                              Compression level used to compress temporary files␣
↪using ZSTD (only applies to the balancing stage).
  -f,--force                  Overwrite existing files (if any).
```

14.9 hictk load

```
Build .cool and .hic files from interactions in various text formats.
Usage: hictk load [OPTIONS] chrom-sizes output-path
Positionals:
  chrom-sizes TEXT:FILE REQUIRED
                              Path to .chrom.sizes file.
  output-path TEXT REQUIRED   Path to output file.
Options:
  -h,--help                   Print this help message and exit
  -b,--bin-size UINT:POSITIVE Excludes: --bin-table
                              Bin size (bp).
                              Required when --bin-table is not used.
  --bin-table TEXT:FILE Excludes: --bin-size
                              Path to a BED3+ file with the bin table.
```

(continues on next page)

(continued from previous page)

```
  -f,--format TEXT:{4dn,validpairs,bg2,coo} REQUIRED
                              Input format.
  --force                     Force overwrite existing output file(s).
  --assembly TEXT [unknown]   Assembly name.
  --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 validapairs 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.
  --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 [1 - 16] [1]
                              Maximum number of parallel threads to spawn.
                              When loading interactions in a .cool file, only a␣
 ↪single thread will be used.
  --tmpdir TEXT [/tmp]        Path to a folder where to store temporary data.
  -v,--verbosity UINT:INT in [1 - 4] []
                              Set verbosity of output to the console.
```

14.10 hictk merge

```
Merge multiple Cooler or .hic files into a single file.
Usage: hictk merge [OPTIONS] input-files...
Positionals:
  input-files TEXT:(Cooler) OR (HiC) 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).
  --resolution UINT:NONNEGATIVE
                              HiC 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 - 16] [1]
```

(continues on next page)

(continued from previous page)

```
                              Maximum number of parallel threads to spawn.
                              When merging interactions in Cooler format, only a␣
↪single thread will be used.
  --tmpdir TEXT [/tmp]        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.
  -v,--verbosity UINT:INT in [1 - 4] []
                              Set verbosity of output to the console.
```

14.11 hictk rename-chromosomes

```
Rename chromosomes found in a Cooler file.
Usage: hictk rename-chromosomes [OPTIONS] uri
Positionals:
  uri TEXT REQUIRED           Path to a or .[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 UINT:INT in [1 - 4] []
                              Set verbosity of output to the console.
```

14.12 hictk validate

```
Validate .hic and Cooler files.
Usage: 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).
  --quiet                     Don't print anything to stdout. Success/failure is␣
↪reported through exit codes
```

14.13 hictk zoomify

```
Convert single-resolution Cooler and .hic files to multi-resolution by coarsening.
Usage: hictk zoomify [OPTIONS] cooler/hic mcool/hic
Positionals:
  cooler/hic TEXT:(Cooler) OR (HiC) REQUIRED
                              Path to a .cool or .hic file (Cooler URI syntax␣
 ↪supported).
  mcool/hic TEXT REQUIRED     Output path.
Options:
  -h,--help                   Print this help message and exit
  --force                     Force overwrite existing output file(s).
  --resolutions UINT ...      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 - 16] [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 [/tmp]        Path to a folder where to store temporary data.
  -v,--verbosity UINT:INT in [1 - 4] []
                              Set verbosity of output to the console.
```

CHAPTER

FIFTEEN

C++ API REFERENCE

hictk C++ API is structured as follows:

15.1 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.

15.1.1 Common

enum class **QUERY_TYPE**

enumerator **BED**

enumerator **UCSC**

15.1.2 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 uri, std::uint32_t resolution = 0, 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**() 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]] std::vector<balancing::Method> **avail_normalizations**() 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
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 noexcept;

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

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

15.1.3 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;

template<typename **N**>
[[nodiscard]] Eigen::SparseMatrix<*N*> **read_sparse**() const;

template<typename **N**>
[[nodiscard]] Eigen::Matrix<*N*, Eigen::Dynamic, Eigen::Dynamic> **read_dense**() const;

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

Accessors

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

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

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

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

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

Advanced

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

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

[[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);
```

15.2 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)

15.2.1 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 **operator bool**() 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()) const;

[[nodiscard]] *PixelSelector* **fetch**(std::shared_ptr<const balancing::Weights> weights) 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;

std::shared_ptr<const balancing::Weights> **read_normalization**(std::string_view normalization, bool rescale = false) const;

std::shared_ptr<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;

std::shared_ptr<const balancing::Weights> **normalization**(const balancing::Method &normalization, bool rescale = false) const;

std::shared_ptr<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;

15.2.2 Multi-resolution Cooler (.mcool)

class **MultiResFile**

Constructors

explicit **MultiResFile**(const std::filesystem::path &path, unsigned int 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);

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

15.2.3 Single-cell Cooler (.scool)

class **SingleCellFile**

Constructors

explicit **SingleCellFile**(const std::filesystem::path &path, unsigned int 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]] 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;

15.2.4 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;

template<typename **N**>
[[nodiscard]] Eigen::SparseMatrix<*N*> **read_sparse**() const;

template<typename **N**>
[[nodiscard]] Eigen::Matrix<*N*, Eigen::Dynamic, Eigen::Dynamic> **read_dense**() const;

Accessors

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

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

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

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

15.3 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

15.3.1 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**

15.3.2 File handle

class **File**

Constructors

explicit **File**(std::string url_, 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::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**() 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;

Fetch methods (1D queries)

[[nodiscard]] PixelSelectorAll **fetch**(balancing::Method norm = balancing::Method::NONE()) const;

[[nodiscard]] *PixelSelector* **fetch**(std::string_view range, balancing::Method norm = 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, balancing::Method norm = balancing::Method::NONE()) const;

[[nodiscard]] *PixelSelector* **fetch**(std::uint64_t first_bin, std::uint64_t last_bin, balancing::Method norm = balancing::Method::NONE()) const;

Fetch methods (2D queries)

[[nodiscard]] *PixelSelector* **fetch**(std::string_view range1, std::string_view range2, balancing::Method norm = 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, balancing::Method norm = 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, balancing::Method norm = balancing::Method::NONE()) 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;

15.3.3 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;

template<typename **N**>
[[nodiscard]] Eigen::SparseMatrix<*N*> **read_sparse**() const;

template<typename **N**>

[[nodiscard]] Eigen::Matrix<*N*, Eigen::Dynamic, Eigen::Dynamic> **read_dense**() const;

Accessors

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

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

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

[[nodiscard]] 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]] 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;

15.4 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.

15.4.1 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;

15.4.2 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, 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');

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;

15.4.3 Genomic bins

class **Bin**

Class modeling genomic bins.

The class is implemented as a thin wrapper around *GenomicIntervals*. 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;

15.4.4 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;

15.4.5 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.

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;

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

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

[[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;

15.4.6 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 **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 **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;

INDEX

H