pairtools Documentation

Release 0.3.1-dev.1

Open2C

Feb 23, 2021

Contents

pairtools is a simple and fast command-line framework to process sequencing data from a Hi-C experiment. *pairtools* perform various operations on Hi-C pairs and occupy the middle position in a typical Hi-C data processing pipeline:

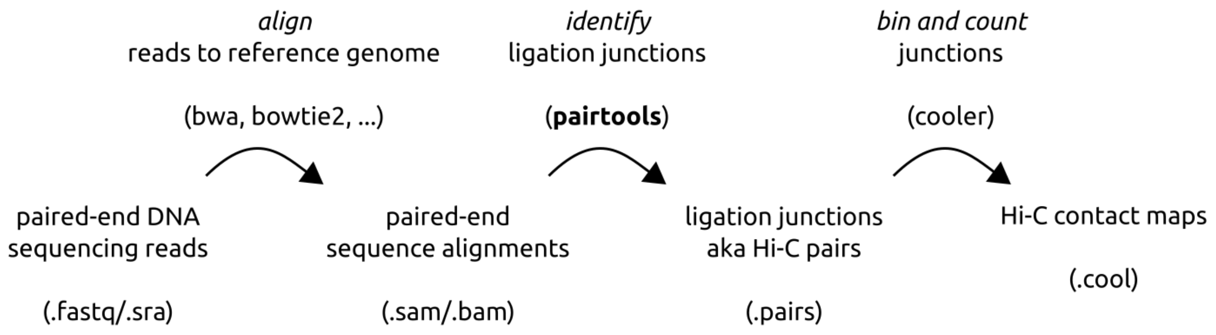

Fig. 1: In a typical Hi-C pipeline, DNA sequences (reads) are aligned to the reference genome, converted into ligation junctions and binned, thus producing a Hi-C contact map.

pairtools aim to be an all-in-one tool for processing Hi-C pairs, and can perform following operations:

- detect ligation junctions (a.k.a. Hi-C pairs) in aligned paired-end sequences of Hi-C DNA molecules
- sort .pairs files for downstream analyses
- detect, tag and remove PCR/optical duplicates
- generate extensive statistics of Hi-C datasets
- select Hi-C pairs given flexibly defined criteria
- restore .sam alignments from Hi-C pairs

pairtools produce .pairs files compliant with the 4DN standard.

pairtools uses a two-character notation to define pair types (see table _section-pair-types)

The full list of available pairtools:

Pairtool	Description
dedup	Find and remove PCR/optical duplicates.
filterbycov	Remove pairs from regions of high coverage.
flip	Flip pairs to get an upper-triangular matrix.
markasdup	Tag pairs as duplicates.
merge	Merge sorted .pairs/.pairsam files.
parse	Find ligation junctions in .sam, make .pairs.
phase	Phase pairs mapped to a diploid genome.
restrict	Assign restriction fragments to pairs.
select	Select pairs according to some condition.
sort	Sort a .pairs/.pairsam file.
split	Split a .pairsam file into .pairs and .sam.
stats	Calculate pairs statistics.

Contents:

CHAPTER 1

Quickstart

Install *pairtools* and all of its dependencies using the conda package manager and the bioconda channel for bioinformatics software.

```
$ conda install -c conda-forge -c bioconda pairtools
```

Setup a new test folder and download a small Hi-C dataset mapped to sacCer3 genome:

```
$ mkdir /tmp/test-pairtools
$ cd /tmp/test-pairtools
$ wget https://github.com/open2c/distiller-test-data/raw/master/bam/MATalpha_R1.bam
```

Additionally, we will need a .chromsizes file, a TAB-separated plain text table describing the names, sizes and the order of chromosomes in the genome assembly used during mapping:

```
$ wget https://raw.githubusercontent.com/open2c/distiller-test-data/master/genome/
↪sacCer3.reduced.chrom.sizes
```

With *pairtools parse*, we can convert paired-end sequence alignments stored in .sam/.bam format into .pairs, a TAB-separated table of Hi-C ligation junctions:

```
$ pairtools parse -c sacCer3.reduced.chrom.sizes -o MATalpha_R1.pairs.gz --drop-sam␣
↪MATalpha_R1.bam
```

Inspect the resulting table:

```
$ less MATalpha_R1.pairs.gz
```

CHAPTER 2

Installation

2.1 Requirements

- Python 3.x
- Python packages *numpy* and *click*
- Command-line utilities *sort* (the Unix version), *bgzip* (shipped with *samtools*) and *samtools*. If available, *pairtools* can compress outputs with *bgzip*, *pbgzip* and *lz4*.

2.2 Install using conda

We highly recommend using the *conda* package manager to install pre-compiled *pairtools* together with all its dependencies. To get it, you can either install the full Anaconda Python distribution or just the standalone conda package manager.

With *conda*, you can install pre-compiled *pairtools* and all of its dependencies from the bioconda channel:

```
$ conda install -c conda-forge -c bioconda pairtools
```

2.3 Install using pip

Alternatively, compile and install *pairtools* and its Python dependencies from PyPI using pip:

```
$ pip install pairtools
```

2.4 Install the development version

Finally, you can install the latest development version of *pairtools* from github. First, make a local clone of the github repository:

```
$ git clone https://github.com/open2c/pairtools
```

Then, you can compile and install *pairtools* in the development mode, which installs the package without moving it to a system folder and thus allows immediate live-testing any changes in the python code. Please, make sure that you have *cython* installed!

```
$ cd pairtools
$ pip install -e ./
```

CHAPTER 3

Parsing sequence alignments into Hi-C pairs

3.1 Overview

Hi-C experiments aim to measure the frequencies of contacts between all pairs of loci in the genome. In these experiments, the spacial structure of chromosomes is first fixed with formaldehyde crosslinks, after which DNA is partially digested with restriction enzymes and then re-ligated back. Then, DNA is shredded into smaller pieces, released from the nucleus, sequenced and aligned to the reference genome. The resulting sequence alignments reveal if DNA molecules were formed through ligations between DNA from different locations in the genome. These ligation events imply that ligated loci were close to each other when the ligation enzyme was active, i.e. they formed "a contact".

`pairtools parse` detects ligation events in the aligned sequences of DNA molecules formed in Hi-C experiments and reports them in the .pairs/.pairsam format.

3.2 Terminology

Throughout this document we will be using the same visual language to describe how DNA sequences (in the .fastq format) are transformed into sequence alignments (.sam/.bam) and into ligation events (.pairs).

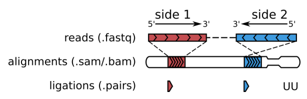

Fig. 1: DNA sequences (reads) are aligned to the reference genome and converted into ligation events

Short-read sequencing determines the sequences of the both ends (or, **sides**) of DNA molecules (typically 50-300 bp), producing **read pairs** in .fastq format (shown in the first row on the figure above). In such reads, base pairs are reported from the tips inwards, which is also defined as the **5'->3'** direction (in accordance of the 5'->3' direction of the DNA strand that sequence of the corresponding side of the read).

Alignment software maps both reads of a pair to the reference genome, producing **alignments**, i.e. segments of the reference genome with matching sequences. Typically, if the read length is not very large (< 150 bp), there will be only two alignments per read pair, one on each side. But, sometimes, the parts of one or both sides may map to different locations on the genome, producing more than two alignments per DNA molecule (see *Multiple ligations (walks)*).

`pairtools parse` converts alignments into **ligation events** (aka **Hi-C pairs** aka **pairs**). In the simplest case, when each side has only one unique alignment (i.e. the whole side maps to a single unique segment of the genome), for each side, we report the chromosome, the genomic position of the outer-most (5') aligned base pair and the strand of the reference genome that the read aligns to. `pairtools parse` assigns to such pairs the type `UU` (unique-unique).

3.3 Unmapped/multimapped reads

Sometimes, one side or both sides of a read pair may not align to the reference genome:

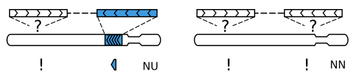

Fig. 2: Read pairs missing an alignment on one or both sides

In this case, `pairtools parse` fills in the chromosome of the corresponding side of Hi-C pair with `!`, the position with `0` and the strand with `-`. Such pairs are reported as type `NU` (null-unique, when the other side has a unique alignment) or `NN` (null-null, when both sides lack any alignment).

Similarly, when one or both sides map to many genome locations equally well (i.e. have non-unique, or, multi-mapping alignments), `pairtools parse` reports the corresponding sides as (chromosome= `!`, position= `0`, strand= `-`) and type `MU` (multi-unique) or `MM` (multi-multi) or `NM` (null-multi), depending on the type of the alignment on the other side.

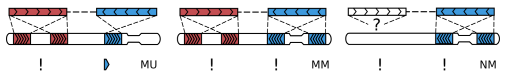

Fig. 3: Read pairs with a non-unique (multi-) alignment on one side

`pairtools parse` calls an alignment to be multi-mapping when its MAPQ score (which depends on the scoring gap between the two best candidate alignments for a segment) is equal or greater than the value specied with the `--min-mapq` flag (by default, 1).

3.4 Multiple ligations (walks)

If the read is long enough (e.g. larger than 150 bp), it may contain more than two alignments:

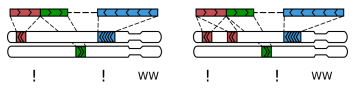

Fig. 4: A sequenced Hi-C molecule that was formed via multiple ligations

Molecules like these typically form via multiple ligation events and we call them walks[1]. The mode of walks reporting is controlled by `--walks-policy` parameter of `pairtools parse`.

A pair of sequential alignments on a single read is **ligation junction**. Ligation junctions are the Hi-C contacts that have been directly observed in the experiment. They are reported in lower-case letters if walks policy is set to `all` (default). For details, wee *Rescuing complex ligations*

[1] Following the lead of C-walks

However, traditional Hi-C pairs do not have direct evidence of ligation because they arise from read pairs that do not necessarily contain ligation junction. To filter out the molecules with complex walks, `--walks-policy` can be set to:

- `mask` to tag these molecules as type `WW` (single ligations are rescued, see *Rescuing single ligations*) ,
- `5any` to report the 5'-most alignment on each side,
- `5unique` to report the 5'-most unique alignment on each side,
- `3any` to report the 3'-most alignment on each side,
- `3unique` to report the 3'-most unique alignment on each side.

3.5 Rescuing complex ligations

The complex walks are DNA molecules containing more than one ligation junction that may end up in more than one alignment on forward, reverse, or both reads:

Fig. 5: Different modes of reporting complex walks

`pairtools parse` detects such molecules and **rescues** them with `--walks-policy all`.

Briefly, the algorithm of complex ligation walks rescue detects all the unique ligation junctions, and do not report the same junction as a pair multiple times. Importantly, these duplicated pairs might arise when both forward and reverse reads read through the same ligation junction. However, these cases are successfully merged by `pairtools parse`:

Fig. 6: Reporing complex walks in case of readthrough

To restore the sequence of ligation events, there is a special field `junction_index` that can be reported as a separate column of .pair file by setting `--add-junction-index`. This field contains information on:

- the order of the junction in the recovered walk, starting from 5'-end of forward read
- type of the junction:
  - "u" - unconfirmed junction, right and left alignments in the pair originate from different reads (forward or reverse). This might be indirect ligation (mediated by other DNA fragments).
  - "f" - pair originates from the forward read. This is direct ligation.
  - "r" - pair originated from the reverse read. Direct ligation.
  - "b" - pair was sequenced at both forward and reverse read. Direct ligation.

With this information, the whole sequence of ligation events can be restored from the .pair file.

3.6 Rescuing single ligations

Importantly, some of DNA molecules containing only one ligation junction may still end up with three alignments:

A molecule formed via a single ligation gets three alignments when one of the two ligated DNA pieces is shorter than the read length, such that that read on the corresponding side sequences through the ligation junction and into the

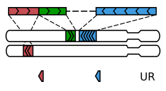

Fig. 7: Not all read pairs with three alignments come from "walks"

other piece[2]. The amount of such molecules depends on the type of the restriction enzyme, the typical size of DNA molecules in the Hi-C library and the read length, and sometimes can be considerable.

`pairtools parse` detects such molecules and **rescues** them (i.e. changes their type from a *walk* to a single-ligation molecule). It tests walks with three aligments using three criteria:

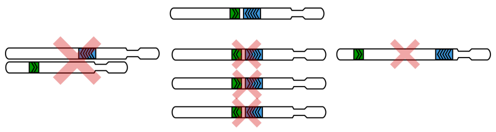

Fig. 8: The three criteria used to "rescue" three-alignment walks: cis, point towards each other, short distance

1. On the side with two alignments (the **chimeric** side), the "inner" (or, 3') alignment must be on the same chromosome as the alignment on the non-chimeric side.
2. The "inner" alignment on the chimeric side and the alignment on the non-chimeric side must point toward each other.
3. These two alignments must be within the distance specified with the `--max-molecule-size` flag (by default, 2000bp).

Sometimes, the "inner" alignment on the chimeric side can be non-unique or "null" (i.e. when the unmapped segment is longer than `--max-inter-align-gap`, as described in *Interpreting gaps between alignments*). `pairtools parse` ignores such alignments altogether and thus rescues such *walks* as well.

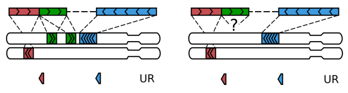

Fig. 9: A walk with three alignments get rescued, when the middle alignment is multi- or null.

3.7 Interpreting gaps between alignments

Reads that are only partially aligned to the genome can be interpreted in two different ways. One possibility is to assume that this molecule was formed via at least two ligations (i.e. it's a *walk*) but the non-aligned part (a **gap**) was missing from the reference genome for one reason or another. Another possibility is to simply ignore this gap (for example, because it could be an insertion or a technical artifact), thus assuming that our molecule was formed via a single ligation and has to be reported:

Both options have their merits, depending on a dataset, quality of the reference genome and sequencing. `pairtools parse` ignores shorter *gaps* and keeps longer ones as "null" alignments. The maximal size of ignored *gaps* is set by the `--max-inter-align-gap` flag (by default, 20bp).

[2] This procedure was first introduced in HiC-Pro and the in Juicer .

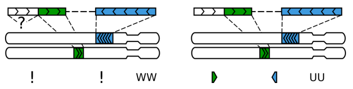

Fig. 10: A gap between alignments can interpeted as a legitimate segment without an alignment or simply ignored

CHAPTER 4

Sorting pairs

In order to enable efficient random access to Hi-C pairs, we **flip** and **sort** pairs. After sorting, interactions become arranged in the order of their genomic position, such that, for any given pair of regions, we easily find and extract all of their interactions. And, after flipping, all artificially duplicated molecules (either during PCR or in optical sequencing) end up in adjacent rows in sorted lists of interactions, such that we can easily identify and remove them.

4.1 Sorting

`pairtools sort` arrange pairs in the order of (chrom1, chrom2, pos1, pos2). This order is also known as *block sorting*, because all pairs between any given pair of chromosomes become grouped into one continuous block. Additionally, `pairtools sort` also sorts pairs with identical positions by *pair_type*. This does not really do much for mapped reads, but it nicely splits unmapped reads into blocks of null-mapped and multi-mapped reads.

We note that there is an alternative to block sorting, called *row sorting*, where pairs are sorted by (chrom1, pos1, chrom2, pos2). In *pairtools sort*, we prefer block-sorting since it cleanly separates cis interactions from trans ones and thus is a more optimal solution for typical use cases.

4.2 Flipping

In a typical paired-end experiment, *side1* and *side2* of a DNA molecule are defined by the order in which they got sequenced. Since this order is essentially random, any given Hi-C pair, e.g. (chr1, 1.1Mb; chr2, 2.1Mb), may appear in a reversed orientation, i.e. (chr2, 2.1Mb; chr1, 1.1Mb). If we were to preserve this order of sides, interactions between same loci would appear in two different locations of the sorted pair list, which would complicate finding PCR/optical duplicates.

To ensure that Hi-C pairs with similar coordinates end up in the same location of the sorted list, we **flip** pairs, i.e. we choose *side1* as the side with the lowest genomic coordinate. Thus, after flipping, for *trans* pairs (chrom1!=chrom2), order(chrom1)<order(chrom2); and for *cis* pairs (chrom1==chrom2), pos1<=pos2. In a matrix representation, flipping is equal to reflecting the lower triangle of the Hi-C matrix onto its upper triangle, such that the resulting matrix is upper-triangular.

In *pairtools*, flipping is done during parsing - that's why `pairtools parse` requires a .chromsizes file that specifies the order of chromosomes for flipping. Importantly, `pairtools parse` also flips one-sided pairs such that side1 is always unmapped; and unmapped pairs such that side1 always has a "poorer" mapping type (i.e. null-mapping<multi-mapping).

4.3 Chromosomal order for sorting and flipping

Importantly, the order of chromosomes for sorting and flipping can be different. Specifically, `pairtools sort` uses the lexicographic order for chromosomes (chr1, chr10, chr11, . . . , chr2, chr21,. . .) instead of the "natural" order (chr1, chr2, chr3, . . .); at the same time, flipping is done in `pairsamtools parse` using the chromosomal order specified by the user.

`pairtools sort` uses the lexicographic order for sorting chromosomes. This order is used universally to sorting strings in all languages and tools[1], which makes it easy to design tools for indexing and searching in sorted pair lists.

At the same time, `pairtools parse` uses a custom user-provided chromosomal order to flip pairs. For performance considerations, for flipping, we recommend ordering chromosomes in a way that will be used in plotting contact maps.

[1] Unfortunately, many existing genomes use rather unconventional choices in chromosomal naming schemes. For example, in sacCer3, chromosomes are enumerated with Roman numerals; in dm3, big autosomes are split 4 different contigs each. Thus, it is impossible to design a universal algorithm that would order chromosomes in a "meaningful" way for all existing genomes.

CHAPTER 5

Formats for storing Hi-C pairs

5.1 .pairs

.pairs is a simple tabular format for storing DNA contacts detected in a Hi-C experiment. The detailed .pairs specification is defined by the 4DN Consortium.

The body of a .pairs contains a table with a variable number of fields separated by a "\t" character (a horizontal tab). The .pairs specification fixes the content and the order of the first seven columns:

index	name	description
1	read_id	the ID of the read as defined in fastq files
2	chrom1	the chromosome of the alignment on side 1
3	pos1	the 1-based genomic position of the outer-most (5') mapped bp on side 1
4	chrom2	the chromosome of the alignment on side 2
5	pos2	the 1-based genomic position of the outer-most (5') mapped bp on side 2
6	strand1	the strand of the alignment on side 1
7	strand2	the strand of the alignment on side 2

A .pairs file starts with a header, an arbitrary number of lines starting with a "#" character. By convention, the header lines have a format of "#field_name: field_value". The .pairs specification mandates a few standard header lines (e.g., column names, chromosome order, sorting order, etc), all of which are automatically filled in by *pairtools*.

The entries of a .pairs file can be flipped and sorted. "Flipping" means that *the sides 1 and 2 do not correspond to side1 and side2 in sequencing data.* Instead, side1 is defined as the side with the alignment with a lower sorting index (using the lexographic order for chromosome names, followed by the numeric order for positions and the lexicographic order for pair types). This particular order of "flipping" is defined as "upper-triangular flipping", or "triu-flipping". Finally, pairs are *typically* block-sorted: i.e. first lexicographically by chrom1 and chrom2, then numerically by pos1 and pos2.

5.2 Pairtools' flavor of .pairs

.pairs files produced by *pairtools* extend .pairs format in a few ways.

1. *pairtools* store null/ambiguous/chimeric alignments as chrom='!', pos=0, strand='-'.
2. *pairtools* store the header of the source .sam files in the '#samheader:' fields of the pairs header. When multiple .pairs files are merged, the respective '#samheader:' fields are checked for consistency and merged.
3. Each pairtool applied to .pairs leaves a record in the '#samheader' fields (using a @PG sam tag), thus preserving the full history of data processing.
4. *pairtools* append an extra column describing the type of a Hi-C pair:

index	name	description
8	pair_type	the type of a Hi-C pair

5.3 Pair types

pairtools use a simple two-character notation to define all possible pair types, according to the quality of alignment of the two sides. The type of a pair can be defined unambiguously using the table below. To use this table, identify which side has an alignment of a "poorer" quality (unmapped < multimapped < unique alignment) and which side has a "better" alignment and find the corresponding row in the table.

.	.	.	Less informative alignment		More informative alignment		.
Pair type	Code	>2 alignments	Mapped	Unique	Mapped	Unique	Sidedness
walk-walk	WW	✓					0[1]
null	NN						0
corrupt	XX						0 **[2]_**
null-multi	NM				✓		0
null-rescued	NR	✓			✓	✓	1[3]
null-unique	NU				✓	✓	1
multi-multi	MM		✓		✓		0
multi-rescued	MR	✓	✓		✓	✓	1[3]
multi-unique	MU		✓		✓	✓	1
rescued-unique	RU	✓	✓	✓	✓	✓	2[3]
unique-rescued	UR	✓	✓	✓	✓	✓	2[3]
unique-unique	UU		✓	✓	✓	✓	2
duplicate	DD		✓	✓	✓	✓	2 **[4]_**

5.4 .pairsam

pairtools also define .pairsam, a valid extension of the .pairs format. On top of the pairtools' flavor of .pairs, .pairsam format adds two extra columns containing the alignments from which the Hi-C pair was extracted:

[1] "walks", or, C-walks are Hi-C molecules formed via multiple ligation events which cannot be reported as a single pair.

[3] *pairtools dedup* detects molecules that could be formed via PCR duplication and tags them as "DD" pair type. These pairs should be excluded from downstream analyses.

index	name	description
9	sam1	the sam alignment(s) on side 1; separate supplemental alignments by NEXT_SAM
10	sam2	the sam alignment(s) on side 2; separate supplemental alignments by NEXT_SAM

Note that, normally, the fields of a sam alignment are separated by a horizontal tab character (\t), which we already use to separate .pairs columns. To avoid confusion, we replace the tab character in sam entries stored in sam1 and sam2 columns with a UNIT SEPARATOR character (\031).

Finally, sam1 and sam2 can store multiple .sam alignments, separated by a string '\031NEXT_SAM\031'

CHAPTER 6

Technical notes

Designing scientific software and formats requires making a multitude of tantalizing technical decisions and compromises. Often, the reasons behind a certain decision are non-trivial and convoluted, involving many factors. Here, we collect the notes and observations made during the desing stage of *pairtools* and provide a justification for most non-trivial decisions. We hope that this document will elucidate the design of *pairtools* and may prove useful to developers in their future projects.

6.1 .pairs format

The motivation behind some of the technical decisions in the pairtools' flavor of .pairs/.pairsam:

- *pairtools* can store SAM entries together with the Hi-C pair information in .pairsam files. Storing pairs and alignments in the same row enables easy tagging and filtering of paired-end alignments based on their Hi-C information.
- *pairtools* use the exclamation mark "!" instead of '.' as 'chrom' of unmapped reads because it has the lowest lexicographic sorting order among all characters. The use of '0' and '-' in the 'pos' and 'strand' fields of unmapped reads allows us to keep the types of these fields as 'unsigned int' and enum{'+','-'}, respectively.
- "rescued" pairs have two types "UR" and "RU" instead of just "RU". We chose this design because rescued pairs are two-sided and thus are flipped based on (chrom, pos), and not based on the side types. With two pair types "RU" and "UR", *pairtools* can keep track of which side of the pair was rescued.
- in "rescued" pairs, the type "R" is assigned to the non-chimeric side. This may seem counter-intuitive at first, since it is the chimeric side that gets rescued, but this way *pairtools* can keep track of the type of the 5' alignment on the chimeric side (the alignment on the non-chimeric side has to be unique for the pair to be rescued).
- *pairtools* rely on a text format, .pairs, instead of hdf5/parquet-based tables or custom binaries. We went with a text format for a few reasons:
  - text tables enable easy access to data from any language and any tool. This is especially important at the level of Hi-C pairs, the "rawest" format of information from a Hi-C experiment.
  - hdf5 and parquet have a few shortcomings that hinder their immediate use in *pairtools*. Specifically, hdf5 cannot compress variable-length strings (which are, in turn, required to store sam alignments and some

  optional information on pairs) and parquet cannot append columns to existing files, modify datasets in place or store multiple tables in one file (which is required to keep table indices in the same file with pairs).

  - text tables have a set of well-developed and highly-optimized tools for sorting (Unix sort), compression (bgzip/lz4) and random access (tabix).
  - text formats enable easy streaming between individual command-line tools.

Having said that, text formats have many downsides - they are bulky when not compressed, compression and parsing requires extra computational resources, they cannot be modified in place and random access requires extra tools. In the future, we plan to develop a binary format based on existing container formats, which would mitigate these downsides.

6.2 CLI

- many *pairtools* perform multiple actions at once, which contradicts the "do one thing" philosophy of Unix command line. We packed multiple (albeit, related) functions into one tool to improve the performance of *pairtools*. Specifically, given the large size of Hi-C data, a significant fraction of time is spent on compression/decompression, parsing, loading data into memory and sending it over network (for cloud/clusters). Packing multiple functions into one tool cuts down the amount of such time consuming operations.
- `pairtools parse` requires a .chromsizes file to know the order of chromosomes and perform pair flipping.
- *pairtools* use bgzip compression by default instead of gzip. Using *bgzip* allows us to create an index with pairix and get random access to data.
- *paritools* have an option to compress outputs with lz4. Lz4 is much faster and only slighly less efficient than gzip. This makes lz4 a better choice for passing data between individual pairtools before producing final result (which, in turn, requires bgzip compression).

CHAPTER 7

Command-line API

7.1 pairtools

Flexible tools for Hi-C data processing.

All pairtools have a few common options, which should be typed _before_ the command name.

```
pairtools [OPTIONS] COMMAND [ARGS]...
```

Options

--post-mortem
Post mortem debugging

--output-profile <output_profile>
Profile performance with Python cProfile and dump the statistics into a binary file

--version
Show the version and exit.

7.1.1 dedup

Find and remove PCR/optical duplicates.

Find PCR duplicates in an upper-triangular flipped sorted pairs/pairsam file. Allow for a +/-N bp mismatch at each side of duplicated molecules.

PAIRS_PATH : input triu-flipped sorted .pairs or .pairsam file. If the path ends with .gz/.lz4, the input is decompressed by bgzip/lz4c. By default, the input is read from stdin.

```
pairtools dedup [OPTIONS] [PAIRS_PATH]
```

Options

-o, --output <output>
output file for pairs after duplicate removal. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, the output is printed into stdout.

--output-dups <output_dups>
output file for duplicated pairs. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. If the path is the same as in –output or -, output duplicates together with deduped pairs. By default, duplicates are dropped.

--output-unmapped <output_unmapped>
output file for unmapped pairs. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. If the path is the same as in –output or -, output unmapped pairs together with deduped pairs. If the path is the same as –output-dups, output unmapped reads together with dups. By default, unmapped pairs are dropped.

--output-stats <output_stats>
output file for duplicate statistics. If file exists, it will be open in the append mode. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, statistics are not printed.

--max-mismatch <max_mismatch>
Pairs with both sides mapped within this distance (bp) from each other are considered duplicates. [default: 3]

--method <method>
define the mismatch as either the max or the sum of the mismatches ofthe genomic locations of the both sides of the two compared molecules [default: max]

--sep <sep>
Separator (t, v, etc. characters are supported, pass them in quotes)

--comment-char <comment_char>
The first character of comment lines

--send-header-to <send_header_to>
Which of the outputs should receive header and comment lines

--c1 <c1>
Chrom 1 column; default 1

--c2 <c2>
Chrom 2 column; default 3

--p1 <p1>
Position 1 column; default 2

--p2 <p2>
Position 2 column; default 4

--s1 <s1>
Strand 1 column; default 5

--s2 <s2>
Strand 2 column; default 6

--unmapped-chrom <unmapped_chrom>
Placeholder for a chromosome on an unmapped side; default !

--mark-dups
If specified, duplicate pairs are marked as DD in “pair_type” and as a duplicate in the sam entries.

--extra-col-pair <extra_col_pair>
Extra columns that also must match for two pairs to be marked as duplicates. Can be either provided as 0-based

column indices or as column names (requires the “#columns” header field). The option can be provided multiple times if multiple column pairs must match. Example: –extra-col-pair “phase1” “phase2”

--nproc-in `<nproc_in>`
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out `<nproc_out>`
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in `<cmd_in>`
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out `<cmd_out>`
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

PAIRS_PATH
Optional argument

7.1.2 filterbycov

Remove pairs from regions of high coverage.

Find and remove pairs with >(MAX_COV-1) neighbouring pairs within a +/- MAX_DIST bp window around either side. Useful for single-cell Hi-C experiments, where coverage is naturally limited by the chromosome copy number.

PAIRS_PATH : input triu-flipped sorted .pairs or .pairsam file. If the path ends with .gz/.lz4, the input is decompressed by bgzip/lz4c. By default, the input is read from stdin.

```
pairtools filterbycov [OPTIONS] [PAIRS_PATH]
```

Options

-o, --output `<output>`
output file for pairs from low coverage regions. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, the output is printed into stdout.

--output-highcov `<output_highcov>`
output file for pairs from high coverage regions. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. If the path is the same as in –output or -, output duplicates together with deduped pairs. By default, duplicates are dropped.

--output-unmapped `<output_unmapped>`
output file for unmapped pairs. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. If the path is the same as in –output or -, output unmapped pairs together with deduped pairs. If the path is the same as –output-highcov, output unmapped reads together. By default, unmapped pairs are dropped.

--output-stats `<output_stats>`
output file for statistics of multiple interactors. If file exists, it will be open in the append mode. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, statistics are not printed.

--max-cov `<max_cov>`
The maximum allowed coverage per region.

--max-dist `<max_dist>`
The resolution for calculating coverage. For each pair, the local coverage around each end is calculated as (1 + the number of neighbouring pairs within +/- max_dist bp)

--method `<method>`
calculate the number of neighbouring pairs as either the sum or the max of the number of neighbours on the two sides [default: max]

--sep `<sep>`
Separator (t, v, etc. characters are supported, pass them in quotes)

--comment-char `<comment_char>`
The first character of comment lines

--send-header-to `<send_header_to>`
Which of the outputs should receive header and comment lines

--c1 `<c1>`
Chrom 1 column; default 1

--c2 `<c2>`
Chrom 2 column; default 3

--p1 `<p1>`
Position 1 column; default 2

--p2 `<p2>`
Position 2 column; default 4

--s1 `<s1>`
Strand 1 column; default 5

--s2 `<s2>`
Strand 2 column; default 6

--unmapped-chrom `<unmapped_chrom>`
Placeholder for a chromosome on an unmapped side; default !

--mark-multi
If specified, duplicate pairs are marked as FF in "pair_type" and as a duplicate in the sam entries.

--nproc-in `<nproc_in>`
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out `<nproc_out>`
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in `<cmd_in>`
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out `<cmd_out>`
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

PAIRS_PATH
Optional argument

7.1.3 flip

Flip pairs to get an upper-triangular matrix.

Change the order of side1 and side2 in pairs, such that (order(chrom1) < order(chrom2)

or (order(chrom1) == order(chrom2)) and (pos1 <=pos2))

Equivalent to reflecting the lower triangle of a Hi-C matrix onto its upper triangle, resulting in an upper triangular matrix. The order of chromosomes must be provided via a .chromsizes file.

PAIRS_PATH : input .pairs/.pairsam file. If the path ends with .gz or .lz4, the input is decompressed by bgzip/lz4c. By default, the input is read from stdin.

```
pairtools flip [OPTIONS] [PAIRS_PATH]
```

Options

-c, --chroms-path `<chroms_path>`
Chromosome order used to flip interchromosomal mates: path to a chromosomes file (e.g. UCSC chrom.sizes or similar) whose first column lists scaffold names. Any scaffolds not listed will be ordered lexicographically following the names provided. [required]

-o, --output `<output>`
output file. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, the output is printed into stdout.

--nproc-in `<nproc_in>`
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out `<nproc_out>`
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in `<cmd_in>`
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out `<cmd_out>`
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

PAIRS_PATH
Optional argument

7.1.4 markasdup

Tag pairs as duplicates.

Change the type of all pairs inside a .pairs/.pairsam file to DD. If sam entries are present, change the pair type in the Yt SAM tag to ‘Yt:Z:DD’.

PAIRSAM_PATH : input .pairs/.pairsam file. If the path ends with .gz, the input is gzip-decompressed. By default, the input is read from stdin.

```
pairtools markasdup [OPTIONS] [PAIRSAM_PATH]
```

Options

-o, --output `<output>`
output .pairsam file. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, the output is printed into stdout.

--nproc-in `<nproc_in>`
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out `<nproc_out>`
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in `<cmd_in>`
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out `<cmd_out>`
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

PAIRSAM_PATH
Optional argument

7.1.5 merge

Merge sorted .pairs/.pairsam files.

Merge triu-flipped sorted pairs/pairsam files. If present, the @SQ records of the SAM header must be identical; the sorting order of these lines is taken from the first file in the list. The ID fields of the @PG records of the SAM header are modified with a numeric suffix to produce unique records. The other unique SAM and non-SAM header lines are copied into the output header.

PAIRS_PATH : upper-triangular flipped sorted .pairs/.pairsam files to merge or a group/groups of .pairs/.pairsam files specified by a wildcard. For paths ending in .gz/.lz4, the files are decompressed by bgzip/lz4c.

```
pairtools merge [OPTIONS] [PAIRS_PATH]...
```

Options

-o, --output `<output>`
output file. If the path ends with .gz/.lz4, the output is compressed by bgzip/lz4c. By default, the output is printed into stdout.

--max-nmerge `<max_nmerge>`
The maximal number of inputs merged at once. For more, store merged intermediates in temporary files. [default: 8]

--tmpdir `<tmpdir>`
Custom temporary folder for merged intermediates.

--memory `<memory>`
The amount of memory used by default. [default: 2G]

--compress-program `<compress_program>`
A binary to compress temporary merged chunks. Must decompress input when the flag -d is provided. Suggested alternatives: lz4c, gzip, lzop, snzip. NOTE: fails silently if the command syntax is wrong. [default:]

--nproc `<nproc>`
Number of threads for merging. [default: 8]

--nproc-in `<nproc_in>`
Number of processes used by the auto-guessed input decompressing command. [default: 1]

--nproc-out `<nproc_out>`
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in `<cmd_in>`
A command to decompress the input. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out `<cmd_out>`
A command to compress the output. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

PAIRS_PATH
Optional argument(s)

7.1.6 parse

Find ligation junctions in .sam, make .pairs. SAM_PATH : an input .sam/.bam file with paired-end sequence alignments of Hi-C molecules. If the path ends with .bam, the input is decompressed from bam with samtools. By default, the input is read from stdin.

```
pairtools parse [OPTIONS] [SAM_PATH]
```

Options

-c, --chroms-path `<chroms_path>`
Chromosome order used to flip interchromosomal mates: path to a chromosomes file (e.g. UCSC chrom.sizes or similar) whose first column lists scaffold names. Any scaffolds not listed will be ordered lexicographically following the names provided. [required]

-o, --output `<output>`
output file. If the path ends with .gz or .lz4, the output is bgzip-/lz4-compressed.By default, the output is printed into stdout.

--assembly `<assembly>`
Name of genome assembly (e.g. hg19, mm10) to store in the pairs header.

--min-mapq `<min_mapq>`
The minimal MAPQ score to consider a read as uniquely mapped [default: 1]

--max-molecule-size `<max_molecule_size>`
The maximal size of a Hi-C molecule; used to rescue single ligations(from molecules with three alignments)

and to rescue complex ligations.The default is based on oriented P(s) at short ranges of multiple Hi-C. [default: 750]

--drop-readid
If specified, do not add read ids to the output

--drop-seq
If specified, remove sequences and PHREDs from the sam fields

--drop-sam
If specified, do not add sams to the output

--add-junction-index
If specified, each pair will have junction index in the molecule

--add-columns <add_columns>
Report extra columns describing alignments Possible values (can take multiple values as a comma-separated list): a SAM tag (any pair of uppercase letters) or mapq, pos5, pos3, cigar, read_len, matched_bp, algn_ref_span, algn_read_span, dist_to_5, dist_to_3, seq.

--output-parsed-alignments <output_parsed_alignments>
output file for all parsed alignments, including walks. Useful for debugging and rnalysis of walks. If file exists, it will be open in the append mode. If the path ends with .gz or .lz4, the output is bgzip-/lz4-compressed. By default, not used.

--output-stats <output_stats>
output file for various statistics of pairs file. By default, statistics is not generated.

--report-alignment-end <report_alignment_end>
specifies whether the 5' or 3' end of the alignment is reported as the position of the Hi-C read.

--max-inter-align-gap <max_inter_align_gap>
read segments that are not covered by any alignment and longer than the specified value are treated as "null" alignments. These null alignments convert otherwise linear alignments into walks, and affect how they get reported as a Hi-C pair (see –walks-policy). [default: 20]

--walks-policy <walks_policy>
the policy for reporting unrescuable walks (reads containing more than one alignment on one or both sides, that can not be explained by a single ligation between two mappable DNA fragments). "mask" - mask walks (chrom="!", pos=0, strand="-"); "all" - report all pairs of consecutive alignments; "5any" - report the 5'-most alignment on each side; "5unique" - report the 5'-most unique alignment on each side, if present; "3any" - report the 3'-most alignment on each side; "3unique" - report the 3'-most unique alignment on each side, if present. [default: mask]

--readid-transform <readid_transform>
A Python expression to modify read IDs. Useful when read IDs differ between the two reads of a pair. Must be a valid Python expression that uses variables called readID and/or i (the 0-based index of the read pair in the bam file) and returns a new value, e.g. "readID[:-2]+'_'+str(i)". Make sure that transformed readIDs remain unique!

--no-flip
If specified, do not flip pairs in genomic order and instead preserve the order in which they were sequenced.

--nproc-in <nproc_in>
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out <nproc_out>
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in <cmd_in>
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out <cmd_out>
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

SAM_PATH
Optional argument

7.1.7 phase

Phase pairs mapped to a diploid genome.

PAIRS_PATH : input .pairs/.pairsam file. If the path ends with .gz or .lz4, the input is decompressed by bgzip/lz4c. By default, the input is read from stdin.

```
pairtools phase [OPTIONS] [PAIRS_PATH]
```

Options

-o, --output <output>
output file. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, the output is printed into stdout.

--phase-suffixes <phase_suffixes>
phase suffixes.

--clean-output
drop all columns besides the standard ones and phase1/2

--nproc-in <nproc_in>
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out <nproc_out>
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in <cmd_in>
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out <cmd_out>
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

PAIRS_PATH
Optional argument

7.1.8 restrict

Assign restriction fragments to pairs.

Identify the restriction fragments that got ligated into a Hi-C molecule.

PAIRS_PATH : input .pairs/.pairsam file. If the path ends with .gz/.lz4, the input is decompressed by bgzip/lz4c. By default, the input is read from stdin.

```
pairtools restrict [OPTIONS] [PAIRS_PATH]
```

Options

-f, --frags <frags>
a tab-separated BED file with the positions of restriction fragments (chrom, start, end). Can be generated using cooler digest. [required]

-o, --output <output>
output .pairs/.pairsam file. If the path ends with .gz/.lz4, the output is compressed by bgzip/lz4c. By default, the output is printed into stdout.

--nproc-in <nproc_in>
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out <nproc_out>
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in <cmd_in>
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out <cmd_out>
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

PAIRS_PATH
Optional argument

7.1.9 sample

Select a random subset of pairs in a pairs file.

FRACTION: the fraction of the randomly selected pairs subset

PAIRS_PATH : input .pairs/.pairsam file. If the path ends with .gz or .lz4, the input is decompressed by bgzip/lz4c. By default, the input is read from stdin.

```
pairtools sample [OPTIONS] FRACTION [PAIRS_PATH]
```

Options

-o, --output <output>
output file. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, the output is printed into stdout.

-s, --seed <seed>
the seed of the random number generator.

--nproc-in <nproc_in>
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out <nproc_out>
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in <cmd_in>
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out <cmd_out>
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

FRACTION
Required argument

PAIRS_PATH
Optional argument

7.1.10 select

Select pairs according to some condition.

CONDITION : A Python expression; if it returns True, select the read pair. Any column declared in the #columns line of the pairs header can be accessed by its name. If the header lacks the #columns line, the columns are assumed to follow the .pairs/.pairsam standard (readID, chrom1, chrom2, pos1, pos2, strand1, strand2, pair_type). Finally, CONDITION has access to COLS list which contains the string values of columns. In Bash, quote CONDITION with single quotes, and use double quotes for string variables inside CONDITION.

PAIRS_PATH : input .pairs/.pairsam file. If the path ends with .gz or .lz4, the input is decompressed by bgzip/lz4c. By default, the input is read from stdin.

The following functions can be used in CONDITION besides the standard Python functions:

- csv_match(x, csv) - True if variable x is contained in a list of

comma-separated values, e.g. csv_match(chrom1, 'chr1,chr2')

- wildcard_match(x, wildcard) - True if variable x matches a wildcard,

e.g. wildcard_match(pair_type, 'C*')

- regex_match(x, regex) - True if variable x matches a Python-flavor regex,

e.g. regex_match(chrom1, 'chrd')

Examples:
pairtools select '(pair_type=="UU") or (pair_type=="UR") or (pair_type=="RU")'
pairtools select 'chrom1==chrom2'
pairtools select 'COLS[1]==COLS[3]'
pairtools select '(chrom1==chrom2) and (abs(pos1 - pos2) < 1e6)'
pairtools select '(chrom1=="!") and (chrom2!="!")'
pairtools select 'regex_match(chrom1, "chrd+") and regex_match(chrom2, "chrd+")'

pairtools select 'True' –chr-subset mm9.reduced.chromsizes

```
pairtools select [OPTIONS] CONDITION [PAIRS_PATH]
```

Options

-o, --output <output>
output file. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, the output is printed into stdout.

--output-rest <output_rest>
output file for pairs of other types. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. By default, such pairs are dropped.

--send-comments-to <send_comments_to>
Which of the outputs should receive header and comment lines [default: both]

--chrom-subset <chrom_subset>
A path to a chromosomes file (tab-separated, 1st column contains chromosome names) containing a chromosome subset of interest. If provided, additionally filter pairs with both sides originating from the provided subset of chromosomes. This operation modifies the #chromosomes: and #chromsize: header fields accordingly.

--startup-code <startup_code>
An auxiliary code to execute before filtering. Use to define functions that can be evaluated in the CONDITION statement

-t, --type-cast <type_cast>
Cast a given column to a given type. By default, only pos and mapq are cast to int, other columns are kept as str. Provide as -t <column_name> <type>, e.g. -t read_len1 int. Multiple entries are allowed.

--nproc-in <nproc_in>
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out <nproc_out>
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in <cmd_in>
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out <cmd_out>
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

CONDITION
Required argument

PAIRS_PATH
Optional argument

7.1.11 sort

Sort a .pairs/.pairsam file.

Sort pairs in the lexicographic order along chrom1 and chrom2, in the numeric order along pos1 and pos2 and in the lexicographic order along pair_type.

PAIRS_PATH : input .pairs/.pairsam file. If the path ends with .gz or .lz4, the input is decompressed by bgzip or lz4c, correspondingly. By default, the input is read as text from stdin.

```
pairtools sort [OPTIONS] [PAIRS_PATH]
```

Options

-o, --output `<output>`
    output pairs file. If the path ends with .gz or .lz4, the output is compressed by bgzip or lz4, correspondingly. By default, the output is printed into stdout.

--nproc `<nproc>`
    Number of processes to split the sorting work between. [default: 8]

--tmpdir `<tmpdir>`
    Custom temporary folder for sorting intermediates.

--memory `<memory>`
    The amount of memory used by default. [default: 2G]

--compress-program `<compress_program>`
    A binary to compress temporary sorted chunks. Must decompress input when the flag -d is provided. Suggested alternatives: gzip, lzop, lz4c, snzip. If "auto", then use lz4c if available, and gzip otherwise. [default: auto]

--nproc-in `<nproc_in>`
    Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out `<nproc_out>`
    Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in `<cmd_in>`
    A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out `<cmd_out>`
    A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

PAIRS_PATH
    Optional argument

7.1.12 split

Split a .pairsam file into .pairs and .sam.

Restore a .sam file from sam1 and sam2 fields of a .pairsam file. Create a .pairs file without sam1/sam2 fields.

PAIRSAM_PATH : input .pairsam file. If the path ends with .gz or .lz4, the input is decompressed by bgzip or lz4c. By default, the input is read from stdin.

```
pairtools split [OPTIONS] [PAIRSAM_PATH]
```

Options

--output-pairs <output_pairs>
output pairs file. If the path ends with .gz or .lz4, the output is bgzip-/lz4c-compressed. If -, pairs are printed to stdout. If not specified, pairs are dropped.

--output-sam <output_sam>
output sam file. If the path ends with .bam, the output is compressed into a bam file. If -, sam entries are printed to stdout. If not specified, sam entries are dropped.

--nproc-in <nproc_in>
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out <nproc_out>
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in <cmd_in>
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out <cmd_out>
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

PAIRSAM_PATH
Optional argument

7.1.13 stats

Calculate pairs statistics.

INPUT_PATH : by default, a .pairs/.pairsam file to calculate statistics. If not provided, the input is read from stdin. If –merge is specified, then INPUT_PATH is interpreted as an arbitrary number of stats files to merge.

The files with paths ending with .gz/.lz4 are decompressed by bgzip/lz4c.

```
pairtools stats [OPTIONS] [INPUT_PATH]...
```

Options

-o, --output <output>
output stats tsv file.

--merge
If specified, merge multiple input stats files instead of calculating statistics of a .pairs/.pairsam file. Merging is performed via summation of all overlapping statistics. Non-overlapping statistics are appended to the end of the file.

--nproc-in <nproc_in>
Number of processes used by the auto-guessed input decompressing command. [default: 3]

--nproc-out <nproc_out>
Number of processes used by the auto-guessed output compressing command. [default: 8]

--cmd-in `<cmd_in>`
A command to decompress the input file. If provided, fully overrides the auto-guessed command. Does not work with stdin. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -dc -n 3

--cmd-out `<cmd_out>`
A command to compress the output file. If provided, fully overrides the auto-guessed command. Does not work with stdout. Must read input from stdin and print output into stdout. EXAMPLE: pbgzip -c -n 8

Arguments

INPUT_PATH
Optional argument(s)

- genindex

Index