epical

Release 0.7.0

Liguo Wang

Jan 12, 2024

CONTENTS

CrossMap

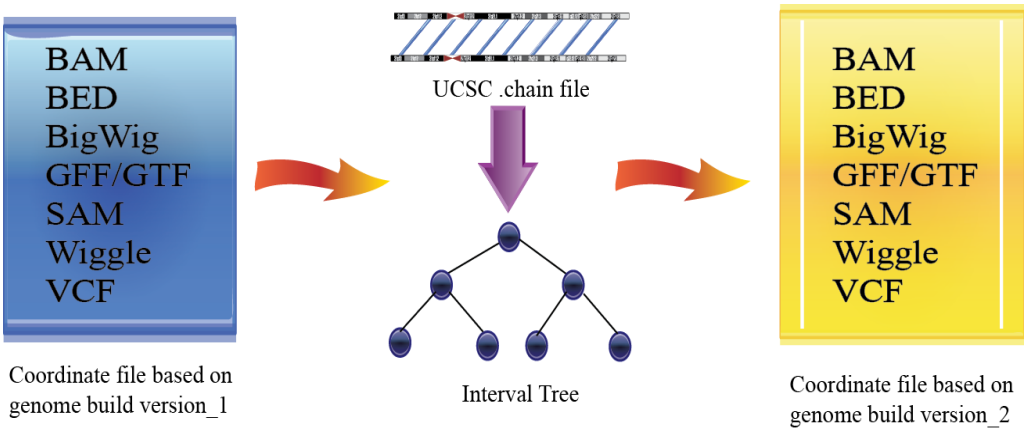
Convert Genome Coordinates Between Assemblies

CrossMap is a program for genome coordinates conversion between *different assemblies* (such as hg18 (NCBI36) <=> hg19 (GRCh37)). It supports commonly used file formats including BAM, CRAM, SAM, Wiggle, BigWig, BED, GFF, GTF, MAF VCF, and gVCF.

CHAPTER

ONE

HOW CROSSMAP WORKS?



CHAPTER

TWO

RELEASE HISTORY

01/11/2024: Release version 0.7.0

1. Fix bugs for VCF varaints liftover.
2. Handle non-DNA ALT alleles such as <DEL>
3. Use pyproject.toml to replace "setup.py".

Note: From v0.7.0 onwards, the main program `CrossMap.py` is now renamed to `CrossMap` due to the restriction on using "." in the script name in the "pyproject.toml" file. To ensure compatibility with the previous pipelines, please include the following line in your `~/.bashrc` file.

```
alias CrossMap.py='CrossMap'
```

07/21/2023: Release version 0.6.4

1. Fix bug when the sequence in BAM file is represented as "*"
2. Change code style

07/12/2022: Release version 0.6.4

1. Fix bug when the input bigwig file does not have coverage signal for some chromosomes.
2. When the input VCF file does not have CONTIG field, use long chromosome ID (e.g., "chr1") as default.

03/04/2022: Release version 0.6.3

1. Fix bug in v0.6.2. "Alternative allele is empty"

02/22/2022: Release version 0.6.2

1. For insertions and deletions, the first nucleotide of the ALT allele (the 5th field in VCF file) is updated to the nucleotide at POS of the reference genome

11/29/2021: Release version 0.6.1

1. Same as v0.6.0. Remove unused modules from the lib folder.

11/16/2021: Release version 0.6.0

1. Use argparse instead of optparse.
2. Use `os.path.getmtime` instead of `os.path.getctime` to check the timestamps of fasta file and its index file.
3. Add '–unmap-file' option to `CrossMap.py bed` command.

4/16/2021: Release version 0.5.3/0.5.4

Add `CrossMap.py viewchain` to convert chain file into block-to-block, more readable format.

12/08/2020: Release version 0.5.2

Add '–no-comp-alleles' flag to `CrossMap.py vcf` and `CrossMap.py gvcf`. If set, CrossMap does not check if the "reference allele" is different from the "alternative allele".

08/19/2020: Release version 0.5.1

In `CrossMap.py region`: keep additional columns (columns after the 3rd column) of the original BED file after conversion.

08/14/2020: Release version 0.5.0

Add `CrossMap.py region` function to convert large genomic regions. Unlike the `CrossMap.py bed` function, which splits big genomic regions, `CrossMap.py region` tries to convert the big genomic region as a whole.

07/09/2020: Release version 0.4.3

Structural Variants VCF files often use INFO/END field to indicate the end of a deletion. v0.4.3 updates "END" coordinate in the INFO field.

05/04/2020: Release version 0.4.2

Support GVCF file conversion.

03/24/2020: Release version 0.4.1

Deal with consecutive TABs in the input MAF file.

10/09/2019: Release version 0.3.8

The University of California holds the copyrights in the UCSC chain files. As requested by UCSC, all UCSC-generated chain files will be permanently removed from this website and the CrossMap distributions.

07/22/2019: Release version 0.3.6

1. Support MAF (mutation annotation format).
2. Fix error "TypeError: AlignmentHeader does not support item assignment (use header.to_dict()" when lifting over BAM files. User does not need to downgrade pysam to 0.13.0 to lift over BAM files.

04/01/2019: Release version 0.3.4

Fix bugs when chromosome IDs (of the source genome) in chain file do not have 'chr' prefix (such as "GRCh37ToHg19.over.chain.gz"). This version also allows CrossMap to detect if a VCF mapping was inverted, and if so, reverse complements the alternative allele (Thanks to Andrew Yates). Improve wording.

01/07/2019: Release version 0.3.3

Version 0.3.3 is exactly the same as Version 0.3.2. The reason to release this version is that CrossMap-0.3.2.tar.gz was broken when uploading to pypi.

12/14/18: Release version 0.3.2

Fix the key error problem (e.g *KeyError: "sequence 'b'7_KI270803v1_alt" not present"*). This error happens when a locus from the original assembly is mapped to an "alternative", "unplaced" or "unlocalized" contig in the target assembly, and this "target contig" does not exist in your target_ref.fa. In version 0.3.2, such loci will be silently skipped and saved to the ".unmap" file.

11/05/18: Release version 0.3.0

1. v0.3.0 or newer will Support Python3. Previous versions support Python2.7.*
2. add pyBigWig as a dependency.

CHAPTER

THREE

INSTALLATION

3.1 Install from PyPI

```
pip3 install CrossMap
```

3.2 Install from GitHub

```
pip3 install git+https://github.com/liguowang/CrossMap.git
```

3.3 Upgrade

```
pip3 install CrossMap --upgrade
```

CHAPTER

FOUR

INPUT AND OUTPUT

4.1 Chain file

A chain file describes a pairwise alignment between two reference assemblies. UCSC and Ensembl chain files are available:

UCSC chain files

- Chain files from hs1 (T2T-CHM13) to hg38/hg19/mm10/mm9 (ore vice versa): https://hgdownload.soe.ucsc.edu/goldenPath/hs1/liftOver/
- Chain files from hg38 (GRCh38) to hg19 and all other organisms: http://hgdownload.soe.ucsc.edu/goldenPath/hg38/liftOver/
- Chain File from hg19 (GRCh37) to hg17/hg18/hg38 and all other organisms: http://hgdownload.soe.ucsc.edu/goldenPath/hg19/liftOver/
- Chain File from mm10 (GRCm38) to mm9 and all other organisms: http://hgdownload.soe.ucsc.edu/goldenPath/mm10/liftOver/

Ensembl chain files

- Human to Human: ftp://ftp.ensembl.org/pub/assembly_mapping/homo_sapiens/
- Mouse to Mouse: ftp://ftp.ensembl.org/pub/assembly_mapping/mus_musculus/
- Other organisms: ftp://ftp.ensembl.org/pub/assembly_mapping/

4.2 User Input file

CrossMap supports the following file formats.

1. BAM, CRAM, or SAM
2. BED or BED-like. (BED file must have at least 'chrom', 'start', 'end')
3. Wiggle ("variableStep", "fixedStep" and "bedGraph" formats are supported)
4. BigWig
5. GFF or GTF
6. VCF
7. GVCF
8. MAF

4.3 Output file

The format of output files depends on the input

Input_format	Output_format
BED	BED (Genome coordinates will be updated)
BAM	BAM (Genome coordinates, header section, all SAM flags, insert size will be updated)
CRAM	BAM (require pysam >= 0.8.2)
SAM	SAM (Genome coordinates, header section, all SAM flags, insert size will be updated)
Wiggle	BigWig
BigWig	BigWig
GFF	GFF (Genome coordinates will be updated to the target assembly)
GTF	GTF (Genome coordinates will be updated to the target assembly)
VCF	VCF (header section, Genome coordinates, reference alleles will be updated)
GVCF	GVCF (header section, Genome coordinates, reference alleles will be updated)
MAF	MAF (Genome coordinates and reference alleles will be updated)

CHAPTER

FIVE

USAGE

Run `CrossMap -h` or `CrossMap --help` print help message

```
$ CrossMap -h

usage: CrossMap [-h] [-v] {bed,bam,gff,wig,bigwig,vcf,gvcf,maf,region,viewchain} ...

CrossMap (v0.6.0) is a program to convert (liftover) genome coordinates between␣
↪different reference
assemblies (e.g., from human GRCh37/hg19 to GRCh38/hg38 or vice versa). Supported file␣
↪formats: BAM,
BED, BigWig, CRAM, GFF, GTF, GVCF, MAF (mutation annotation format), SAM, Wiggle, and␣
↪VCF.

positional arguments:
  {bed,bam,gff,wig,bigwig,vcf,gvcf,maf,region,viewchain}
                        sub-command help
    bed                 converts BED, bedGraph or other BED-like files. Only genome␣
↪coordinates
                        (i.e., the first 3 columns) will be updated. Regions mapped to␣
↪multiple
                        locations to the new assembly will be split. Use the "region"␣
↪command to
                        liftover large genomic regions. Use the "wig" command if you need
                        bedGraph/bigWig output.
    bam                 converts BAM, CRAM, or SAM format file. Genome coordinates,␣
↪header section,
                        all SAM flags, insert size will be updated.
    gff                 converts GFF or GTF format file. Genome coordinates will be␣
↪updated.
    wig                 converts Wiggle or bedGraph format file. Genome coordinates will␣
↪be updated.
    bigwig              converts BigWig file. Genome coordinates will be updated.
    vcf                 converts VCF file. Genome coordinates, header section, reference␣
↪alleles will
                        be updated.
    gvcf                converts GVCF file. Genome coordinates, header section,␣
↪reference alleles
                        will be updated.
    maf                 converts MAF (mutation annotation format) file. Genome␣
↪coordinates and
```

(continues on next page)

(continued from previous page)

```
                        reference alleles will be updated.
    region              converts big genomic regions (in BED format) such as CNV blocks.␣
↪Genome
                        coordinates will be updated.
    viewchain           prints out the content of a chain file into a human readable,␣
↪block-to-block
                        format.

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit

https://crossmap.readthedocs.io/en/latest/
```

5.1 Convert BED format files

A BED (Browser Extensible Data) file is a tab-delimited text file describing genome regions or gene annotations. It consists of one line per feature, each containing 3-12 columns. CrossMap converts BED files with less than 12 columns to a different assembly by updating the chromosome and genome coordinates only; all other columns remain unchanged. Regions from the old assembly mapping to multiple locations to the new assembly will be split. For 12-columns BED files, all columns will be updated accordingly except the 4th column (name of bed line), 5th column (score value), and 9th column (RGB value describing the display color). 12-column BED files usually define multiple blocks (e.g., exons); if any of the exons fails to map to a new assembly, the whole BED line is skipped.

The input BED file can be plain text file, compressed file with extension of .gz, .Z, .z, .bz, .bz2 and .bzip2, or even a URL pointing to accessible remote files (http://, https:// and ftp://). Compressed remote files are not supported. The output is a BED format file with exactly the same number of columns as the original one.

Standard BED format has 12 columns, but CrossMap also supports BED-like formats:

- BED3: The first three columns ("chrom", "start", "end") of the BED format file.
- BED6: The first six columns ("chrom", "start", "end", "name", "score", "strand") of the BED format file.
- Other: Format has at least three columns ("chrom", "start", "end") and no more than 12 columns. All other columns are arbitrary.

Note:

1. For BED-like formats mentioned above, CrossMap only updates the "chrom", "start", "end", and "strand" columns. All other columns will be kept AS-IS.
2. Lines starting with '#', 'browser', 'track' will be skipped.
3. Lines less than three columns will be skipped.
4. The 2nd and 3rd columns must be integers.
5. The "+" strand is assumed if no strand information is found.
6. For standard BED format (12 columns). If any of the defined exon blocks cannot be uniquely mapped to target assembly, the whole entry will be skipped.
7. The "input_chain_file" and "input_bed_file" can be regular or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file, local file or URL (http://, https://, ftp://) pointing to remote file.

8. If the output_file is not specified, results will be printed to screen (console). In this case, the original bed entries (including entries failed to convert) were also printed out.
9. If the input region cannot be consecutively mapped to the target assembly, it will be split.
10. The *.unmap file contains regions that cannot be unambiguously converted.

Typing `CrossMap bed -h` will print help message:

```
$ CrossMap bed  -h

usage: CrossMap bed [-h] [--chromid {a,s,l}] [--unmap-file UNMAP_FILE]
                    input.chain input.bed [out_bed]

positional arguments:
  input.chain           Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html␣
↪describes
                        pairwise alignments between two genomes. The input chain file␣
↪can be a plain
                        text file or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.
  input.bed             The input BED file. The first 3 columns must be “chrom”, “start”,
↪ and “end”.
                        The input BED file can be plain text file, compressed file with␣
↪extension of
                        .gz, .Z, .z, .bz, .bz2 and .bzip2, or even a URL pointing to␣
↪accessible
                        remote files (http://, https:// and ftp://). Compressed remote␣
↪files are not
                        supported.
  out_bed               Output BED file. if argument is missing, CrossMap will write BED␣
↪file to the
                        STDOUT.

optional arguments:
  -h, --help            show this help message and exit
  --chromid {a,s,l}     The style of chromosome IDs. "a" = "as-is"; "l" = "long style"␣
↪(eg. "chr1",
                        "chrX"); "s" = "short style" (eg. "1", "X").
  --unmap-file UNMAP_FILE
                        file to save unmapped entries. This will be ignored if [out_bed]␣
↪was not
                        provided.
```

Example 1

run `CrossMap bed` with **no** *output_file*:

```
$ CrossMap bed hg18ToHg19.over.chain.gz test.hg18.bed3

# Conversion results were printed to screen directly (column1-3 are hg18 based, column5-
↪7 are hg19 based)::
chr1    142614848       142617697       ->      chr1    143903503       143906352
chr1    142617697       142623312       ->      chr1    143906355       143911970
chr1    142623313       142623350       ->      chr1    143911971       143912008
```

Example 2

run `CrossMap bed` with *output_file* (test.hg19.bed3) specified:

```
$ CrossMap bed hg18ToHg19.over.chain.gz test.hg18.bed3 test.hg19.bed3

$ cat test.hg19.bed3
chr1    143903503       143906352
chr1    143906355       143911970
chr1    143911971       143912008
```

Example 3

One input region was split because it cannot map consecutively to the target assembly:

```
$ CrossMap bed hg18ToHg19.over.chain.gz test.hg18.bed3

chr10  81369946        81370453        +       ->      chr10   81380000        81380507 ␣
↪       +
chr10  81370483        81371363        +       ->      chr10   81380539        81381419 ␣
↪       +
chr10  81371363        81371365        +       ->      chr10   62961832        62961834 ␣
↪       +
chr10  81371412        81371432        +       (split.1:chr10:81371412:81371422:+)     ␣
↪chr10   62961775        62961785        +
chr10  81371412        81371432        +       (split.2:chr10:81371422:81371432:+)     ␣
↪chrX    63278348        63278358        +
```

Example 4

BedGraph format file can be converted using either `CrossMap bed` or `CrossMap wig`; however, the output formats are different:

- Use `CrossMap bed` command to convert a bedGraph file, the output is a **bedGraph** file.
- Use `CrossMap wig` command to convert a bedGraph file, the output is a **bigWig** file.

```
$ CrossMap bed hg19ToHg38.over.chain.gz 4_hg19.bgr

chrX    5873316 5873391 2.0     ->      chrX    5955275 5955350 2.0
chrX    5873673 5873710 0.8     ->      chrX    5955632 5955669 0.8
chrX    5873710 5873785 1.4     ->      chrX    5955669 5955744 1.4
chrX    5873896 5873929 0.9     ->      chrX    5955855 5955888 0.9
chrX    5873929 5874004 1.5     ->      chrX    5955888 5955963 1.5
chrX    5874230 5874471 0.3     ->      chrX    5956189 5956430 0.3
chrX    5874471 5874518 0.9     ->      chrX    5956430 5956477 0.9

$ python3 CrossMap wig hg19ToHg38.over.chain.gz 4_hg19.bgr output_hg38
@ 2018-11-06 00:09:11: Read chain_file:  hg19ToHg38.over.chain.gz
@ 2018-11-06 00:09:12: Liftover wiggle file: 4_hg19.bgr ==> output_hg38.bgr
@ 2018-11-06 00:09:12: Merging overlapped entries in bedGraph file ...
@ 2018-11-06 00:09:12: Sorting bedGraph file:output_hg38.bgr
@ 2018-11-06 00:09:12: Writing header to "output_hg38.bw" ...
@ 2018-11-06 00:09:12: Writing entries to "output_hg38.bw" ...
```

Example 5

Use `CrossMap region` command to convert large genomic regions (such as CNV blocks) in BED format.

```
# a genomic region of 3.48Mb
$ cat test.bed
chr2   239716679       243199373
```

If we use `CrossMap bed` command to convert this 3.48 Mb region. It will be split into 74 small blocks:

```
$CrossMap bed GRCh37_to_GRCh38.chain.gz  test.bed

chr2   239716679       243199373       (split.1:chr2:239716679:239801978:+)    chr2   ␣
↪238808038       238893337
chr2   239716679       243199373       (split.2:chr2:239831978:240205681:+)    chr2   ␣
↪238910282       239283985
chr2   239716679       243199373       (split.3:chr2:240205681:240319336:+)    chr2   ␣
↪239283986       239397641
... (split 74 times)
```

If we use `CrossMap region` command to convert this 3.48Mb region. Note: `-r` (the minimum ratio of bases that must remap) is 0.85 by default:

```
$CrossMap region GRCh37_to_GRCh38.chain.gz  test.bed

chr2   239716679       243199373       ->      chr2    238808038       242183529      ␣
↪map_ratio=0.9622
```

If we increase `-r` to 0.99, this region will fail:

```
$CrossMap region GRCh37_to_GRCh38.chain.gz  test.bed -r 0.99

chr2   239716679       243199373       Fail    map_ratio=0.9622
```

5.2 Convert BAM/CRAM/SAM format files

SAM (Sequence Alignment Map) format is a generic format for storing sequencing alignments, and BAM is the binary and compressed version of SAM (Li et al., 2009). CRAM was designed to be an efficient reference-based alternative to the SAM and BAM file formats. Most high-throughput sequencing (HTS) alignments were in SAM/BAM format and many HTS analysis tools work with SAM/BAM format. CrossMap updates chromosomes, genome coordinates, header sections, and all SAM flags accordingly. CrossMap's version number is inserted into the header section, along with the names of the original BAM file and the chain file. For pair-end sequencing, insert size is also recalculated. The input BAM file should be sorted and indexed properly using Samtools (Li et al., 2009). The output format is determined by the input format, and the BAM output will be sorted and indexed automatically.

Typing `CrossMap bam -h` will print help message:

```
$ CrossMap bam -h

usage: CrossMap bam [-h] [-m INSERT_SIZE] [-s INSERT_SIZE_STDEV] [-t INSERT_SIZE_FOLD] [-
↪a]
                    [--chromid {a,s,l}]
                    input.chain input.bam [out_bam]

positional arguments:
  input.chain           Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html)␣
```

(continues on next page)

(continued from previous page)

```
↪describes
                        pairwise alignments between two genomes. The input chain file␣
↪can be a plain
                        text file or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.
  input.bam             Input BAM file (https://genome.ucsc.edu/FAQ/FAQformat.html
↪#format5.1).
  out_bam               Output BAM file. if argument is missing, CrossMap will write BAM␣
↪file to the
                        STDOUT.

optional arguments:
  -h, --help            show this help message and exit
  -m INSERT_SIZE, --mean INSERT_SIZE
                        Average insert size of pair-end sequencing (bp).
  -s INSERT_SIZE_STDEV, --stdev INSERT_SIZE_STDEV
                        Stanadard deviation of insert size.
  -t INSERT_SIZE_FOLD, --times INSERT_SIZE_FOLD
                        A mapped pair is considered as "proper pair" if both ends mapped␣
↪to different
                        strand and the distance between them is less then '-t' * stdev␣
↪from the mean.
  -a, --append-tags     Add tag to each alignment in BAM file. Tags for pair-end␣
↪alignments include:
                        QF = QC failed, NN = both read1 and read2 unmapped, NU = read1␣
↪unmapped,
                        read2 unique mapped, NM = read1 unmapped, multiple mapped, UN =␣
↪read1
                        uniquely mapped, read2 unmap, UU = both read1 and read2 uniquely␣
↪mapped, UM =
                        read1 uniquely mapped, read2 multiple mapped, MN = read1␣
↪multiple mapped,
                        read2 unmapped, MU = read1 multiple mapped, read2 unique mapped,␣
↪MM = both
                        read1 and read2 multiple mapped. Tags for single-end alignments␣
↪include: QF =
                        QC failed, SN = unmaped, SM = multiple mapped, SU = uniquely␣
↪mapped.
  --chromid {a,s,l}     The style of chromosome IDs. "a" = "as-is"; "l" = "long style"␣
↪(eg. "chr1",
                        "chrX"); "s" = "short style" (eg. "1", "X").
```

Example

Convert BAM from hg19 to hg18:

```
# add optional tags using '-a' (recommend always use '-a' option)

$ CrossMap bam -a ../data/hg19ToHg18.over.chain.gz test.hg19.bam test.hg18
Insert size = 200.000000
Insert size stdev = 30.000000
Number of stdev from the mean = 3.000000
Add tags to each alignment = True
@ 2016-10-07 15:29:06: Read chain_file:  ../data/hg19ToHg18.over.chain.gz
```

(continues on next page)

(continued from previous page)

```
@ 2016-10-07 15:29:07: Liftover BAM file: test.hg19.bam ==> test.hg18.bam
@ 2016-10-07 15:29:14: Done!
@ 2016-10-07 15:29:14: Sort "test.hg18.bam" ...
@ 2016-10-07 15:29:15: Index "test.hg18.sorted.bam" ...
Total alignments:99914
       QC failed: 0
       R1 unique, R2 unique (UU): 96094
       R1 unique, R2 unmapp (UN): 3579
       R1 unique, R2 multiple (UM): 0
       R1 multiple, R2 multiple (MM): 0
       R1 multiple, R2 unique (MU): 233
       R1 multiple, R2 unmapped (MN): 8
       R1 unmap, R2 unmap (NN): 0
       R1 unmap, R2 unique (NU): 0
       R1 unmap, R2 multiple (NM): 0
```

BAM/SAM header sections was updated:

```
$ samtools view -H  test.hg19.bam
@SQ    SN:chr1 LN:249250621
@SQ    SN:chr2 LN:243199373
@SQ    SN:chr3 LN:198022430
...
@SQ    SN:chrX LN:155270560
@SQ    SN:chrY LN:59373566
@SQ    SN:chrM LN:16571
@RG    ID:Sample_618545BE      SM:Sample_618545BE      LB:Sample_618545BE      ␣
↪PL:Illumina
@PG    ID:bwa  PN:bwa  VN:0.6.2-r126

$ samtools view -H  test.hg18.bam
@HD    VN:1.0  SO:coordinate
@SQ    SN:chr1 LN:247249719
@SQ    SN:chr10        LN:135374737
@SQ    SN:chr11        LN:134452384
...
@SQ    SN:chrX LN:154913754
@SQ    SN:chrX_random  LN:1719168
@SQ    SN:chrY LN:57772954
@RG    ID:Sample_618545BE      SM:Sample_618545BE      LB:Sample_618545BE      ␣
↪PL:Illumina
@PG    PN:bwa  ID:bwa  VN:0.6.2-r126
@PG    ID:CrossMap     VN:0.5.0
@CO    Liftover from original BAM/SAM file: test.hg19.bam
@CO    Liftover is based on the chain file: ../test/hg19ToHg18.over.chain.gz
```

Optional tags:

Q
: QC. QC failed.

N
: Unmapped. Originally unmapped or originally mapped but failed to lift over to new assembly.

M
Multiple mapped. Alignment can be lifted over to multiple places.

U
Unique mapped. Alignment can be lifted over to only 1 place.

Tags for pair-end sequencing include:

- QF = QC failed
- NN = both read1 and read2 unmapped
- NU = read1 unmapped, read2 unique mapped
- NM = read1 unmapped, multiple mapped
- UN = read1 uniquely mapped, read2 unmap
- UU = both read1 and read2 uniquely mapped
- UM = read1 uniquely mapped, read2 multiple mapped
- MN = read1 multiple mapped, read2 unmapped
- MU = read1 multiple mapped, read2 unique mapped
- MM = both read1 and read2 multiple mapped

Tags for single-end sequencing include:

- QF = QC failed
- SN = unmaped
- SM = multiple mapped
- SU = uniquely mapped

Note:

1. All alignments (mapped, partial mapped, unmapped, QC failed) will write to one file. Users can filter them by tags.
2. The header section will be updated to the target assembly.
3. Genome coordinates and all SAM flags in the alignment section will be updated to the target assembly.
4. If the input is a CRAM file, pysam version should >= 0.8.2
5. Optional fields in the alignment section will not update.

5.3 Convert Wiggle format files

Wiggle (WIG) format is useful in displaying continuous data such as GC content and the reads intensity of high-throughput sequencing data. BigWig is a self-indexed binary-format Wiggle file and has the advantage of supporting random access. Input wiggle data can be in variableStep (for data with irregular intervals) or fixedStep (for data with regular intervals). Regardless of the input, the output files are always in bedGraph format.

Typing `CrossMap wig -h` will print help message:

```
$ CrossMap  wig -h

usage: CrossMap wig [-h] [--chromid {a,s,l}] input.chain input.wig out_wig

positional arguments:
  input.chain        Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html)␣
↪describes
                     pairwise alignments between two genomes. The input chain file can␣
↪be a plain
                     text file or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.
  input.wig          The input wiggle/bedGraph format file
                     (http://genome.ucsc.edu/goldenPath/help/wiggle.html). Both
↪"variableStep" and
                     "fixedStep" wiggle lines are supported. The input wiggle/bedGraph␣
↪file can be
                     plain text file, compressed file with extension of .gz, .Z, .z, .bz,
↪ .bz2 and
                     .bzip2, or even a URL pointing to accessible remote files (http://,␣
↪https:// and
                     ftp://). Compressed remote files are not supported.
  out_wig            Output bedGraph file. Regardless of the input is wiggle or bedGraph,
↪ the output
                     file is always in bedGraph format.

optional arguments:
  -h, --help         show this help message and exit
  --chromid {a,s,l}  The style of chromosome IDs. "a" = "as-is"; "l" = "long style" (eg.
↪"chr1",
                     "chrX"); "s" = "short style" (eg. "1", "X").
```

Note: To improve performance, this script calls GNU "sort" command internally. If the "sort" command is not callable, CrossMap will exit.

5.4 Convert BigWig format files

If an input file is in BigWig format, the output is BigWig format if UCSC's 'wigToBigWig' executable can be found; otherwise, the output file will be in bedGraph format.

After v0.3.0, UCSC's `wigToBigWig` command is no longer needed.

Typing `CrossMap bigwig -h` will print help message:

```
$ CrossMap bigwig -h

usage: CrossMap bigwig [-h] [--chromid {a,s,l}] input.chain input.bw output.bw

positional arguments:
  input.chain        Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html)␣
↪describes
                     pairwise alignments between two genomes. The input chain file can␣
```

(continues on next page)

(continued from previous page)

```
↪be a plain
                    text file or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.
  input.bw          The input bigWig format file
                    (https://genome.ucsc.edu/goldenPath/help/bigWig.html).
  output.bw         Output bigWig file.

optional arguments:
  -h, --help        show this help message and exit
  --chromid {a,s,l} The style of chromosome IDs. "a" = "as-is"; "l" = "long style" (eg.
↪"chr1",
                    "chrX"); "s" = "short style" (eg. "1", "X").
```

Example (Convert BigWig file from hg18 to hg19):

```
$ python CrossMap bigwig  hg19ToHg18.over.chain.gz  test.hg19.bw test.hg18
@ 2013-11-17 22:12:42: Read chain_file:  ../data/hg19ToHg18.over.chain.gz
@ 2013-11-17 22:12:44: Liftover bigwig file: test.hg19.bw ==> test.hg18.bgr
@ 2013-11-17 22:15:38: Merging overlapped entries in bedGraph file ...
@ 2013-11-17 22:15:38: Sorting bedGraph file:test.hg18.bgr
@ 2013-11-17 22:15:39: Convert wiggle to bigwig ...
```

Note: To improve performance, this script calls GNU "sort" command internally. If "sort" command does not exist, CrossMap will exit.

5.5 Convert GFF/GTF format files

GFF (General Feature Format) is another plain text file used to describe gene structure. GTF (Gene Transfer Format) is a refined version of GTF. The first eight fields are the same as GFF. Plain text, compressed plain text, and URLs pointing to remote files are all supported. Only chromosome and genome coordinates are updated. The format of the output is determined from the input.

Typing `CrossMap gff -h` will print help message:

```
$ CrossMap  gff -h

usage: CrossMap gff [-h] [--chromid {a,s,l}] input.chain input.gff [out_gff]

positional arguments:
  input.chain       Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html)␣
↪describes
                    pairwise alignments between two genomes. The input chain file can␣
↪be a plain
                    text file or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.
  input.gff         The input GFF (General Feature Format,
                    http://genome.ucsc.edu/FAQ/FAQformat.html#format3) or GTF (Gene␣
↪Transfer Format,
                    http://genome.ucsc.edu/FAQ/FAQformat.html#format4) file. The input␣
↪GFF/GTF file
                    can be plain text file, compressed file with extension of .gz, .Z, .
```

(continues on next page)

(continued from previous page)

```
↪z, .bz, .bz2
                      and .bzip2, or even a URL pointing to accessible remote files␣
↪(http://, https://
                      and ftp://). Compressed remote files are not supported.
  out_gff             Output GFF/GTF file. if argument is missing, CrossMap will write␣
↪GFF/GTF file to
                      the STDOUT.

optional arguments:
  -h, --help          show this help message and exit
  --chromid {a,s,l}   The style of chromosome IDs. "a" = "as-is"; "l" = "long style" (eg.
↪"chr1",
                      "chrX"); "s" = "short style" (eg. "1", "X").
```

Example (Convert GTF file from hg19 to hg18):

```
$ python CrossMap gff  hg19ToHg18.over.chain.gz test.hg19.gtf test.hg18.gtf
@ 2013-11-17 20:44:47: Read chain_file:  ../data/hg19ToHg18.over.chain.gz

$ head test.hg19.gtf
chr1   hg19_refGene    CDS     48267145        48267291        0.000000        -       0␣
↪       gene_id "NM_001194986"; transcript_id "NM_001194986";
chr1   hg19_refGene    exon    66081691        66081907        0.000000        +       .␣
↪       gene_id "NM_002303"; transcript_id "NM_002303";
chr1   hg19_refGene    CDS     145334684       145334792       0.000000        +       2␣
↪       gene_id "NM_001039703"; transcript_id "NM_001039703";
chr1   hg19_refGene    exon    172017752       172017890       0.000000        +       .␣
↪       gene_id "NM_001136127"; transcript_id "NM_001136127";
chr1   hg19_refGene    CDS     206589249       206589333       0.000000        +       2␣
↪       gene_id "NM_001170637"; transcript_id "NM_001170637";
chr1   hg19_refGene    exon    210573812       210574006       0.000000        +       .␣
↪       gene_id "NM_001170580"; transcript_id "NM_001170580";
chr1   hg19_refGene    CDS     235850249       235850347       0.000000        -       0␣
↪       gene_id "NM_000081"; transcript_id "NM_000081";
chr1   hg19_refGene    CDS     235880012       235880078       0.000000        -       1␣
↪       gene_id "NM_000081"; transcript_id "NM_000081";
chr1   hg19_refGene    exon    3417741 3417872 0.000000        -       .       gene_id
↪"NM_001409"; transcript_id "NM_001409";
chr1   hg19_refGene    exon    10190773        10190871        0.000000        +       .␣
↪       gene_id "NM_006048"; transcript_id "NM_006048";

$ head test.hg18.gtf
chr1   hg19_refGene    CDS     48039732        48039878        0.000000        -       0␣
↪       gene_id "NM_001194986"; transcript_id "NM_001194986";
chr1   hg19_refGene    exon    65854279        65854495        0.000000        +       .␣
↪       gene_id "NM_002303"; transcript_id "NM_002303";
chr1   hg19_refGene    CDS     144046041       144046149       0.000000        +       2␣
↪       gene_id "NM_001039703"; transcript_id "NM_001039703";
chr1   hg19_refGene    exon    170284375       170284513       0.000000        +       .␣
↪       gene_id "NM_001136127"; transcript_id "NM_001136127";
chr1   hg19_refGene    CDS     204655872       204655956       0.000000        +       2␣
↪       gene_id "NM_001170637"; transcript_id "NM_001170637";
```

(continues on next page)

(continued from previous page)

```
chr1   hg19_refGene    exon    208640435       208640629       0.000000        +       .␣
↪       gene_id "NM_001170580"; transcript_id "NM_001170580";
chr1   hg19_refGene    CDS     233916872       233916970       0.000000        -       0␣
↪       gene_id "NM_000081"; transcript_id "NM_000081";
chr1   hg19_refGene    CDS     233946635       233946701       0.000000        -       1␣
↪       gene_id "NM_000081"; transcript_id "NM_000081";
chr1   hg19_refGene    exon    3407601 3407732 0.000000        -       .       gene_id
↪"NM_001409"; transcript_id "NM_001409";
chr1   hg19_refGene    exon    10113360        10113458        0.000000        +       .␣
↪       gene_id "NM_006048"; transcript_id "NM_006048";
```

Note:

1. Each feature (exon, intron, UTR, etc) is processed separately and independently, and we do NOT check if features originally belonging to the same gene were converted into the same gene.
2. If a user wants to lift over gene annotation files, use BED12 format.
3. If no output file was specified, the output will be printed to screen (console). In this case, items that failed to convert are also printed out.

5.6 Convert VCF format files

VCF (variant call format) is a flexible and extendable line-oriented text format developed by the 1000 Genome Project. It is useful for representing single nucleotide variants, indels, copy number variants, and structural variants. Chromosomes, coordinates, and reference alleles are updated to a new assembly, and all the other fields are not changed.

Typing `CrossMap vcf -h` will print help message:

```
$ CrossMap vcf -h

usage: CrossMap vcf [-h] [--chromid {a,s,l}] [--no-comp-alleles] [--compress]
                      input.chain input.vcf refgenome.fa out_vcf

positional arguments:
  input.chain        Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html)␣
↪describes
                     pairwise alignments between two genomes. The input chain file can␣
↪be a plain
                     text file or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.
  input.vcf          Input VCF (variant call format, https://samtools.github.io/hts-
                     specs/VCFv4.2.pdf). The VCF file can be plain text file, compressed␣
↪file with
                     extension of .gz, .Z, .z, .bz, .bz2 and .bzip2, or even a URL␣
↪pointing to
                     accessible remote files (http://, https:// and ftp://). Compressed␣
↪remote files
                     are not supported.
  refgenome.fa       Chromosome sequences of target assembly in FASTA
                     (https://en.wikipedia.org/wiki/FASTA_format) format.
```

(continues on next page)

(continued from previous page)

```
  out_vcf            Output VCF file.

optional arguments:
  -h, --help         show this help message and exit
  --chromid {a,s,l}  The style of chromosome IDs. "a" = "as-is"; "l" = "long style" (eg.
↪"chr1",
                     "chrX"); "s" = "short style" (eg. "1", "X").
  --no-comp-alleles  If set, CrossMap does NOT check if the reference allele is␣
↪different from the
                     alternate allele.
  --compress         If set, compress the output VCF file by calling the system "gzip".
```

Example: filter out variants [reference_allele == alternative_allele]:

```
$ CrossMap vcf GRCh37_to_GRCh38.chain.gz test02_hg19.vcf hg38.fa out.hg38.vcf
@ 2020-12-08 22:33:16: Read the chain file:  ../data/human/GRCh37_to_GRCh38.chain.gz
@ 2020-12-08 22:33:17: Filter out variants [reference_allele == alternative_allele] ...
@ 2020-12-08 22:33:17: Updating contig field ...
@ 2020-12-08 22:33:17: Lifting over ...
@ 2020-12-08 22:33:17: Total entries: 882
@ 2020-12-08 22:33:17: Failed to map: 2
```

Example: Keep variants [reference_allele == alternative_allele]. Turn on `--no-comp-allele`:

```
$ CrossMap vcf GRCh37_to_GRCh38.chain.gz test02_hg19.vcf hg38.fa out.hg38.vcf --no-comp-
↪allele
@ 2020-12-08 22:36:51: Read the chain file:  ../data/human/GRCh37_to_GRCh38.chain.gz
@ 2020-12-08 22:36:51: Keep variants [reference_allele == alternative_allele] ...
@ 2020-12-08 22:36:51: Updating contig field ...
@ 2020-12-08 22:36:51: Lifting over ...
@ 2020-12-08 22:36:51: Total entries: 882
@ 2020-12-08 22:36:51: Failed to map: 1
```

Note:

1. Genome coordinates and reference alleles will be updated to target assembly.
2. The reference genome is the genome sequences of target assembly.
3. If the reference genome sequence file (../database/genome/hg18.fa) was not indexed, CrossMap will automatically index it (only the first time you run CrossMap).
4. Output files: *output_file* and *output_file.unmap*.
5. In the output VCF file, whether the chromosome IDs contain "chr" or not depends on the format of the input VCF file.

Interpretation of Failed tags:

- Fail(Multiple_hits) : This genomic location was mapped to two or more locations to the target assembly.
- Fail(REF==ALT) : After liftover, the reference allele is the same as the alternative allele (i.e. this is NOT an SNP/variant after liftover). In version 0.5.2, this checking can be turned off by setting '–no-comp-alleles'.
- Fail(Unmap) : Unable to map this location to the target assembly.

- Fail(KeyError) : Unable to find the contig ID (or chromosome ID) from the reference genome sequence (of the target assembly).

5.7 Convert MAF format files

MAF (mutation annotation format) files are tab-delimited files that contain somatic and/or germline mutation annotations. Please do not confuse with the Multiple Alignment Format.

Typing `CrossMap maf -h` will print help message:

```
$ CrossMap  maf -h

usage: CrossMap maf [-h] [--chromid {a,s,l}] input.chain input.maf refgenome.fa build_
↪name out_maf

positional arguments:
  input.chain        Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html)␣
↪describes
                     pairwise alignments between two genomes. The input chain file can␣
↪be a plain
                     text file or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.
  input.maf          Input MAF (https://docs.gdc.cancer.gov/Data/File_Formats/MAF_Format/
↪) format
                     file. The MAF file can be plain text file, compressed file with␣
↪extension of
                     .gz, .Z, .z, .bz, .bz2 and .bzip2, or even a URL pointing to␣
↪accessible remote
                     files (http://, https:// and ftp://). Compressed remote files are␣
↪not supported.
  refgenome.fa       Chromosome sequences of target assembly in FASTA
                     (https://en.wikipedia.org/wiki/FASTA_format) format.
  build_name         the name of the *target_assembly* (eg "GRCh38").
  out_maf            Output MAF file.

optional arguments:
  -h, --help         show this help message and exit
  --chromid {a,s,l}  The style of chromosome IDs. "a" = "as-is"; "l" = "long style" (eg.
↪"chr1",
                     "chrX"); "s" = "short style" (eg. "1", "X").
```

5.8 Convert GVCF format files

GVCF file format is described in here.

Typing `CrossMap gvcf -h` will print help message:

```
$ CrossMap  gvcf -h

usage: CrossMap gvcf [-h] [--chromid {a,s,l}] [--no-comp-alleles] [--compress]
                        input.chain input.gvcf refgenome.fa out_gvcf
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  input.chain        Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html)␣
↪describes
                     pairwise alignments between two genomes. The input chain file can␣
↪be a plain
                     text file or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.
  input.gvcf         Input gVCF (genomic variant call format, https://samtools.github.io/
↪hts-
                     specs/VCFv4.2.pdf). The gVCF file can be plain text file,␣
↪compressed file with
                     extension of .gz, .Z, .z, .bz, .bz2 and .bzip2, or even a URL␣
↪pointing to
                     accessible remote files (http://, https:// and ftp://). Compressed␣
↪remote files
                     are not supported.
  refgenome.fa       Chromosome sequences of target assembly in FASTA
                     (https://en.wikipedia.org/wiki/FASTA_format) format.
  out_gvcf           Output gVCF file.

optional arguments:
  -h, --help         show this help message and exit
  --chromid {a,s,l}  The style of chromosome IDs. "a" = "as-is"; "l" = "long style" (eg.
↪"chr1",
                     "chrX"); "s" = "short style" (eg. "1", "X").
  --no-comp-alleles  If set, CrossMap does NOT check if the reference allele is␣
↪different from the
                     alternate allele.
  --compress         If set, compress the output VCF file by calling the system "gzip".
```

Example (Convert GVCF file from hg19 to hg38):

```
$ CrossMap gvcf GRCh37_to_GRCh38.chain.gz test10_hg19.gvcf hg38.fa out.hg38.gvcf
@ 2020-12-08 22:19:44: Read the chain file:  ../data/human/GRCh37_to_GRCh38.chain.gz
@ 2020-12-08 22:19:44: Filter out variants [reference_allele == alternative_allele] ...
@ 2020-12-08 22:19:44: Updating contig field ...
@ 2020-12-08 22:19:44: Lifting over ...
@ 2020-12-08 22:19:44: Total variants: 10
@ 2020-12-08 22:19:44: Variants failed to map: 1
@ 2020-12-08 22:19:44: Total non-variant regions: 22
@ 2020-12-08 22:19:44: Non-variant regions failed to map: 0
```

5.9 Convert large genomic regions

For **large genomic regions** such as CNV blocks, the `CrossMap bed` will split each large region into smaller blocks that are 100% matched to the target assembly. `CrossMap region` will NOT split large regions, instead, it will calculate the **map ratio** (i.e. {bases mapped to target genome} / {total bases in query region}). If the **map ratio** is larger than the threshold specified by `-r`, the coordinates will be converted to the target genome, otherwise, it fails.

Typing `CrossMap region -h` will print help message:

```
usage: CrossMap region [-h] [--chromid {a,s,l}] [-r MIN_MAP_RATIO] input.chain input.bed␣
↪[out_bed]

positional arguments:
  input.chain           Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html)␣
↪describes
                        pairwise alignments between two genomes. The input chain file␣
↪can be a plain
                        text file or compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.
  input.bed             The input BED file. The first 3 columns must be “chrom”, “start”,
↪ and “end”.
                        The input BED file can be plain text file, compressed file with␣
↪extension of
                        .gz, .Z, .z, .bz, .bz2 and .bzip2, or even a URL pointing to␣
↪accessible
                        remote files (http://, https:// and ftp://). Compressed remote␣
↪files are not
                        supported.
  out_bed               Output BED file. if argument is missing, CrossMap will write BED␣
↪file to the
                        STDOUT.

optional arguments:
  -h, --help            show this help message and exit
  --chromid {a,s,l}     The style of chromosome IDs. "a" = "as-is"; "l" = "long style"␣
↪(eg. "chr1",
                        "chrX"); "s" = "short style" (eg. "1", "X").
  -r MIN_MAP_RATIO, --ratio MIN_MAP_RATIO
                        Minimum ratio of bases that must remap.
```

Example:

```
$CrossMap region  GRCh37_to_GRCh38.chain.gz test11_hg19_region.bed

@ 2020-08-14 16:46:04: Read the chain file:  ../data/human/GRCh37_to_GRCh38.chain.gz
chr1   0       2500000 ->      chr1    10000   2568561 map_ratio=0.9360
chr1   145394955       145807817       ->      chr1    145627235       146040039       ␣
↪map_ratio=0.9994
chr1   146527987       147394444       ->      chr1    147056425       147922330       ␣
↪map_ratio=0.9989
chr10  82045472        88931651        ->      chr10   80285716        87171894        ␣
↪map_ratio=1.0000
chr11  43940000        46020000        ->      chr11   43918450        45998449        ␣
↪map_ratio=1.0000
chr15  22805313        23094530        Fail    map_ratio=0.3607
chr15  22805313        28390339        ->      chr15   22598414        28145193        ␣
↪map_ratio=0.8967
chr15  31080645        32462776        ->      chr15   30788442        32170575        ␣
↪map_ratio=1.0000
chr15  72900171        78151253        ->      chr15   72607830        77858911        ␣
↪map_ratio=1.0000
chr15  83219735        85722039        ->      chr15   82550985        85178808        ␣
↪map_ratio=0.9800
```

(continues on next page)

(continued from previous page)

```
chr16  15511655        16293689        ->      chr16   15417798        16199832        ␣
↪map_ratio=1.0000
chr16  21950135        22431889        ->      chr16   21938814        22420568        ␣
↪map_ratio=1.0000
chr16  28823196        29046783        ->      chr16   28811875        29035462        ␣
↪map_ratio=1.0000
chr16  29650840        30200773        ->      chr16   29639519        30189452        ␣
↪map_ratio=1.0000
chr17  1247834 1303556 ->      chr17   1344540 1400262 map_ratio=1.0000
chr17  2496923 2588909 ->      chr17   2593629 2685615 map_ratio=1.0000
chr17  16812771        20211017        ->      chr17   16909457        20307704        ␣
↪map_ratio=1.0000
chr17  29107491        30265075        ->      chr17   30780473        31938056        ␣
↪map_ratio=1.0000
chr17  34815904        36217432        Unmap
chr17  43705356        44164691        ->      chr17   45627990        46087325        ␣
↪map_ratio=1.0000
chr2   50145643        51259674        ->      chr2    49918505        51032536        ␣
↪map_ratio=1.0000
chr2   96742409        97677516        ->      chr2    96076661        97011779        ␣
↪map_ratio=1.0000
chr2   111394040       112012649       ->      chr2    110636463       111255072       ␣
↪map_ratio=1.0000
chr2   239716679       243199373       ->      chr2    238808038       242183529       ␣
↪map_ratio=0.9622
chr22  19037332        21466726        Fail    map_ratio=0.8490
chr22  21920127        23653646        ->      chr22   21565838        23311459        ␣
↪map_ratio=0.9996
chr22  51113070        51171640        ->      chr22   50674642        50733212        ␣
↪map_ratio=1.0000
chr3   195720167       197354826       ->      chr3    195993296       197627955       ␣
↪map_ratio=1.0000
chr4   1552030 2091303 ->      chr4    1550303 2089576 map_ratio=1.0000
chr5   175720924       177052594       ->      chr5    176293921       177625593       ␣
↪map_ratio=1.0000
chr7   72744915        74142892        ->      chr7    73330912        74728554        ␣
↪map_ratio=0.9997
chr8   8098990 11872558        ->      chr8    8241468 12015049        map_ratio=1.0000
chr9   140513444       140730578       ->      chr9    137618992       137836126       ␣
↪map_ratio=1.0000
```

Note:

1. Input BED file should have at least 3 columns (chrom, start, end). Additional columns will be kept as is.

5.10 View chain file

Typing `CrossMap viewchain -h` will print help message:

```
usage: CrossMap viewchain [-h] input.chain

positional arguments:
  input.chain  Chain file (https://genome.ucsc.edu/goldenPath/help/chain.html) describes␣
↪pairwise
               alignments between two genomes. The input chain file can be a plain text␣
↪file or
               compressed (.gz, .Z, .z, .bz, .bz2, .bzip2) file.

optional arguments:
  -h, --help   show this help message and exit
```

Example:

```
$CrossMap viewchain ../data/human/GRCh37_to_GRCh38.chain.gz >chain.tab
$head chain.tab
1  10000 177417   +  1  10000 177417   +
1  227417   267719   +  1  257666   297968   +
1  317719   471368   +  1  347968   501617   -
1  521368   1566075  +  1  585988   1630695  +
1  1566075  1569784  +  1  1630696  1634405  +
1  1569784  1570918  +  1  1634408  1635542  +
1  1570918  1570922  +  1  1635546  1635550  +
1  1570922  1574299  +  1  1635560  1638937  +
1  1574299  1583669  +  1  1638938  1648308  +
1  1583669  1583878  +  1  1648309  1648518  +
```

CHAPTER

SIX

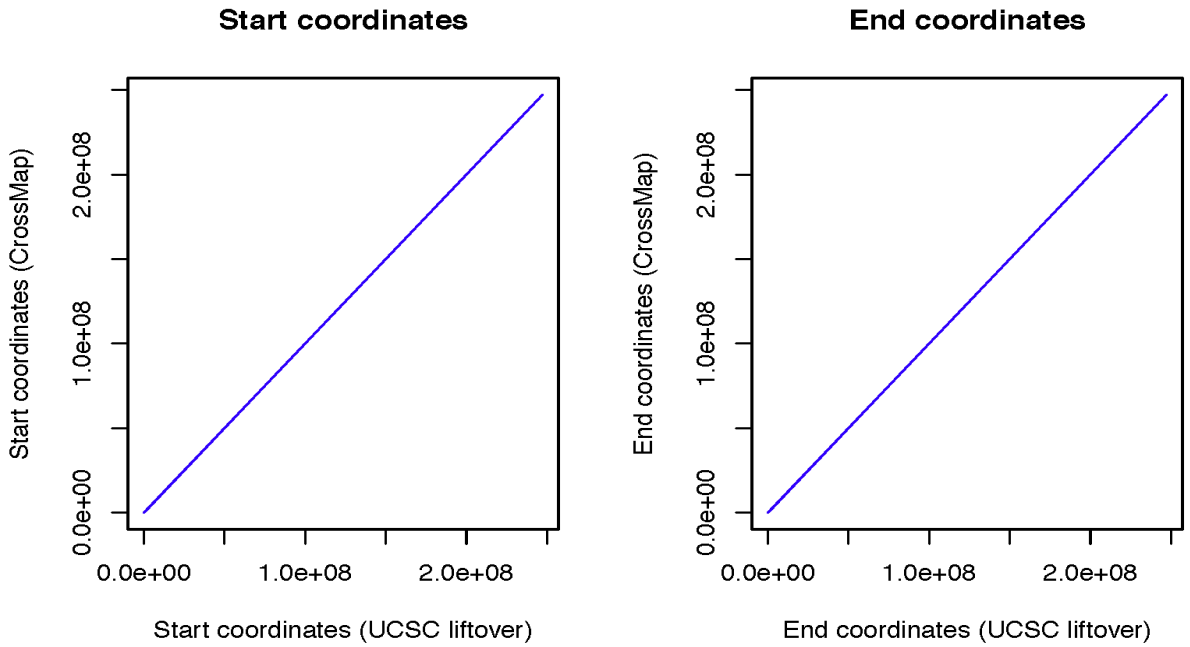
COMPARE TO UCSC LIFTOVER TOOL

To assess the accuracy of CrossMap, we randomly generated 10,000 genome intervals (download from here) with the fixed interval size of 200 bp from hg19. Then we converted them into hg18 using CrossMap and UCSC liftover tool with default configurations. We compare CrossMap to UCSC liftover tool because it is the most widely used tool to convert genome coordinates.

CrossMap failed to convert 613 intervals, and the UCSC liftover tool failed to convert 614 intervals. All failed intervals are exactly the same except for one region (chr2 90542908 90543108). UCSC failed to convert it because this region needs to be split twice:

Original (hg19)	Split (hg19)	Target (hg18)
chr2 90542908 90543108 -	chr2 90542908 90542933 -	chr2 89906445 89906470 -
chr2 90542908 90543108 -	chr2 90542933 90543001 -	chr2 87414583 87414651 -
chr2 90542908 90543108 -	chr2 90543010 90543108 -	chr2 87414276 87414374 -

For genome intervals that were successfully converted to hg18, the start and end coordinates are exactly the same between UCSC conversion and CrossMap conversion.



CHAPTER

SEVEN

CITATION

Zhao, H., Sun, Z., Wang, J., Huang, H., Kocher, J.-P., & Wang, L. (2013). CrossMap: a versatile tool for coordinate conversion between genome assemblies. Bioinformatics (Oxford, England), btt730

CHAPTER

EIGHT

LICENSE

CrossMap is distributed under GNU General Public License

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

CHAPTER

NINE

CONTACT

- Wang.Liguo AT mayo.edu