capC-MAP Documentation

Release 1.1.3

Chris Brackley

May 27, 2019

Contents

CHAPTER 1

Introduction

capC-MAP is a software package for the analysis of sequencing data from Capture-C experiments [Hughes2014] [Davies2016]. It is actually a suit of programs written in C++ and Python, along with a Python wrapper script which allows a full analysis pipeline to be run using a single command on Unix-based systems. capC-MAP was written so as to be as easy to use as possible, but allow maximum customisation and separate use of the component programs by advanced users. While other other tools which can analyse Capture-C data are available, these are mostly extensions to software designed with HiC data in mind. To our knowledge capC-MAP is the first software dedicated to, and optimised for Capture-C data; it is designed to be used by beginners and 'C-method' experts alike.

The aim of a Capture-C experiment is to obtain an interaction profile for a set of "target" genomic loci. This is similar to the aim of a 4C experiment, but the method allows multiple targets to be probes in a single experiment. This is achieved using oligo capture technology. A frequently cutting restriction enzyme is used to fragment the DNA so as to obtain interactions at high-resolution. Oligos are designed against a set of restriction enzyme fragments of interest. Throughout this manual we use the term "target" to refer to restriction enzyme fragments for which oligos have been designed, and "reporter" for fragments which are found ligated to target fragments. The term "target" is synonymous with the "viewpoint" or "bait" fragment in 4C.

CHAPTER 2

Installation

2.1 Requirements

capC-MAP requires a C++ compiler and Python; also the following software should be installed and visible on the system path:

- cutadapt (>=1.11)
- bowtie (>=1.1.1 - note that capC-MAP is not compatible with bowtie2)
- samtools (>=1.3.1)

and the following python packages should be installed:

- BioPython

Version numbers are those which have been tested, and other versions may also work.

2.2 Installation

There are several ways to install capC-MAP. One of the easiest is if the bioconda/conda packaging system is available on your system (see https://bioconda.github.io/#install-bioconda for details on how to get it). In this case capC-MAP and all of its requirements can be installed with the single command.

```
conda install capc-map
```

If you do not have conda on your system, you will need to install the requirements listed above separately. Then install capC-MAP perform the following steps:

1. Download the software and unzip into a directory in your home directory. Or clone from git using the command:

   ```
   git clone https://git.ecdf.ed.ac.uk/cbrackle/capC-MAP.git
   ```

2. capC-MAP consists of a set of programs written in C++ and a Python package, which both need to be installed on your system. If you have root privileges on your system you can compile and install the C++ programs by running the following commands in the capC-MAP root directory

```
./configure
make
make install
```

and you can install the Python package using pip with the command

```
pip install .
```

when in the capC-MAP package directory.

Alternatively, if you do not have root privileges, you can install a local copy of capC-MAP in your home directory using, for example, the commands

```
./configure --prefix=${HOME}/.local/
make
make install
```

and

```
pip install --user .
```

If you do not have pip available on your system you can instead install the python package using the command

```
python setup.py install
```

or without root privileges

```
python setup.py install --user
```

3. If you have installed capC-MAP in your home directory, you will need to ensure the binaries directory is present on the system PATH. For example, by adding the following line to your `~/.bashrc` file

```
export PATH=$PATH:~/.local/bin
```

CHAPTER 3

Quick start

A typical pipeline for analysis of capture c data has three main steps:

3.1 Build genome index and restriction enzyme fragment list.

Since capC-MAP uses bowtie [Langmead2009] for sequence alignment, a bowtie index for the reference genome must be built. This requires a single fasta file containing the reference genome. For example

```
bowtie-build mygenome.fasta mygeneome
```

The same fasta file must then be used to generate a list of restriction enzyme fragments for the genome. This is done using the `genomedigest` function in capC-MAP.

```
capC-MAP genomedigest -f mygenome.fasta -r DpnII \
                      -o mygenome_dpnII_fragments.bed
```

where the options are as follows:

`-f mygenome.fasta`	specifies the fasta for the reference genome
`-r DpnII`	specifies the restriction enzyme used in the experiment
`-o mygenome_dpnII_f ragments.bed`	specifies the output bed file

Note that the \ character means that a single command is broken across lines.

Pre-built bowtie indexes for many genomes are available for download on the bowtie website (http://bowtie-bio.sourceforge.net), though a fasta file is required to build the list of restriction enzyme fragments. It is essential to ensure that the index and fragments list are built from the same reference genome (for this reason we recommend building you own index).

3.2 Perform quality control on fastq data files.

We recommend performing standard quality control on the fastq files, for example using the FastQC software. Since the core aim in capture c data it to enrich the library for specific 'target' fragments, this may get flagged up the the FastQC report. Also, since the capture c protocol recommends fragments are sonicated to have an average length of 200-300bp, depending on the sequencing read length, it could be that there is a significant proportion of read-through into the adapters – again this may be flagged up by FastQC. By default the capC-MAP pipeline includes an adapter trimming step.

3.3 Run capC-MAP analysis pipeline.

The main capC-MAP pipeline is then run using the command

```
capC-MAP pipeline -c config_file.txt \
                  -o mycaptureCexperiment
```

where the options are as follows:

`-c config_file.txt`	specifies the experiment configuration file
`-o mycaptureCexperiment`	specifies a directory where all output will be saved

All capC-MAP options are specified in the configuration text file. We recommend using the example configuration file provided with capC-MAP as a template. The output directory must not exist. Required inputs (specified in the configuration file) are:

- a pair of fastq files;
- a bed file containing a list of target restriction enzyme fragments;
- the bowtie index for the reference genome; and
- a bed file containing a genome wide list of restriction enzyme fragments for the reference genome.

The following output files will be generated in the output directory:

- an *in silico* digested version of the input sequencing data in a single fastq file.
- a SAM file containing mapped reads; the same data is also given in the compressed BAM format.
- a capC-MAP report file giving details of mapped fragments.
- for each target specified in the targets bed file, a 'validpairs' file containing a list of all valid intrachromosomal interactions, and a 'validinterchrom' files containing a list of valid interchromosomal interactions.
- for each target specified in the targets bed file, and depending on options specified in the configuration file, a bedGraph showing binned, smoothed, and normalized to reads per million genome wide interactions.

Once the analysis is complete, some of these outputs are not needed for most down-stream analyses. For example the *in silico* digested fastq file, SAM files and the raw pairs files can usually be deleted to save disk space.

In order to generate further bedGraphs with different binning or normalization options without re-running the full analysis, capC-MAP can be run in `postprocess` mode (see section *capC-MAP postprocess*).

CHAPTER 4

Using capC-MAP

The capC-MAP software is essentially a collection of separate tools along with a "wrapper script", and can be used in two ways. Either, the full Capture-C analysis can be performed by using the single command `capC-MAP run`, or for advanced use individual steps of the pipeline can be run separately.

The `capC-MAP` command can be run in one of five mode:

- `genomedigest` is used to generate a list of restriction enzyme fragments covering the reference genome;
- `run` is used to run the full analysis pipeline;
- `postprocess` is used to generate binned contact profiles;
- `combinereps` is used to combine processed data sets from replicates; and
- `getchromsizes` is used to generate a list of chromosome sizes from a list or restriction enzyme fragments.

The `run`, `postprocess` and `combinereps` modes read from a "configuration file"; the other modes take options on the command line.

4.1 capC-MAP `genomedigest`

In order to identify interactions between restriction enzyme fragments, capC-MAP needs a genome-wide map of where the restriction sites are. Using the `genomedigest` mode capC-MAP can generate this list from a fasta file for the reference genome. Once generated this can be re-used for any Capture-C experiments using that reference genome.

capC-MAP stores the list of fragments as a standard bed file; this can be generated with the following command:

```
capC-MAP genomedigest -f <reference genome fasta file> \
                      -r <name of restriction enzyme> \
                      -o <output bed file>
```

where all three options are required. The `-r` option specifies the name of a restriction enzyme which is known to capC-MAP (e.g. DpnII which is typically used in Capture-C experiments). New restriction enzymes can be added to capC-MAP by editing the "restriction_enzymes.txt" file which is part of the install (run the genomedigest command with the name of a restriction enzyme - if it is not known, capC-MAP will give the location of "restriction_enzymes.txt"). To

add a new enzyme the recognition sequence and also the "cut point" must be specified; the "cut point" does not have to coincide with the true enzyme cutting position, it is just used internally by capC-MAP and will not affect results. (See also section *Restriction Enzymes* below.)

4.2 capC-MAP `run`

The capC-MAP software performs each of the following processing steps using a single command line:

- trim adapters from paired-end read fastq files using the **cutadapt** software [Martin2011]
- perform *in silico* restriction enzyme digestion,
- align restriction enzyme fragments to reference genome using **bowtie** [Langmead2009] in single-end read mode,
- sort the resulting SAM file by read name using **samtools** [Li2009],
- identify "read groups" of mapped restriction enzyme fragments and remove duplicates,
- identify "target" and "reporter" fragments in each group,
- remove interactions between targets, interactions within "exclusion zones", and groups with multiple non-adjacent reporters,
- remove and count interchromosomal interactions,
- generate pile-ups of interactions for each target,
- generate normalized, binned and smoothed interaction profiles for each target.

Full details of each of these steps are given in the associated paper [capC-MAP]. This pipeline is based on that detailed in the original Capture-C publications from the Hughes Lab [Hughes2014] (and see also [Davies2016]).

The command line takes the form:

```
capC-MAP run -c <configuration file> \
             -o <output directory>
```

where all options and input files are specified in the configuration file, and capC-MAP will create a new directory at the specified location for all output files. capC-MAP will not overwrite existing files, and will instead fail with an error.

4.2.1 The configuration file

The configuration file is a text file which sets all of the options required for the capC-MAP pipeline. We recommend taking the example configuration file provided with the capC-MAP installation as a template. The file is structured such that each line represents an option specified by the first word on that line. Any line beginning with a hash '#' character is treated as a comment and ignored by capC-MAP. Some options are required to be in the configuration file, whereas others will take a default value if not specified.

The available options are as follows

`FASTQ1 <file>` *Required.* Specifies the relative path to the first of the pair of fastq files. Takes exactly one argument; subsequent arguments are ignored.

`FASTQ2 <file>` *Required.* Specifies the relative path to the second of the pair of fastq files. Takes exactly one argument; subsequent arguments are ignored.

`TARGETS <file>` *Required.* Specifies a bed file containing a list of "target" restriction enzyme fragments. Each target fragment must match as fragment given in the `RESTFRAGS` file. Details are given in section :ref:'sectiontargets' below. Takes exactly one argument; subsequent arguments are ignored.

INDEX <index> *Required*. Specifies the bowtie index for the reference genome; this will be passed to bowtie for the alignment step of the pipeline. The relative path must be given, with the name as specified when the reference was built using `bowtie-build`. Takes exactly one argument; subsequent arguments are ignored. Pre-built bowtie indexes are available to download on the bowtie website, though it is essential to ensure that the index is built from the same reference genome as the restriction enzyme fragments (for this reason we recommend building the index from the reference).

RESTFRAGS <file> *Required*. Specifies a bed file for the list of restriction enzyme fragments covering the reference genome, as generated using capC-MAP with the `genomedigest` mode. Note that this must have been generated for the same restriction enzyme as is specified by the `ENZYME` option, and must be generated from the same reference genome as the bowtie index specified by the `INDEX` option. Takes exactly one argument; subsequent arguments are ignored.

ENZYME <enzyme name|cut sequence> *Required*. Specifies the restriction enzyme used in the experiment, either by name or by specifying the recognition sequence directly. As detailed in section *capC-MAP genomedigest* above, this is typically DpnII in a Capture-C experiment, but alternatively the recognition sequence can be specified directly. See section *Restriction Enzymes* below for further details. Takes exactly one argument, case insensitive; subsequent arguments are ignored.

TRIMADAPTERS [TRUE|FALSE] *Optional*. Default: TRUE. Since the Capture-C protocol recommends sonicating the library to give short fragments, it is expected that for many reads sequencing will have gone through into the adapter sequence. By default capC-MAP uses the cutadapt software to trim adapters from the input fastq files. This step can be skipped by setting this option to FALSE. Takes exactly one argument; subsequent arguments are ignored.

PARALLEL <N> *Optional*. Default: 1. To speed up processing, some step of the capC-MAP pipeline can be run on multiple processors. Specifically sequence alignment using bowtie and sorting and file conversion using samtools can be run in parallel. This option specifies the number of processors, and this is passed to bowtie and samtools. Since these are the slowest steps in the pipeline, no other steps are run on multiple processors. Takes exactly one integer argument; subsequent arguments are ignored.

ALIGNMODE [CONSERVATIVE|RELAXED|CUSTOM] *Optional*. Default: CONSERVATIVE. Determines the alignment options which are passed to bowtie. See section *Alignment mode* below for details. The two preset options CONSERVATIVE and RELAXED require no further options. If CUSTOM is specified, everything following it on the same line is taken to be an option for the aligner, and is passed verbatim to bowtie (bowtie's '-p' option, the index and input/output file names should not be included as capC-MAP will add these).

EXCLUDE <N> *Optional.* Default: 1000. Sets the distance in base-pairs, where if a reporter fragment is closer to a target than this it is discarded.

INTERCHROM *Optional.* Default: FALSE. Sets whether capC-MAP generates pile-ups for interchromosomal interactions. Note that valid interchromosomal interaction pairs are always saved in 'validinterchrom' output files.

BIN <S> <W> *Optional.* Tells capC-MAP to generate binned interaction profiles as well as restriction enzyme fragment level pile-ups. Since restriction enzyme fragments have an irregular size, some binning is recommended. The step size for bins in base-pairs is set by the integer `<S>`. Smoothing can also be applied via a sliding window of width `<W>` base-pairs, i.e. each bin gives the number of interactions from within a window of that width. To bin without smoothing set W=S. If the `NORMALIZE` option is also set TRUE, the binned profiles will be normalized.

NORMALIZE *Optional.* Default: FALSE. Sets whether binned interaction profiles are also normalized to be in units of "reads-per-million".

COMBINEMODE [TRUE|FALSE] *Optional*. Default: FALSE. Specifies whether the analysis will be run with 'combine mode' activated. This will combine interactions from selected targets into a single interaction profile. See section *Combine mode* below for details. Takes exactly one argument; subsequent arguments are ignored.

COMBINECOUNT <N> *Optional*. Default: 2. Only relevant when combine mode is active. Specifies how many targets are to be combined. See section *Combine mode* below for details. Takes exactly one integer argument;

subsequent arguments are ignored.

DRYRUN [TRUE|FALSE] *Optional*. Default: FALSE. If set TRUE capC-MAP will be run in "dry run" mode, which steps through each stage of the pipe-line without actually running it. This is useful for testing all required files etc. are present, and generating the 'capC-MAP.commands.log' file, which lists all pipe-line steps as bash command lines (see section *Outputs* below).

4.2.2 Alignment mode

Alignment of fragments to the reference genome is done using the bowtie software. By default capC-MAP uses a rather conservative set of parameters for the alignment, as recommended in the original Capture-C protocol [Hughes2014]: only fragments which map to a single genomic location are reported. Another, more relaxed, pre-set for the alignment parameters where the best alignment for multi-mapping reads is reported is also available. Alternatively users can specify their own custom set of parameters to be passed to bowtie. Note that since the number of processors which bowtie uses is specified separately, this should not be included in the custom alignment mode line of the configuration file.

4.2.3 The targets file

A bed file containing a list of all targeted restriction enzyme fragments is a required input, and is specified with the `TARGETS` option in the configuration file. Each line must contain four fields separated by tabs: chromosome, start, end, and target name; for example

```
chr2    12345    67890    firsttargetname
chr4    23456    78901    secondtargetname
```

Target names must be unique, and the same fragment cannot appear more than once. Each target fragment must be also present in the restriction enzymes file specified with the `RESTFRAGS` option in the configuration file (i.e. there must be a line with the same chrom:start:end fields). capC-MAP provides a utility `location2fragment` which is useful for generating a valid targets file, e.g. from a bed file containing a list of oligo regions - see section *capClocation2fragment*.

4.2.4 Restriction Enzymes

capC-MAP needs to know the restriction enzyme used to digest the genome. For Capture-C experiments this is typically DpnII. To use a different restriction enzyme, the DNA recognition sequence can be specified with the `ENZYME` option. Note that for recognition sequences with an overhang, such as HindIII, there is a nucleotide fill-in step during ligation - this means that the recognition sequence used to generate the restriction enzyme map with the capC-MAP `genomedigest` command may be different to the one used for the rest of the analysis.

4.2.5 Combine mode

Sometimes a genome feature of interest might appear at multiple locations in the genome. For example, in the paper in which the Capture-C method was originally described, the authors studied interactions with the promoter of the mouse α-globin gene. There are two copies of α-globin the mouse genome, with largely the same sequence. While oligos designed to target those promoter will lead to enrichment of fragments containing either copy, these will be associated to only a single genomic location when aligned to the reference genome. Thus interactions for the two targets should be combined into a single interaction profile. This is handled automatically when capC-MAP is run in "combine mode", provided that the targets to be combined are named in a specific way. Names of targets which start with the same string, and end with "`_C1`", "`_C2`", "`_C3`" . . . etc. For example, an experiment targeting the two copies of the mouse α-globin gene (mm9) might use the following targets:

```
chr11    32182970    32183819    AGLOB_C1
chr11    32195805    32196636    AGLOB_C2
```

When run with combine mode set TRUE, capC-MAP will generate a set of output files with "AGLOB_combined" as the target name, as well as output for "AGLOB_C1" and "AGLOB_C2" individually. When capC-MAP is run with "conservative" alignment mode (recommended), options are passed to bowtie which specify that only reads which map uniquely to a single location are reported. When combine mode is used, target fragments are likely to map to multiple locations, so we must relax this restriction. This is done with the `COMBINECOUNT` option: if two targets are to be combined, this should be set to 2; if three targets are to be combined, this should be set to 3, etc. By default `COMBINECOUNT` will be set to 2 when combine mode is active.

4.2.6 Outputs

capC-MAP generates the following files in the output directory:

capC-MAP.commands.log A log file showing a list of command lines for each step of the analysis. This is also generated in DRYRUN mode.

captured_report.dat A report file from the main processing stage of the pipe-line. Shows counts of various points where reads were discarded, useful for evaluating the quality of the data.

captured_interactioncounts.dat Contains counts for each target of the number of valid interactions, and how many were intra/inter chromosomal.

srt_aligned.bam BAM file for the aligned read fragments sorted by name

captured_validpairs_*targetname*.pairs A set of files containing a list of all valid intrachromosomal interactions, one file for each target. Restriction enzyme fragment coordinates are given in bed file format.

captured_validinterchom_*targetname*.pairs Similar files showing interchromosomal interactions.

captured_rawpileup_*targetname*.bdg Set of bedGraph files, one for each target, giving the "piled-up" intrachromosomal interactions. Each entry refers to a single restriction enzyme fragment, so these have irregular widths. Units are numbers of reads.

captured_normalizedpileup_*targetname*.bdg When the `NORMALIZE` parameter is set TRUE, capC-MAP also generates a set of bedGraph files where the piled-up intrachromosomal interaction counts have been normalized to reads-per-million, i.e. the number of reads for each target genome wide will sum to one million.

captured_rawpileup_interchom_*targetname*.bdg Set of bedGraph files, giving the "piled-up" interchromosomal interactions. Only present if option `INTERCHROM` was set TRUE in the configuration file.

captured_normalizedpileup_interchom_*targetname*.bdg When the `NORMALIZE` and `INTERCHROM` options are both set TRUE, capC-MAP also generates a set of bedGraph files where the piled-up interchromosomal interaction counts have been normalized to reads-per-million.

captured_bin_*S*_*W*_*targetname*.bdg Here *S* ans *W* are integers. Set of bedGraph files containing the intrachromosomal interaction profile which has been binned using a step size *S* and a window size *W*, one file for each target. Units are numbers of reads.

captured_bin_*S*_*W*_RPM_*targetname*.bdg As above, but units are in reads-per-million (RPM). These are generated instead of the above if option `NORMALIZE` was set TRUE in the configuration file.

Additionally log files and error files are generated from each step of the pipe-line, and these contain any output from the programs used in each step - this is useful for troubleshooting if capC-MAP fails with an error.

4.3 capC-MAP `postprocess`

The postprocess mode is used to generate additional interaction profiles from a data set which has already been analysed using the `capC-MAP run` command. By adding new `BIN` or `NORMALIZE` lines to the configuration file, this mode can be used to generate new captured_bin_*S_W_targetname*.bdg or captured_bin_*S_W*_RPM_*targetname*.bdg files from the captured_rawpileup_*targetname*.bdg or captured_normalizedpileup_*targetname*.bdg files.

The command line takes the form

```
capC-MAP postprocess -c <configuration file> \
                     -o <output directory>
```

where the configuration file and output directory are the same ones used in the original `capC-MAP run` command.

If the pile-up files are not present in the output directory, capC-MAP will try to generate them from the captured_validpairs_*targetname*.pairs files. By default capC-MAP will not overwrite any existing files.

4.4 capC-MAP `combinereps`

capC-MAP provides a facility for combining replicate data sets. Each set must first be analysed independently using the `capC-MAP run` command; then the `capC-MAP combinereps` command is used to combine the data into a single set of files for each target, and generate binned, smoothed and normalized interaction profiles. A typical set of commands might take the form

```
capC-MAP run -c config_rep1.txt -o output_rep1
capC-MAP run -c config_rep2.txt -o output_rep2
capC-MAP combinereps -c config_rep1.txt  \
                     -i output_rep1 \
                     -i output_rep2 \
                     -o output_combinedreps_1_2
```

where the first two commands run the analysis on each of the replicates, and the third combines that data. The two replicate data sets must be generated using the same targets file, genome index and restriction enzyme, as specified in the configuration file. Multiple instances of the `-i` option are used to select the directories containing the capC-MAP output for each individual replicate. Once the combined results directory has been generated, new Capture-C profile files with different binning smoothing and normalization options can be generated using the `capC-MAP postprocess` command as detailed above.

4.5 capC-MAP `getchromsizes`

It is often useful to have a list of the chromosome sizes for a reference genome, and capC-MAP provides a tool to generate this from a restriction enzyme map, as generated using the `capC-MAP genomedigest` command. The command line takes the form

```
capC-MAP getchromsizes -f <fragments file> \
                       [-o <output file>]
```

where the fragments file must be in the format generated by `capC-MAP genomedigest`, and if the optional output file is not specified, the file name `chrom.sizes` will be used.

CHAPTER 5

Advanced usage

capC-MAP is actually a suit of programs written in C++ along with a Python "front end" which allows a whole processing pipeline to be run via a single command line. For advanced usage each of the component programs can be run independently, and these are documented here. As well as the four core capC-MAP programs, there is a further additional tool `capClocation2fragment`.

5.1 capCdigestfastq

The program `capCdigestfastq` performs a *in silico* restriction enzyme digestion of a pair of fastq sequence files. The program can run in two modes. In the standard mode, as used in the capC-MAP pipeline, the program reads from the two paired end read fastq files line-by-line, checking that the names match for each of the pair. The program splits the read pair into smaller restriction enzyme fragments at the specified cutting sequence. As well as the cut sequence, a cut position within that sequence has to be specified; this does not have to match the real cut position of the enzyme and will not affect downstream capC-MAP results. All fragments are output to a single fastq file, with read pair names given in a format suitable for use the the capC-MAP program `capCmain`, once the fastq has been mapped to the reference genome.

The program can also run in an alternative "long" mode, where only the longest restriction enzyme fragment from each of the pair is retained, and output is given in two separate fastq files. This output is not suitable for use with the `capCmain` program.

5.2 capCmain

The `capCmain` is the main work-horse program of capC-MAP, and takes as an input a name-sorted SAM file generated using bowtie to map a fastq file which was generated by the `digestfastq` program. It also requires a map of restriction enzyme fragments for the reference genome (as generated by `capC-MAP genomedigest`), and bed file containing a list of target restriction enzyme fragments. The output is a list of intrachromosomal interactions and a list of interchromosomal interactions for each target.

5.3 capCpair2bg

The `capCpair2bg` program reads in a single bed file list of intrachromosomal interactions (as output by `capCmain`), and generates a "pile-up" of interaction counts at each restriction enzyme fragment in bedGraph format - i.e. an interaction profile.

5.4 capCpileup2binned

The `capCpileup2binned` program reads in a restriction enzyme fragment level intrachromosomal interaction profile (as generated by `capCpair2bg`) and generates a binned, smoothed, and normalized interaction profile. Optional parameters are a binning step size `S` and windows size `W` (as detailed in section *capC-MAP run*), and a total number of reads `T` for normalization to reads-per-million. If only `S` and `W` are specified the profile is not normalized; if only `T` is provided, a normalized profile at restriction enzyme fragment resolution is generated.

5.5 capClocation2fragment

The `capClocation2fragment` program reads in a bed file of genome intervals, and a genome wide map of restriction enzyme fragments (as generated by `capC-MAP genomedigest`). It finds the mid-point of each interval in the input file, and outputs the restriction enzyme fragment which it falls in. This can be useful for generating the targets file required by capC-MAP, where each target must appear in the genome wide fragments map.

A typical procedure to generate the targets file might be to

1. Run the capture oligo sequences through BLAST to find their locations within the reference genome.
2. Format the resulting list of locations into a bed file. Run this file though `capClocation2fragment` to find the list of restriction enzyme fragments to which the oligos map.
3. Edit the output bed file to add useful target names and remove duplicated entries (typically oligos will be designed such that there is one at either end of a restriction enzyme fragment).

Bibliography

[Hughes2014] Hughes J.R. *et al.*, "Analysis of hundreds of cis-regulatory landscapes at high resolution in a single, high-throughput experiment" *Nature Genetics* **46** (2014)

[Davies2016] Davies J.O. *et al.*, "Multiplexed analysis of chromosome conformation at vastly improved sensitivity" *Nature Methods* **13** (2016)

[capC-MAP] Brackley C.A. *et al.*, "capC-MAP : Analysis software for Capture-C data" (2018)

[Langmead2009] Langmead B. *at al.*, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome" *Genome Biology* **10** (2009)

[Martin2011] Martin M. "Cutadapt removes adapter sequences from high-throughput sequencing reads" *EMBnet.journal* **17** (2011)

[Li2009] Li H. *et al.*, "The Sequence Alignment/Map format and SAMtools" *Bioinformatics* **25** (2009)