4Cseqpipe for mapping and analyzing 4C-Seq experiments

Tanay, deLaat Groups

Version 0.7, May 2012

1, What is 4Csepipe and what do you need to run it.

4Cseqpipe provides support for analyzing 4C-seq experiments, including sequence extraction, mapping, normalization and generation of high resolution contact profiles around the “viewpoint”.

The current version can be applied on standard linux systems. It requires a standard R installation (version 2.13.1) and the "zoo" package. To install R see http://www.r-project.org/.

The pipeline includes support for processing and mapping fastq sequence files to the genome, a process that may require over 10 minutes and may take 3.5GB of RAM. Processed raw files can take up to 2GB per experiment and mapped files up to 2.7GB per experiment.

Components and archives: (see http://compgenomics.weizmann.ac.il/tanay/?page_id=99)

4cseq_pipe_manual.pdf – This file, describing the contents and operation of the 4C-seq data normalization and analysis pipeline.

4cseq_pipe.tgz – Contains the pipeline directory structure, pipeline script, mapper binary executable, configuration file, shared objects and associated scripts.

mm9_RE_seq.tgz – Restriction-enzyme-digested genomic sequence files appropriate for *NlaIII-DpnII and DpnII-Csp6I* 4C-Seq experiments. A similar archive for hg19 is also available.

mm9_genome_seq.tgz – The mm9 genome sequences (repackaging the standard mm9 assembly). A similar archive for hg19 is also available.

example_data.tgz – Contains the fastq files for two sample experiments performed using the alpha1 globin gene promoter as the viewpoint in mouse fetal liver and fetal brain, as well as expected resulting figures. These experiments are described in Van de Werken, Landan, et. al., 2012.

4cseq_primer_db.tgz – Contains a primer database for various combinations of 4-basepair restriction enzymes for both the mouse and human genomes.

4cseq_primer_db_6cut.tgz – As above, for experiments involving 6-basepair primary restriction enzyme and 4-basepair secondary restriction enzyme.

4c_primer_db_manual.pdf – Describes the format of the tables within the primer database.

2. Using 4Cseqpipe

4cseqpipe is a simple wrapper for several programs that take the output of a 4C-Seq experiment, map it to the genome, combine/normalize experiments and generate near-cis domainograms along with additional tables for further exploration.

An example of typical use follows. A pre-built database of restriction fragment locations and fragment end sequences is part of the standard download of the package. The standard download also provides example fastq files for two experiments. Let us assume one wishes to analyze one 4Cseq experiment, and that its sequenced reads are contained in a fastq file. First, edit the file rawdata/index.txt to specify the characteristics of this experiment (the default index file specifies an experiment in fetal liver cells

using the highly active alpha1 globin promoter as viewpoint, provided with the examples package as id number 1). Next type:

```
cd mypath/4cseqpipe(or a different location in which the installation tar file was uncompressed)
perl 4cseqpipe.pl -fastq2raw -ids 1 -fastq_fn examples/fastq/alpha_FL.fastq
```

This will extract the appropriate sequences (those that look like valid 4Cseq products for the given experiment) from the fastq file and parse them into a "raw" file. Then type:

```
perl 4cseqpipe.pl -map -ids 1
```

This will map the sequences in the raw file to the fragmented genome, and typically takes ~5 minutes, though the process may take >10 minutes, depending on one's hardware, the number of reads, and the experiment's mapability. Finally, type:

```
perl 4cseqpipe.pl -nearcis -calc_from 32000000 -calc_to 32300000 -stat_type
median -trend_resolution 5000 -ids 1 -figure_fn hba_FL.png -feat_tab
rawdata/alpha_globin_features.txt
```

This will generate a graphical depiction of the contact profile around the viewpoint, as described in Van de Werken, Landan et. al., specifying many parameters that can control the graphics and analysis, all of which are described below. The output figure and tables can be found in the directory `figures/`. The directory `examples/figures/` contains the "correct" output figure generated for the alpha globin data, for comparison.

One can compare these results to another experiment, using the same viewpoint in a different tissue (fetal brain, in which the alpha1 globin gene is not active):

```
perl 4cseqpipe.pl -fastq2raw -ids 2 -fastq_fn examples/fastq/alpha_FB.fastq
perl 4cseqpipe.pl -map -ids 2
perl 4cseqpipe.pl -nearcis -calc_from 32000000 -calc_to 32300000 -stat_type
median -trend_resolution 5000 -ids 2 -figure_fn hba_FB.png -feat_tab
rawdata/alpha_globin_features.txt
```

3. 4Cseqpipe reference

To use the pipeline run the script 4cseqpipe.pl with the options specified below. The configuration file 4cseqpipe.conf contains some default option values that can be edited and/or supplemented. For example the configuration option `index` points to an index file defining the 4C experiments and their characteristics, which is used by all pipeline stages (described in section 5.1.1). In order to override default options, supply the 4cseqpipe.pl script with options through the standard –x y format.

3.1. Building restriction site tracks

These tracks are relatively large tables specifying the locations and lengths of fragments and fragment-ends for a given combination of restriction enzymes. Note that restriction site tracks for the common enzyme combinations for the human and mouse genomes can be downloaded with the 4Cseq_pipe files. New tables can be constructed as follows (once per restriction enzyme combination, requiring <1Mb RAM and ~10 minutes):

```
perl 4cseqpipe.pl –build_re_db \
          –first_cutter XXXX \
```

```
          -second_cutters YYYY [,ZZZZ] \
          -trackdb_root root_directory
```

Where `first_cutter` is the DNA recognition sequence of the first cutter, and `second_cutters` is the list of second cutters you may use following this first cutter. This function creates three types of binary files per chromosome in the directory specified by `root_directory`, describing primary restriction fragment lengths, the distance between primary restriction site and adjacent 3' secondary restriction site, and the distance between primary and 5' secondary restriction site (section 5.2.3). To generate these files, the pipeline uses .seq files located in `root_directory/seq` (section 5.2.2), each of which contains the entire sequence of a single chromosome. The outcome of this stage is required by the near-cis normalization phase, but not by the mapper, which specifies the restriction fragment grid from scratch.

3.2 Converting fastq files to "raw" format:

```
perl 4cseqpipe.pl -fastq2raw  \
          -ids ID1[,ID2..] \
          -rawdir directory \
          -fastq_fn input_file_name \
          [-convert_qual] 1 or 0
```

Where `ids` are a set of experiment IDs, each pointing to a raw file specified in the index file (section 5.1.1), `rawdir` is the name of the directory storing raw sequence files (section 5.1.2), `fastq_fn` is the full path to the input fastq file and `convert_qual` indicates if base qualities should be converted by adding 31 to each value. See below for full details on the raw format and the index file.

3.3 Mapping valid 4C-Seq procucts to restriction fragments in the genome:

```
perl 4cseqpipe.pl -map \
          -ids ID1[,ID2,..] \
          -trackset_name trackset_name \
          -binsize bin_size \
          -trackdb_root database_file_name \
          -rawdir root_file_name
          [-read_length] read_length_in_bp
```

Where `ids` specifies experiment IDs as above, `trackset_name` specifies the directory storing experimental tracks (within `trackdb_root/tracks/`, see section 5.2.4), `binsize` specifies the size of genomic bins in basepairs, `trackdb_root` is the path to the directory that stores restriction site grids and 4C experiment tracks (sections 5.2.2 and 5.2.3), and `rawdir` is the location of the directory housing the index file (section 5.1.1) and raw data files (section 5.1.2). Optional `read_length` allows the user to use a specified number of prefix nucleotides for mapping (for example in the case of extremely long reads that may slow down mapping).

Input: a raw sequence file (extracted from the source fastq)
Output: a set of binary track files describing the coverage and multiplicity of each restriction fragment, as descirbed in 5.2.4, and a textual description of the mapping statistics (described in 5.5).

3.4 Normalizing and generating near-cis domainograms:

```
perl 4cseqpipe.pl -nearcis \
          -ids ID1[,ID2..] \
```

```
-trackset_name trackset_name \
-binsize bin_size \
-expname experiment_name \
-[nonunique] 0 or 1 \
-Rscript_path R_location
-[feat_tab] feature_tab_file_name \
-[stat_type] [median,..] \
-trend_resolution resolution_of_main_trend \
-[interval_type] [quantile or standard_deviation] \
-[interval_down] lower_bound_of_interval_behind_main_trend \
-[interval_up] upper_bound_of_interval_behind_main_trend \
-[truncate_percentile] upper_and_lower_percentiles_to_discard \
-[max_truncation] maximum_number_of_values_to_discard \
-horiz5 horizon5' \
-horiz3 horizon3' \
-calc_from 5'_start_coordinate \
-calc_to 3'_start_coordinate \
-[plot_from] 5'_start_coordinate \
-[plot_to] 3'_start_coordinate \
-[max_height] ylim \
-figure_fn figure_file_name \
-[color_def_script] color_definition_script
```

This is a relativelty flexible and parameter-rich interface for generating normalized contact profiles. Many parameters can be changed, but it is possible to initially use default values (as examplified in section 2).

Parameters:

`ids:` specifies experiment IDs as above

`trackset_name:` specifies the directory storing experimental tracks as above,

`binsize:` specifies the size of genomic bins in basepairs.

`expname:` allows the user to provide an alternative experiment name instead of the default ID number when naming the output figure files.

`nonunique:` determines whether only unique or only nonunique fragment ends will be used; the default is 0, corresponding to unique fragment ends.

`Rscript_path:` indicates the location of R, used to generate the figures.

`feat_tab:` the full path to a list of genomic features and their coordinates. This file (which can be empty if not needed) is a tab delimited table with the fields *from, to, name,* and *color*, (a header is required, see `rawdata/feat_tab.txt` for an example). For each feature, a triangle is positioned at the top of the figure at the midpoint between the *from* and *to* coordinates, labeled with the appropriate name and specified color.

`stat_type:` indicates the type of statistic to calculate and plot, with the following options:

median – calculates the median of values in sliding windows of size 2-50kb. This is the default statistic.

mean – as above, but calculates the mean.
linear_mean – performs a linearly weighted mean, wherein values are weighted by their distance from the center of the window to the window's edge. A value at the very center is multiplied by 1, values at the two edges are multiplied by 0.
log_mean – calculates the average in log-space before transforming the result back to linear-space. Equivalent to geometric mean.
trunc_mean – calculates a standard mean, but truncates high and low values according to options **truncate_percentile** and **max_truncation**. **truncate_percentile** defines the upper and lower cutoffs (e.g. "5" means the top and bottom 5 percentiles will be discarded); the default value is 2.
max_truncation defines the maximum number of values that will be discarded from each direction (top or bottom, not total) given **truncate_percentile**; the default values is 4.
linear_trunc_mean – performs linear weighting on truncated data.
trunc_log_mean – calculates the geometric mean of truncated data.

`trend_resolution` is the resolution (size of window in basepairs) at which the main trend will be plotted, with resolutions from 2-50kb plotted below the main trend using a color-code (see Van de Werken, Landan, et. al., 2012 for details).

`interval_type` defines how confidence intervals are to be plotted above and below the main trend. The following options are available:
quantile – determines the lower and upper bounds according to options **interval_down** and **interval_up**, which are provided as percentiles (e.g. 20 = 20th percentile, usage: `-interval_type quantile -interval_down 20 -interval_up 80`). The default values are 20th and 80th percentiles. This option is available for all *stat_types*.
standard_deviation – determines the lower and upper bounds according to **interval_down** and **interval_up**, which are provided as multiples of the standard deviation of the mean (e.g. 2 = 2*standard deviation, usage: `-interval_type standard_deviation -interval_down 1 -interval_up 1`). The default values are 1 standard deviation above and below the mean, which is calculated according to the *stat_type* chosen. This option is not available when using `-stat_type median`.

`horiz5` and `horiz3`: indicate how far to plot upstream and downstream in basepairs.

`calc_from` and `calc_to` allow the user to define the genomic range for which a normalized trend will be computed using exact chromosomal coordinates.

`plot_from` and `plot_to` define plot boundaries using chromosomal coordinates; when omitted these parameters default to *calc_from* and *calc_to*. These parameters are applicable when one wishes to calculate trends using a larger window than one intends to plot.

`max_height` dictates the maximal value on the y-axis (i.e. ylim). The default value is 1, corresponding to the maximal value of the *main_trend*.

`figure_fn` is the name of the output png file, which will be stored in the `figures/` directory.

`color_def_script` allows the user to alter default color parameters for the output plot, including the color of points, the color and width of lines, and the color gradient of the multi-scale display below the main trend (see `scripts/color_def_script.r` for an example).

Input: the binary tracks describing a mapped 4C-Seq experiment
Output: A set of tables representing the normalized contact profile (section 5.4) and a near-cis domainogram in the figures directory.

3.5 Running the entire pipeline:

```
perl 4cseqpipe.pl –dopipe \
           -all parameters above [...]
```

Running `dopipe` with all required and desired parameters will execute `fastq2raw`, `map`, and `nearcis` successively.

4. Installation and required packages

`4cseq_pipe.pl` requires Perl and R installation (with the "zoo" package). These are available on most Linux systems and can be installed from the R and Perl websites. The configuration `Rscript_path` can be modified in case one wishes to use a specific R version and not the system default. Otherwise, the 4cseqpipe files can be extracted into any directory with read/write permissions and do not require any privileged system access.

5. Directories, files and settings

5.1 *rawdata* - Contains the raw experimental sequences (converted from fastq, see sections 3.2 and 5.1.2)

5.1.1 *rawdata/index.txt* – Defines the 4C experiments to be analyzed. A tab delimited text file with the following fields:

ID - Numerical identifier of the experiment
First_cutter_sequence – 4bp DNA sequence
Second_cutter_sequence – 4bp DNA sequence
Bait_chromosome – The name should be compatible with the chromosome sequence files in `trackdb_root` (e.g. *chr7*)
Bait_coordinate – Starting chromosomal coordinate of the 4C primer
Raw_sequence_file – Full path for the raw sequence file containing the reads for this experiment (see sections 3.2 and 5.1.2)
Lane – The lane number of the experiment. Currently set this to 0.

5.1.2 raw sequence file – These are stored in a simple tab delimited file format, including one line per read, with the first field containing the lane number (in the current version – always 0), the second field containing the basecalling quality values and the third field containing the sequence itself. Note that quality values are assumed to be stored in Phred format. Once can convert fastq files to raw format (including an option for converting basecalling qualities) using the pipeline script (section 3.2).

5.2. Trackdb

This directory stores chromosomal sequences, restriction site grids and 4C experiment tracks. Experiment tracks are stored in a simple binary format used internally in the Tanay group. Each track is a directory containing one file for each chromosome. The chromosome file is a binary vector of floats, with the first 4 bytes storing the resolution of the track (bin_size) in bps. The pipeline uses tracks as intermediates. Additional tools for analyzing tracks will be made available as part of a larger software distribution from the Tanay group.

5.2.1. `trackdb/chrom_sizes.txt` – A tab delimited file specifying the names and sizes of chromosomes in basepairs.

5.2.2. `trackdb/tracks/seq` –Stores genome sequence, one file per chromosome.

5.2.3. `trackdb/tracks/re` –Stores data on restriction site distribution in several tracks:

`XXXX_fraglen` – Stores the fragment lengths of the fragment pool generated by a cutter with recognition site XXXX. The track consists of “NA”s in all bins except for the bins at the center of restriction fragments.

`XXXX_dist2YYYY_3prime (_5prime)` – Stores the fragment end length (or the distance between the primary restriction site XXXX and the secondary one YYYY, looking at the 3 prime (or 5 prime) end of the primary restriction fragment.

5.2.4 `trackdb/tracks/DATASET` (where DATASET is any name, as specified in the configuration file) – This is where the results of the 4C mapper are stored. We use four tracks per experment:

`ID_5cov, ID_3cov` – Store the total sequence coverage of each fragment, using statistics from the 5' fragment end or 3’ fragment end. The data is stored in tracks that include “NA” in all bins except for bins at the center of primary restriction fragments.

`ID_5multi, ID_3multi` – These tracks contain the multiplicity of each fragment end in the genome, given a particular restriction enzyme combination. “NA”s are present in all bins except those in the middle of the primary restriction fragment. Values of “1” indicate that the 3’ or 5’ fragment end was mapped uniquely. Note that fragment ends that are not mapped uniquely are still resolved by the mapping algorithm according to a weighted model – see Van de Werken, Landan, et. al., 2012 for details.

5.3 `scripts/, bin/` - Perl and R scripts implementing the pipeline, and the mapper program (implemented in C++). Currently these are poorly document – updates will be available from the Tanay group.

5.4 `tables/` – Stores normalized near-cis profiles and normalization factors. There are three tables and two normalization factors:

ID.nearcis.cover.txt – The mapped coverage data (see Methods section in manuscript). Each row represents a genomic bin of size bin_size (see section 3.3 on Mapping). Includes the following tab-delimited columns:

intervalID – always 1 in the current implementation
chrom – the viewpoint chromosome
start – the starting coordinate of the relevant genomic bin
eID_3 – the value of ID_3cov (see section 5.2.4) in the genomic bin
eID_5 – as above but ID_5cov
eID_3m – the value of ID_3multi in the genomic bin
eID_5m – as above but ID_5multi
eID_fl – the length of the fragment associated with the genomic bin
eID_fe3 – the length of the 3' fragment end
eID_fe5 – the length of the 5' fragment end

ID.nearcis.norm.txt – The normalized intensity data (see Methods section in manuscript). Each row represents a genomic bin of size bin_size. Includes the following tab-delimited columns:

1 – the starting coordinate of the genomic bin.
2 – normalized contact intensity of 3' fragment end if associated fragment is non-blind
3 – normalized contact intensity of 5' fragment end if associated fragment is blind
4 – normalized contact intensity of 3' fragment end if associated fragment is non-blind
5 – normalized contact intensity of 5' fragment end if associated fragment is blind

ID.nearcis.norm.median.scales – Median (or other statistic, see Section 3.4 above) of normalized contact intensity in 1kb windows. Each row represents a genomic bin of size 1kb. Includes the following tab-delimited columns:

1 – the starting coordinate of the 1kb bin.
2 – the median of values within a 2kb window, centered on the 1kb genomic bin.
3 – the 80th quantile of values in the window (or other statistic, see Section 3.4 above).
4 – the 20th quantile of values in the window (or other statistic)
5-148 – as columns 2-4, but with increasing window size, going up to 50kb.

ID.nearcis.norm.txt.median.normfactor_t – The normalizing factor by which data in the table **ID.nearcis.norm.median.scales** is divided to generate the main trend line. Corresponds to the maximum median value at the resolution defined by the user for the main trend line (default 5kb), and ensures the data range from 0-1.

ID.nearcis.norm.txt.median.normfactor_m – The normalizing factor by which data at all scales in the table **ID.nearcis.norm.median.scales** is divided to generate the multi-scale color-coded domainograms. Corresponds to the maximum median value at a resolution of 12kb

5.5 `stats/` – Stores mapper statistics per experiment, including the number (and percent of total when applicable) of *Total reads, Mapped reads, Ignored reads* (these are reads that account for more than 2% of the data and represent either self-ligation of the viewpoint fragment or random PCR domination, and are therefore not considered in the normalization), the *Ignored sequence(s)* (the sequences of the above *Ignored reads*), *Low quality reads, Cis reads, Total sites, Unique sites, Sites with multiplicity between 1 and 10,* and *Sites with multiplicity great than 10.*

5.6 `tmp/` –Stores intermediates

5.7 `figures/` – Stores output near-cis figures

5.8 Pipeline script and its configuration file

`4cseqpipe.pl` - The pipeline script, implemented in perl

`4cseqpipe.conf` - Default configuration file