SICER 1.1

1. Introduction

For details description of the algorithm, please see

"*A clustering approach for identification of enriched domains from histone modification ChIP-Seq data" Chongzhi Zang, Dustin E. Schones, Chen Zeng, Kairong Cui, Keji Zhao, and Weiqun Peng*, **Bioinformatics** 25, 1952 - 1958 (2009)

If you use SICER to analyze your data in a published work, please cite the above paper in the main text of your publication.

For questions about usage, please sign up, check and post at google group **SICER users**. For suggestions and comments, please email chongzhizang@gmail.com and wpeng@gwu.edu

2. Installation

Installation of SICER only requires unpacking the files in the SICER.tgz file. Prerequisites include:

2.1 Install the numpy and scipy packages. More information on this can be found at: http://www.scipy.org/. To check whether numpy and scipy are properly installed, please run

```
$python
>>> import numpy
>>> import scipy
```

If there is no error message, this step is done. For further discussion on installation of scipy and numpy, please see 4.1 in Chapter 4: Additional Notes.

2.2 Define environment variables. Please open the master shell scripts (SICER.sh, SICER-rb.sh, SICER-df.sh, SICER-df-rb.sh), replace {PATHTO} in the definition of $SICER with the directory where you want your SICER to be. For alternative approaches, please see Chapter 4: Additional notes.

3. Running SICER

SICER first and foremost is a filtering tool. Its main functions are

1. Delineation of the significantly ChIP-enriched regions, which can be used to associate with other genomic landmarks.
2. Identification of reads on the ChIP-enriched regions, which can be used for profiling and other quantitative analysis.

Caution: please do not run multiple instances of SICER scripts originated from the same directory in parallel. Each instance of SICER script generates temporary files with hard-coded names. If multiple instances of SICER scripts originated from the same directory are run in parallel, their temporary files would interfere with each other, resulting in absurd outcomes. If you run multiple instances of SICER simultaneously, please make sure to run them under separate directories to avoid interference.

The raw data need to be in the BED format. See the test.bed file in the ex/ directory for an example.

3.1: Running SICER with a control library: SICER.sh

Most of the important parameters are on command line. There are 11 command line parameters.
$ sh **DIR**/SICER.sh ["InputDir"] ["bed file"] ["control file"] ["OutputDir"] ["Species"] ["redundancy threshold"] ["window size (bp)"] ["fragment size"] ["effective genome fraction"] ["gap size (bp)"] ["FDR"]

Here the DIR shall be replaced by the directory of SICER.sh in practice.

Meanings of the parameters that are not self-explanatory:

- Species: allowed species and genome versions are listed in GenomeData.py. You can add your own species and/or genome versions and relevant data there.
- Redundancy Threshold: The number of copies of identical reads allowed in a library.
- Window size: resolution of SICER algorithm. For histone modifications, one can use 200 bp
- Fragment size: is for determination of the amount of shift from the beginning of a read to the center of the DNA fragment represented by the read. FRAGMENT_SIZE=150 means the shift is 75.
- Effective genome fraction: Effective Genome as fraction of the genome size. It depends on read length.
- Gap size: needs to be multiples of window size. Namely if the window size is 200, the gap size should be 0, 200, 400, 600, ….

Let's use the example under ex/ to illustrate the procedure and the outcome. There are two raw bed files test.bed and control.bed under the directory /ex. We want to find significant islands with a redundancy threshold of 1, a window size of 200bp, a gap size of 600bp and FDR=1E-2. The shell script, run_SICER.sh, for running SICER.sh in this particular case can be found under ex/.

There are a number of output files.

- test-1-removed.bed: redundancy-removed test bed file
- control-1-removed.bed: redundancy-removed control bed file
- test-W200.graph: summary graph file for test-1-removed.bed with window size 200, in bedGraph format.
- test-W200-normalized.wig: the above file normalized by library size per million and converted into wig format. This file can be uploaded to the UCSC genome browser
- test-W200-G600.scoreisland: an intermediate file for debugging usage.
- test-W200-G600-islands-summary: summary of all candidate islands with their statistical significance. It has the format:

  chrom, start, end, ChIP_island_read_count, CONTROL_island_read_count, p_value, fold_change, FDR_threshold

- test-W200-G600-islands-summary-FDR.01: summary file of significant islands with requirement of FDR=0.01.
- test-W200-G600-FDR.01-island.bed: delineation of significant islands in "chrom start end read-count-from-redundancy-removed-test.bed" format
- test-W200-G600-FDR.01-islandfiltered.bed: library of raw redundancy-removed reads on significant islands.
- test-W200-G600-FDR.01-islandfiltered-normalized.wig: wig file for the island-filtered redundancy-removed reads. This file can be uploaded to the UCSC genome browser and be compared with the track for test-W200-normalized.wig for visual examination of parameter choices and SICER performance.

Of all these files, the test-W200-G600-islands-summary-FDR.01 and test-W200-G600-FDR.01-island.bed are most important for further analysis. The first one contains the details about each significant island. The second one contains the redundancy-removed raw reads filtered by islands. In addition, the two wig files shall be used for visual examination of the raw and processed data on the genome browser.

Note:

- The choice of window size and gap size has a large effect on outcome. In general, the broader the domain, the bigger the gap should be. For histone modifications H3K4me3, W=200 and (g=1 window) are suggested. For H3K27me3, W=200 and (g = 3 windows) are suggested for first try. If even bigger gap size is found to work better, you might also want to try increasing the window size (eg, window size = 1K, and gap size = 3 windows)
- Additional details and adjustable parameters can be found in the SICER.sh script, which allow further tailoring for advanced uses.
- The FDR is calculated using p-value adjusted for multiple testing, following the approach developed by Benjamini and Hochberg. The value reported in the last column is the biggest FDR threshold under which the region is deemed significant. The significance values of all candidate islands are listed in file test-W200-G600-islands-summary. If you want to try a different FDR without changing other parameters, there is no need to run the entire SICER.sh again. Only the last substep in SICER.sh needs to be rerun, which can be done by commenting out the previous substeps.
- You can also use p-value or fold-change or combination of them to control for significance. They are all reported in test-W200-G600-islands-summary as well.

3.2: Running SICER without a control library: SICER-rb.sh

There are 10 command line parameters:

$sh DIR/SICER-rb.sh ["InputDir"] ["bed file"] ["OutputDir"] ["species"] ["redundancy threshold"] ["window size (bp)"] ["fragment size"] ["effective genome fraction"] ["gap size (bp)"] ["E-value"]

Here the DIR shall be replaced by the directory of SICER-rb.sh in practice.

An example of shell script for running SICER-rb.sh, run_SICER-rb.sh, with a redundancy threshold of 1, a window size of 200bp, a gap size of 400bp and E-value=100 can be found under ex/.

There are a number of output files.

- test-1-removed.bed: redundancy-removed test bed file
- test-W200.graph: summary graph file for test-1-removed.bed with window size 200 in bedGraph format.
- test-W200-normalized.wig: the above file normalized by library size per million and converted into wig format. This file can be uploaded to the UCSC genome browser
- test-W200-G400-E100.scoreisland: delineation of significant islands controlled by E-value of 100, in "chrom start end score" format
- test-W200-G400-E100-islandfiltered.bed: library of raw redundancy-removed reads that are on significant islands.
- test-W200-G400-E100-islandfiltered-normalized.wig: wig file for the island-filtered redundancy-removed reads. This file can be uploaded to the UCSC genome browser and be compared with the track for test-W200-normalized.wig for visual examination of choices of parameters and SICER performance.

Of all these files, the test-W200-G400-E100.scoreisland and test-W200-G400-E100-islandfiltered.bed are most important for further analysis. The first one contains the delineation of each significant island. The second one contains the redundancy-removed raw reads filtered by significant islands. In addition, the two wig files shall be used for visual examination of the raw and processed data on genome browser.

Note:

- The choice of window size and gap size has a large effect on outcome. In general, the broader the domain, the bigger the gap should be. For histone modifications H3K4me3, W=200 and (g=1 wondow) are suggested. For H3K27me3, W=200 and (g = 3 windows) are suggested for first try. If even bigger gap size is found to work better, you might also want to try increasing the window size (eg, window size = 1K, and gap size = 3 windows)
- E-value is not p-value. Suggestion for first try on histone modification data: E-value=100. If you find ~10000 islands using this evalue, an empirical estimate of FDR is 1E-2.
- Additional details and adjustable parameters can be found in the SICER-rb.sh script, which allow further tailoring for advanced uses.

3.3: Running SICER to identify differentially enriched regions

A frequently encountered case in epigenomic analysis is to identify significant changes between two conditions: wild type cells vs treated cells; normal cells vs pathological cells; un-differentiated cells vs differentiated cells. To deal with these situations, we developed SICER-df.sh and SICER-df-rb.sh. In the following, we will use wild-type (WT) vs Knock-out (KO) as an example.

3.3.1 SICER-df.sh

SICER-df.sh applies to the situation where there are two pairs of libraries: 1) wild-type (WT) and its control and 2) Knock-out (KO) and its control. The basic strategy is to 1) identify significant islands using SICER.sh in each of the two pairs. 2) Merge the two sets of significant islands. The merged islands constitute the units for comparison. 3) On each merged island, level in KO is compared with that in WT to determine the significance of changes.

Copy the shell script SICER-df.sh to the directory where the bed files are stored.

$sh SICER-df.sh ["KO bed file"] ["KO control file"] ["WT bed file"] ["WT control file"] ["window size (bp)"] ["gap size (bp)"] ["FDR for KO vs KOCONTROL or WT vs WTCONTROL"] ["FDR for WT vs KO"]

In addition to the output files from SICER.sh for each of the two pairs, output files also include:

- $UNIONISLAND: the set of merged islands
- $MERGEDISLANDSUMMARYFILE: This file stores the summary of merged islands, including read counts, normalized read count, p-value, fold change and BH-corrected p-value
- $INCREASEDISLANDS: This file stores the summary of islands with significantly increased level in KO identified with BH-corrected p-value criterion.
- $DECREASEDISLANDS: This file stores the summary of islands with significantly decreased level in KO identified with BH-corrected p-value criterion.

The last three files have the following columns:

- Chrom
- start
- end
- Readcount_KO
- Normalized_Readcount_KO
- Readcount_WT
- Normalized_Readcount_WT
- Fc_KO_vs_WT
- pvalue_KO_vs_WT
- FDR_KO_vs_WT
- Fc_WT_vs_KO
- pvalue_WT_vs_KO
- FDR_WT_vs_KO

Note:

1) If you want to try a different significance criterion without changing other parameters, there is no need to run the entire SICER-df.sh again. Only the last substep in SICER-df.sh needs to be rerun, which can be done by commenting out the previous substeps.

2) Islands with very high read count are likely to received very small p-value even with modest changes. An additional fold-change cut-off might be desirable to select most significantly differentiated regions.
3) In the current scheme, the control libraries are used only in delineation of islands. They do not affect the significance of change between WT and KO. Further development is needed to remove this limitation.

3,3.2 SICER-df-rb.sh

SICER-df-rb.sh applies to the situation where there are two libraries: 1) wild-type (WT) and 2) Knock-out (KO). The basic strategy is to 1) identify significant islands using SICER-rb.sh in each library. 2) Merge the two sets of significant islands. The merged islands constitute the units for comparison. 3) On each merged island, level in KO is compared with that in WT to determine the significance of changes.

Copy the shell script SICER-df.sh to the directory where the bed files are stored.

$sh SICER-df-rb.sh ["KO bed file"] ["WT bed file"] ["window size (bp)"] ["gap size (bp)"] ["E-value"] ["FDR"]

The additional output files are similar to those of SICER-df.sh

4. Additional Notes

4.1 **Installation of scipy**. Installation of scipy package turned out not to be an easy task under certain distributions of linux. An option is to install the free SAGE distribution package (http://www.sagemath.org). This is not a pure python distribution per se, but "inside the SAGE directory lies a local/ folder containing all the binaries, libraries and Python packages it used. It even contains its own Python. Set the PATH, LD_LIBRARY_PATH and PYTHONPATH environment variables right and suddenly you have a perfectly consistent installation of everything that's needed to do scientific work in Python! Other users on the same machine just need to change the same variables, and they can play too! Apotheose²! So in addition to its primary goals of providing a replacement for Mathematica/Maple/etc, SAGE, as a side-effect, provides the whole Python scientific shebang compiled and wrapped up in a nice package, for your pleasure." (from http://vnoel.wordpress.com/2008/05/03/bye-matlab-hello-python-thanks-sage/) . Another way to get it installed properly is to use the Enthought python distribution (http://www.enthought.com). The 32bit version is free for educational use.

4.2 **Alternative approaches to defining $PYTHONPATH**.

4.2.1 You can define $SICER and $PYTHONPATH as global environment variables, so that the modules under /lib are always recognized and the python modules can run on their own without shell script. To do this, Please edit Utility/setup.sh and replace {PATHTO} with the directory under which you will put SICER. Then incorporate the content in setup.sh into the bash configuration file .bash_profile under your home directory. After pasting the content to .bash_profile, please run

$source .bash_profile

Then all your newly created shells will know $SICER and lib/ .
To check, please run
$echo $SICER

$echo $PYTHONPATH

Note setup.sh is applicable only to bash. If you use other shells, contents in setup.sh needs to be modified accordingly.

4.2.2 The above approach depends on the shell used. A shell-independent approach is to insert a sitecustomize.py under ${pythondir}/lib/site-packages/. sitecustomize.py is a special script; Python will try to import it on startup, so any code in it will be run automatically. If sitecustomize.py does not exist, then add it to ${pythondir}/lib/site-packages/. If sitecustomize.py exists under ${pythondir}/lib/site-packages/, then edit it.In sitecustomize.py, please add (if not there already)

```
import sys
sys.path.append("{PATHTO}/SICER/lib")
```

4.2.3 If none of the above works, copy the modules under /lib and /utility to /src, then you are good to go.

4.3 **Additional utilities**. There are a number of modules under utility/, quite useful for additional analysis:

- calculate_cross_correlation_long_range.py: used by fragment-size-estimation.sh
- convert_summary_to_bed.py: extract the positional information from a summary file (e.g., test-W200-G600-islands-summary-FDR1E-3) and make a bed file that can be read by BED.py: chr start end ChIP_island_read_count
- filter_raw_tags_by_islands.py: identify all reads that are on significant islands
- filter_summary_graphs.py: used by islands_statistics_pr.py
- fragment-size-estimation.sh: Estimating ChIP fragment size by cross correlation between watson and crick tags. Caution: time consuming.
- find_overlapped_islands.py: compare two sets of islands and identify unique and overlapped ones
- get_windows_histogram.py: generate window read-count statistics
- islands_statistics_pr.py: generate island score and length statistics
- slice_raw_bed.py: randomly sample a given number of reads from a raw read library for saturation analysis.

4.4 **Customization of Genome**.

4.4.1 Information about the genomes is stored in GenomeData.py. If what your genome is not listed there, you can add it on your own following the template.

4.4.2 If you want to add a species or genome, four things need to be done in GenomeData.py:
1) Add a list of chromosomes as ${SPECIES}_chrom
2) Add a dictionary of the length of each chromosomes as ${SPECIES}_chrom_lengths
3) Modify the dictionary species_chrom by appending an element directing the genome name to the chrom list.
4) Modify the dictionary species_chrom_lengths by appending an element directing the genome name to the chrom_lengths dictionary.

4.4.3 If you do not want chrM in your analysis, simply delete chrM from the entries in the GenomeData.py.

4.5 **Effective Genome size**.

This can be calculated using the software describe in*The uniqueome: a mappability resource for short-tag sequencing* by Ryan Koehler, Hadar Issac, Nicole Cloonan, and Sean M. Grimmond, *Bioinformatics* (2011) 27 (2): 272-274.

4.6 **BED format**.

The output of SICER for the delineation of islands does not follow the convention of BED format in the strictest sense. In the canonical BED format, the end position is not included, whereas we have the end position included.