StereoGene
Software Introduction

Contents

1 Introduction....1
2 Software availability....1
3 Source code and license....2
4 Input and output....2
5 How to run: an example....2
6 The calculation workflow....3
6.1 Preprocessing....3
6.2 Computing correlation....3
6.3 Results output....5
7 Advanced features....5
7.1 Filtering....5
7.2 Partial correlation....6
7.3 Models as input data....6
7.4 Genomic scale parameters....7
7.4.1 Kernel width....7
7.4.2 Window size....7

1 Introduction

High throughput sequencing methods produce massive amounts of data. The most common first step in the interpretation of these data is to map the data to genomic intervals and then overlap with genome annotations. A major interest in computational genomics is a spatial genome-wide correlation among genomic features (e.g. between transcription and histone modification). The key hypothesis here is that features that are similarly distributed along a genome may be functionally related.

Here, we propose the StereoGene method that rapidly estimates genomewide correlation of two genomic annotations. The method enables correlation of continuous data, to avoid the loss of data that occurs upon reduction to intervals. To include analysis of nonoverlapping but spatially related features, we use kernel correlation. Another novel powerful feature of our approach is the local correlation track output that enables overlap with other correlations (correlation of correlations).

2 Software availability

The StereoGene C++ source code, program documentation, Galaxy integration scrips and examples are available at the project homepage at http://stereogene.bioinf.fbb.msu.ru/ and at the github repository https://github.com/favorov/stereogene.

The repository includes four folders:

- src is the source folder
- www contains a fresh copy of the project web page
- galaxy contains the stub files for Galaxy (https://galaxyproject.org/) integration
- example is a set of files for a minimalistic run

3 Source code and license

The software is provided in a form of C++ source code with a makefile for GCC compiler. The code was tested both with GCC versions 4 and 5. After cloning or downloading and unpacking the source, type 'make' from the shell in the source folder.

Our source code is covered by MIT artistic license. We use FFT code by Jens Jorgen Nielsen, its license is shown in the `mixfft.c` source file.

4 Input and output

The program reads standard (bed, wig, etc) formats for the features to be correlated and a tab-delimited text file with chromosome name and length on each line, as the chromosome length file. The output is a set of text files. A pair of them (`statistics` and `parameters`) form a report that grows line-by-line with each StereoGene execution in the current working folder. Others are text files that correspond to a naming scheme (see below) that allow batch post-processing. If requested (by `Rscript` parameter), StereoGene generates R script and an R markdown template that generate a report in human-readable form. The R itself is not run by the program, it is supposed to be done by the user.

5 How to run: an example

StereoGene has a lot of parameters that modify the software behavior. All parameters have default values that work in most situation. A run requires two track files and a file with chromosome lengths. The example below is started from the example folder. First, copy or link the StereoGene executable to the example folder. Then, in a shell prompt, say:

```
./StereoGene chrom=chromLength H3K4me1.bed H3K4me1.bed
```

There are more examples or run is `example.sh` and `example-cnf.sh` files in the examples (see http://stereogene.bioinf.fbb.msu.ru/, the StereoGene web page).Briefly, aside from the interval files, `chrom` is the only necessary parameter, specifying the name of the chromosome length file. Any parameter can be shown in ‘par=value’ form or in ‘-par value’ as a command-line switch, or as par=value line in the configuration file. The configuration file is referenced to by `-cfg` command-line parameter. Important parameters are introduced below. A table with a comprehensive list of parameters is provided below.

6 The calculation workflow

Once started, the software first preprocesses the input data and then it executes the requested calculations. The preprocessing step results are saved to save time. Subsequent runs of StereoGene on the same track can reuse the preprocessing results.

6.1 Preprocessing

The input is read from the track text file in one of the standard formats for genomic tracks (BED, WIG, bedGraph, and broadPeak) and then it is converted into a proprietary binary format file for correlation analysis. The conversion averages all the input profiles in small tiling windows (bins) defined in the bin parameter. The values are scaled to an integer value [0..250] to fit into a single byte. The scaling can be either linear (y=ax) or logarithmic (y=a log (1+scaleFactor * x)). The scaling mode is either set by the `scale parameter` or chosen automatically (recommended) between linear and logarithmic by the input data analysis.

The software allows users to refer to specific properties of specific data types. For example, for genes the user can additionally specify what elements of annotation are to be used, intron, exons, gene or intron starts (`intervals` parameter). Moreover, the user can specify whether to use the strand information. The BroadPeak format standard defines a list of possible values for each nucleotide position, and the user can specify which of this BroadPeak data types are to be used (`bpType` parameter).

The software allows to users to combine several input tracks in a linear combination and use the results of the linear model as one of the tracks used for correlation analysis.

6.2 Computing correlation

Instead of correlating the whole genome, we calculate the correlation of the preprocessed tracks within large windows of fixed size (~1,000,000 bp). The size of these windows is specified in two input parameters (`wSize` and `wStep`). In these parameters, `wSize` is the width of the window and `wStep` is the distance between starting coordinates of each window. If `wSize` equals to `wStep` (default), the windows are tiling and do not overlap. The windows are smaller than whole chromosomes to increase the efficiency and to obtain a distribution of values from which to estimate correlation p-values. `wSize` is a technical parameter. If it is too large, there are too few windows for a statistical estimate and the program is too slow, still, if `wSize` is too low, we get a very wide variety of local correlation measure instead of a common, whole-genome correlation. We recommend using `wSize` = `wStep` = 100k..1M.

The data on the input tracks are routinely are gapped. Some wig files carry zeros. They also contain genomic regions not covered with data. There are several ways to interpret this missing data. The coverage of an area could actually be unknown, for example, because of a mapping problem. On the other hand, the level could really be 0. Depending on how we interpret these areas (`NA` parameter) we can fill them with zeroes or with random values defined by `noiseLevel` parameter. The data that are written to the profile is defined by formula *x*=Norm(*e, noiseLevel*sd*), where Norm – is a random normally distributed value, *e*, *sd* – mean and standard deviation of the input data. If the input data are highly sparse, windows may contain only 0 or NA values. The correlation in these windows is undefined, hence the program ignores them. There are parameters (`maxZero`, `maxNA`) that define which windows are to be treated as noninformative.

If the DNA strand is known, `complFg` parameter determines whether strand information is accounted for in the correlation. The finite size of windows could cause marginal effects, thus, to avoid them, the user can ask the program to add random-signal flanks to each window (`flankSize` parameter). Still, our experience shows that the flanks have minimal influence on the correlation for large window sizes (e.g. the default 100,000 bp).

It is challenging to analytically estimate the null distribution of the correlation coefficients that accounts for dependencies of all features in the neighboring positions of the genome. We use permutation to test the statistical reliability of the result. To do this, we randomly sample pairs of windows, first from one track, next from the other, and assess the correlation value for each pair. The number of sampled pairs is defined by parameters (`nShuffle`, `MaxShuffle`, `MinShuffle`). The first defines the number of samples as a percentage of the number of informative windows (default is 100%). For too sparse data, 100% could be insufficient; on the other hand, if the data are too dense, 100% could be time-consuming. For more fine adjustment, the user can use `MaxShuffle` and `MinShuffle` parameters that post upper and lower limits for the number of sampled pairs.

Another usecase to compare the result with a null model in the case when we work with coverage tracks that are supplied with the input track for control is to calculate the concordant window pairs correlation distribution (foreground distribution) for a pair of features (tracks) and to compare it with the same distribution for a feature and the input of other feature or with the distribution for input-to-input comparison. In this case, the background calculation is for all the runs is switched off by the `corrOnly=ON` switch.

6.3 Results output

The main results of the program are the foreground Kernel Correlation (KC) for coherent windows, the background KC over the shuffled windows (null distribution), and the p-value for the difference in the distribution between the foreground and background KC. Each value in foreground and background distribution is a correlation (real value in [-1..1] band) of the two features in a pair of windows, coherent for foreground or randomly sampled for background. The program reports the total correlation as well (the average correlation over the coherent windows).

The program provides a variety of additional result output types. First, the details of each of the run of StereoGene are saved into two files: `params` and `statistics`. Each run has a unique id, which hashes the parameters and the time of the start. A single run appends a line to the `params` and `statistics`. Each line starts with the run id. The results also can be written as XML to `statistics.xml` file if it is required by the `outRes` parameter. We append to the files rather than rewriting to make the batch runs simpler.

The set of the output files is defined by several input parameters. The `outDistr` parameter outputs the distribution of the KC values when setting ON (default) or excludes them when setting OFF to *.dist file. The `outDistr`= OFF mode is useful when the user needs only the correlation values, one number per comparison. Distances specify that the program to output cross-correlation function (CCF) between input tracks computed chromosome-wise (DETAIL), the common CCF (TOTAL), or not to output it (NONE) to *.dist file. When `Rscript` setting is ON, StereoGene writes an R script that prepares the report with plots and results included. The `outLC` parameter asks to output local correlations in a wig track and to write the distribution of local correlations to a text file.

7 Advanced features

There are a variety of additional options that the program can be run with in addition to the standard options described above. One can switch off the permutation-based background calculation by the setting the `corrOnly` parameter from its default value of OFF to ON . This option can be useful to reduce computational cost if only the correlation estimation is of the interest (e.g. in a batch multi-sample run preparing a distance matrix) or if the user wants to use another foreground distribution as a background for current run, e.g. to compare the correlations of a feature with signal of some NGS experiment and its correlation with the input of the same experiment.

7.1 Filtering

The data can be filtered by a threshold (`threshold`) in the [0..250] interval or by proximity to another annotation (`map`, `mapIv`). The program outputs the autocorrelation function for each input track if `lAauto parameter` is not `0`.

7.2 Partial correlation

The two-track correlation can be calculated in a space that is orthogonal to some third track. In other words, partial correlation of two tracks conditional the third is calculated. This usecase is to remove a confounder, which is represented by the third track, from the data correlation. The third track data is provided in the same way as the two correlated tracks.

7.3 Models as input data

Instead of each track data, a linear combination of tracks (a model) can be used. Stereogene identifies *.mod and *.model files as these models. For batch mode, lists of tracks can be specified as the two input parameters for tracks and all the calculation will be run for each pair of a track from first and second lists of tracks. To check this mode we prepare two models.

model1.mod: **`1 * f_br.H3K4me3.wig`**

model2.mod: **`-1* f_br.H3K4me3.wig`**

Then we run the Stereogene for fetal Brain f_br.H3K4me3 vs models and obtain expected results:

inputs	Total correlation	Average correlation
model1.mod vs f_br.H3K4me3	1.0000	0.9999
model2.mod vs f_br.H3K4me3	-1.0000	-1.0000

7.4 Genomic scale parameters

7.4.1 Kernel width.

To show the influence of the Kernel width on the program result we run StereoGene for the Fetal brain H3K27me3 vs gene starts from RefSeq annotation. The results are presented in the following Table 1. The dependence of the Kernel width is obvious.

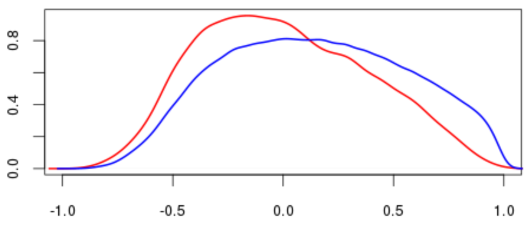
Table 1. Fetal Brain H3K4me3 vs gene-starts. Widow size=1Mb

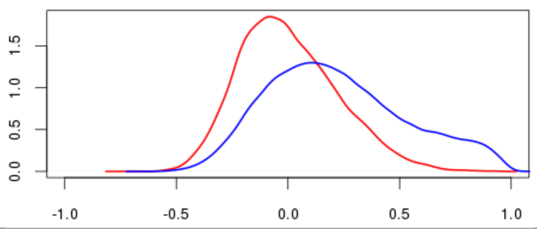
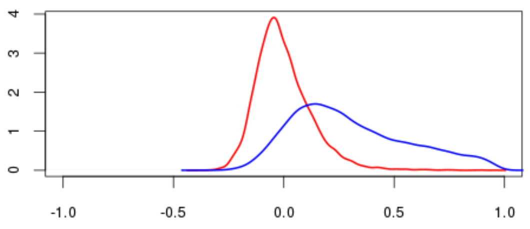
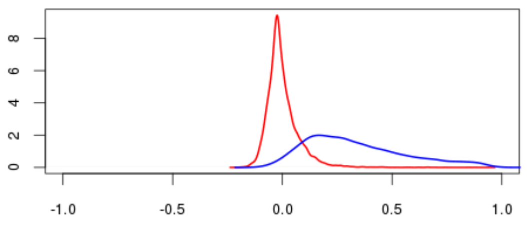
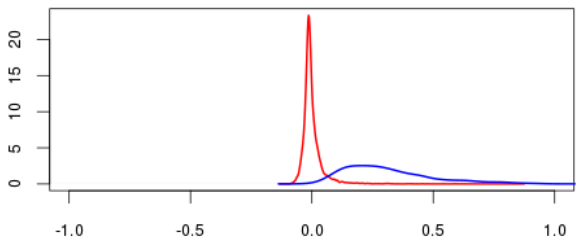
Kernel width, bp	total KC	average-KC	p-val
10	0.07	0.09	0
100	0.11	0.12	0
1000	0.36	0.42	0

7.4.2 Window size.

To understand the dependence of the results on the window size we run the program for H1 cells H3K4me3 vs H3K27me1 with different window sizes. As the window size became greater than 30k the total KC does not change. The average KC changes slightly after the window size became greater than 100k. The foreground and background distributions of KC over the windows became more narrow as the window size increases. If small windows are used, a variety of correlations in the windows becomes large and the distributions became wide. The default window size is wSize>100k. But if the data are very sparse the number of significant non-zero windows may become small to get appropriate statistics. In this case, a smaller window size recommended.

Table 2. Dependence of the results on the window size for H1 H3K4me3 vs H1 H3K27me3: Kernel width=1000 bp

Window	total KC	av-KC	p-val	time[*], s	Distributions of KC over the windows
10000	0.14	0.15	0[**]	10.1	

30000	0.14	0.22	0	11.2	
100000	0.14	0.30	0	13.3	
300000	0.15	0.33	0	14.6	
1000000	0.14	0.31	0	15.5	

[*]without preprocessing
[**] p-value of 0 actually means that p-value is lower that the precision; still we did not change it in output for the sake of parsing