LogitGRO Software Manual

Joey Azofeifa
University of Colorado Boulder
Computer Science
Dowell Laboratory
joseph[dot]azofeifa[at]colorado[dot]edu

We present a fast and simple algorithm to detect nascent RNA transcription in Global nuclear run on sequencing (GRO-seq). GRO-seq is a relatively new protocol that captures nascent transcripts from actively engaged polymerase, providing a direct read out on bona fide transcription. More traditional assays, such as RNA-seq, measure steady state RNA levels and are therefore a combination of transcription, post-transcriptional processing, and RNA stability. A detailed study of GRO-seq data has the potential to inform on many aspects of the transcriptional process. GRO-seq data, however, presents unique analysis challenges that are only beginning to be addressed. Here we describe a new algorithm, LogitGRO, that takes advantage of two popular machine-learning techniques, a hidden markov model (HMM) and logistic regression, to robustly classify which regions of the genome are transcribed. Our algorithm builds on the strengths of previous approaches but is accurate, dependent on very little training data, robust to varying read depth, annotation agnostic, and fast.

Contents

1 Downloading and Compiling

1.1 Zip File and Compiler Type

Please visit `https://github.com/jgazofeifa/LogitGRO.git` to download the zipped file of LogitGRO. This zipped file contains the source and header files to compile. Navigate to where you will want to store the software and unzip the file

-bash-$ gunzip LogitGRO.zip

1.2 Compiling with make

Please check your compiler type. To do so on Mac OS and Linux machines, navigate to terminal and type:

-bash-$ g++ –version

your GNU compiler should be later in date that 4.2..1
If this is not the case, download either the latest version of xcode or GNU compiler from `https://developer.apple.com/xcode/` , `http://www.gnu.org/software/gcc/releases.html` resprectively.
Once you have placed the unzipped files into a directory, navigate to the the src/ directory within the LogitGRO pack /path/to/LogitGRO/src/ and type

-bash-$ make

Please note that if you move this package to another place on your OS, you will need to recompile by typing

-bash-$ make clean

and then

-bash-$ make

2 Running Unit Tests

Once you have compiled the source code, you can run a unit-test to make sure the algorithm can run on your OS. To run, navigate to the src/ direction and type

-bash-$ ./segment –run-unit-test

Error codes will be reported if necessary and can be looked up in the error code book. If errors persist please contact Joey with your OS and compiler type. If all works well you should see this:

********** All tests passed **********

and you will be good to go!

3 File Types and Training Sets

3.1 BedGraph File Format

LogitGRO segments strands in an independent fashion that is the amount of read coverage on the anti-sense strand does not affect the probabilistic classification of the sense strand. For this reason, LogitGRO takes as input bed files that are either positive or negative strand but not both.
All lines of the input file format should read:
chromosome [tab-delimiter] genomic coordinate start [tab-delimiter] genomic coordinate stop [tab-delimiter] read coverage over interval
Example:

chr1 100 150 4

Lastly this file should be sorted by chromosome and sorted by starting coordinate in each chromosome.

This is exactly the format of a bedGraph file (`http://www.genome.ucsc.edu/goldenPath/help/bedgraph.html`) however a bed file (`http://genome.ucsc.edu/FAQ/FAQformat.html`) can be manipulated into this parsing structure as well. Either way the file extension of the input file should have .BedGraph at the end (capitalization important)

The header of the input bed file should indicate which strand it is coming from. For example if your file is called GRO_SEQ_INPUT.BedGraph
then there should be two files

GRO_SEQ_INPUT.pos.BedGraph or GRO_SEQ_INPUT.+.BedGraph

and

GRO_SEQ_INPUT.neg.BedGraph or GRO_SEQ_INPUT.-.BedGraph

for the negative and positive strand respectively. These are fed to LogitGRO separately (two command calls).

3.2 Training Files

The power of the LogitGRO algorithm is that it learns from the data. There are two ways to do this: (1) feed a list of genomic coordinates you consider true nascent transcription and a list of genomic coordinates you consider true inactive transcriptional noise or (2) a list of genes you consider active and a list of genes you consider inactive. Indeed these lists could come from a third party experiment such as RNA-seq or Microarray experiments. Or can manually gathered. We recommend the latter as this gives closer to what you, the user, would want by intuition. Indeed, little training data. We recommend 3 intervals considered active and 3 intervals considered inactive. Increasing the number of training examples will not hurt your classifications but you will begin to diminishing returns on the quality of your classifications.

3.2.1 Option 1

If you are attempting to classify a file called: GRO_SEQ_INPUT.pos.BedGraph then the training file should read:

FILE:[tab-delimiter]GRO_SEQ_INPUT.pos.BedGraph[new-line-delimiter]
+[tab-delimiter]chr1[tab-delimiter]100[tab-delimiter]900000[tab-delimiter]1[new-line-delimiter]
+[tab-delimiter]chr1[tab-delimiter]900000[tab-delimiter]905100[tab-delimiter]0[new-line-delimiter]

In this example, you have specified the region on the positive strand, on chromosome 1 from position 100 to 900000 to be active (denoted by 1) and the region on the positive strand on chromosome 1 from position 900000 to 905100 to be inactive (denoted by 0). The header "FILE:[tab-delimiter]GRO_SEQ_INPUT.pos.BedGraph[new-line-delimiter]" is important as it points to which GRO-seq experiment you want to learn from.

3.2.2 Option 2

This option allows you to feed in a list of genes or refseq annotations you consider inactive or active. The gene names can either by NM_ accession codes or gene common names. The file format is very similar to Option 1.

FILE:[tab-delimiter]GRO_SEQ_INPUT.pos.BedGraph[new-line-delimiter]
RAB17A[tab-delimiter]1[new-line-delimiter]
CDK1A[tab-delimiter]0[new-line-delimiter]

Here you consider the entire length of RAB17A to be showing characteristics of active nascent transcription and the entire length of CDK1A to be showing characteristics of inactive nascent transcription.

3.2.3 Training Sensitivity and Default Parameters

LogitGRO uses a sophisticated learning algorithm that optimizes to your specific dataset. However, if you miss-label an active interval as inactive or vice versa, you will see odd behavior.

If you wish to run the algorithm without specifying a training set that is okay. Here is a list of three GRO-seq experiments with their read depths and optimized parameter values from []. If your experiment matches in read depth to one of these experiments you might consider simply running with these pre-specified parameters

4 Parameter Use

Here, I provide a discussion into LogitGRO parameter use. For the theory behind the algorithm please see the LogitGRO paper[]. There are two ways to run the same command either by a single '-' followed by a one or two letter key or by a '--' followed by a full word exposition. * indicate *required* parameters

4.1 Running Unit Tests and Help

Please send all unresolved issues to Joey at joseph[dot]azofeifa[at]colorado[edu]

-t ; --run-unit-test

This should be ran once following source code compilation. If any error codes are reported back please look them up in the error code book in the following section. If errors persist, please contact Joey at information above.

-h ; --help

This allows you to get a briefer description of parameters available for use with this software. For help while you are running commands in the instant.

4.2 Input / Output

As stated in the file format sections above, file format is important and in general should be a tab-delimitated files with the appropriate headers.

-i ; --input-bedgraph-file*

This specifies the full path to the BedGraph File and should either be positive or negative strand. LogitGRO takes common Linux shortcuts such as and ./ The file extension of the BedGraph File must be ".BedGraph" and filename body should indicate whether this is the positive or negative strand by ".+." or ".pos." for the positive strand and ".-." or ".neg." for the negative strand.

-uf; --name-of-IGV-file

This allows the user to specify the name of the outputted IGV file. This could be particularly useful if you are rerunning the with multiple training sets. The default removes the .BedGraph file extension and appends _IGV.bed to the user inputted BedGraph file.

-l ; --learn

This option tells the algorithm that you are going to learn the parameters from training data. This parameter takes as input the full file path to the training file (appropriate formats with headers and indexing are outlined above. This directory and file must exist either LogitGRO will throw an error.

-o ; --output-directory

This option allows you to specify where you want the IGV Bed file outputted to on your OS. This parameter takes as input a full directory path. This directory path must exist either either LogitGRO will throw an error. Default is the segmentation outputs.

-eRNA ; --predict-eRNA

This option can only be run once you have classified the positive and negative strand. As such this parameter takes three inputs: (1) The IGV output for the positive strand (2) the IGV output for the negative strand and (3) a float number that indicates the confidence interval of the normal distribution defining the amount of positive and negative strand overlap (default = 0.05 which indicates a confidence interval of 95%). These three input parameters are separated by a space and follow one another in the above order.

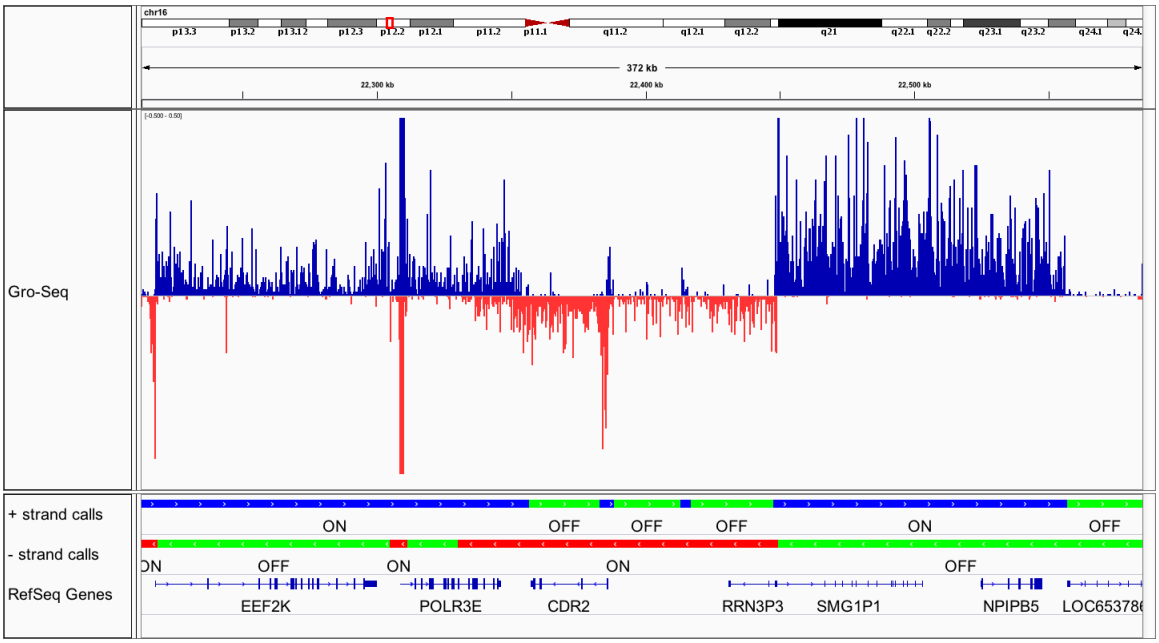

Figure 1: LogitGRO output.

The output of two Bed files are visualized within IGV showing a sub-region in chromosome 1. The first track shows typical GRO-seq data from the HCT116 dataset, the positive strand in blue, the negative strand in red. LogitGRO output is below for each strand. Green indicates areas of inactive transcriptional activity, blue represents areas of active transcription along the positive strand and red along the negative strand. RefSeq annotations are shown at the bottom.

-st ; --strand

If you have not specified the strand type in the bedGraph file name, you can specify here. This takes as input either "+" or "-"

-rf ; --refseq-file

Path to the refseq file you would like to compare the LogitGRO classifications to. There is both the hg18 and hg19 in the refseqannotations/ directory. And so this parameter takes as input either hg18 or hg19 to specify these already existing annotation files or a full path to the refseq annotations file you would like to use. Please keep in mind that this file must be in the *exact same format* as the files specified in the refseqannotations/ directory. Default is hg19.

4.3 Algorithm Parameters

There are a few ways to modify the algorithmic behavior of LogitGRO active and inactive classifications.

-d ; --feature-dimension

This parameter takes as input a integer and must be greater than 1 and less than 9. The number corresponds to the number of contig features that LogitGRO considers.

Table 1: Feature Vector $\vec{x}$. y_i equals the i^{th} read count between [t,t+l] where t is the genomic start of the contig and l the length of the contig. Y equals the ordered set of read counts between [t,t+l].

$\vec{x}$-dimension	formalism	description
x_1	1	Bias Term
x_2	l	Contig Length
x_3	$\frac{1}{l}\sum_{i=t}^{t+l} y_i$	mean
x_4	$\frac{1}{l-1}\sum_{i=t}^{t+l}(y_i - mean)^2$	variance
x_5	$\sum_{i=t}^{t+l}(y_i)$	total
x_6	$center(Y)$	median
x_7	$max(Y)$	max
x_8	$min(Y)$	min

If -d = 3 then the bias term, contig length and mean number of read counts is considered across the contig, similary if -d = 5 then the bias term, contig length, mean, variance and total number of read counts is considered (see Table 1 for complete description of feature dimensions and orderings). The default parameter is 2.

-al ; --alpha-learning-rate

The theta parameters are estimated from the training data via the Raphson-Newton method.

$$\theta^{t+1} = \theta^t - \alpha \cdot H^{-1} l(\vec{\theta}, D) \cdot \nabla l(\vec{\theta}, D) \tag{1}$$

alpha is a real-valued number that designates the speed of the update rule. This particularly important if you have many training examples and want a fast convergence time (alpha ¿ 1) or if you want an accurate, more precise convergence (0 ¡ alpha ¡1) the default is set at 1. An alpha value to large (¿10) may give poor parameter estimates.

-w ; --logisitic-regression-weights

This parameter is set when –learn is not indicated. This specifies in order the weights of logistic regression parameters. Indeed specifying -w 0.01 0.005 0.09 means that you are indicating that for the bias term, contig length and average read count coverage the corresponding theta values are 0.01 0.005 0.09.

-a ; --self-transition-HMM-weight

A 2 state markov process is defined by 4 parameters. However, if the markov process is symmetric and erogdic than you need only one parameter which defines the probability of remaining in a state. Setting this parameter to 0.9 means that 90% of the time you will stay in the active state and inactive state and 10% of the time you transition from the active to inactive state or inactive state to the active state. setting this parameter forces the markov to be symmetric we certainly encourage use of the baum welch as this will learn non-symmetric transition parameters.

5 Error Codes

(1)

Path to src is not set properly. Consider recompiling by 'make clean' and then 'make'.

(2)

Could not find: TEST_DMSO2_3.pos.BedGraph. Please make sure the examples directory exists and that TEST_DMSO2_3.pos.BedGraph is placed inside.

(3)

Could not find: training_examples/testTrainingFile.tsv. Please make sure the training_examples/ directory exists and that testTrainingFile.tsv is placed inside.

(4)

Could not find: segmentation_outputs/ directory. Please make sure segmentation_outputs directory exists.

(5)

Could not find: refSeqAnnotations/refSeqGene_hg19.txt. Please make sure refSeqAnnotations directory exists and that refSeqGene_hg19.txt is placed inside.

(6)

Could not create contigs and coverage statistics for TEST_DMSO2_3.pos.BedGraph.

(7)

Could not read binary format. Please check your compiler version and make sure it is above 4.2.1/

(8)

Was unable to gather training data from training_examples/testTrainingFile.tsv.

(9)

The Raphson - Newton method did not learn / output parameter values for logistic regression.

(10)

The Baum-Welch method did not learn / output parameter values for the HMM stochastic transition matrix

(11)

Viterbi algorithm didn't run.

(12)

The IGV bed files was not written to segmation_outputs/