MIRZA-G Documentation

Release 2.0.0

Rafal Gumienny

Jan 11, 2017

Contents

Contents:

CHAPTER 1

Instalation

1.1 Dependencies

1.1.1 MIRZA

Download and install MIRZA. It is statically compiled so you do not need to compile it again

1.1.2 CONTRAfold

Download and install CONTRAfold. It might be that you experience an error when compiling CONTRAfold. Something like this:

```
In file included from LBFGS.hpp:52:0,
                 from InnerOptimizationWrapper.hpp:12,
                 from OptimizationWrapper.hpp:12,
                 from Contrafold.cpp:16:
LBFGS.ipp: In instantiation of 'Real LBFGS<Real>::Minimize(std::vector<T>&) [with␣
↪Real = double]':
OptimizationWrapper.ipp:260:9:   required from 'void OptimizationWrapper<RealT>::
↪LearnHyperparameters(std::vector<int>, std::vector<T>&) [with RealT = double]'
Contrafold.cpp:451:9:   required from 'void RunTrainingMode(const Options&, const std:
↪:vector<FileDescription>&) [with RealT = double]'
Contrafold.cpp:68:54:   required from here
LBFGS.ipp:112:105: error: 'DoLineSearch' was not declared in this scope, and no␣
↪declarations were found by argument-dependent lookup at the point of instantiation␣
↪[-fpermissive]
LBFGS.ipp:112:105: note: declarations in dependent base 'LineSearch<double>' are not␣
↪found by unqualified lookup
LBFGS.ipp:112:105: note: use 'this->DoLineSearch' instead
make: *** [Contrafold.o] Error 1
```

To fix it:

- add -fpermissive flag to CSXXFLAGS in Makefile:

```
CXXFLAGS = -O3 -DNDEBUG -W -pipe -Wundef -Winline --param large-function-
↪growth=100000 -Wall -fpermissive

instead of
CXXFLAGS = -O3 -DNDEBUG -W -pipe -Wundef -Winline --param large-function-
↪growth=100000 -Wall
```

- add in Utilities.hpp:

```
#include <limits.h>
```

1.1.3 Jobber

Download and setup Jobber python library for workflow managment.

```
pip install Jobber
```

After installation start the Jobber daemon:

```
$ nohup jobber_server > jobber.log 2>&1 &
```

Note: If you installed Jobber as user you might not have an access to the jobber_server. By default the binary location is $HOME/.local/bin and you have to export it in bash:

```
$ export PATH="$HOME/.local/bin:$PATH"
```

or add this statement to .bashrc file.

jobber_server produces ~/.jobber/jobber.pid file that indicates whether the Jobber is already running. If the file exists one cannot start new instance of the jobber_server. This file is not clean when jobber_server is killed - only when it was stopped with stop command. Thus, after some crash one have to remove this file in order to start jobber_server again.

This will automatically create a ~/.jobber and ~/jobber/log directories and it will put there config.py and executers.py files. Look at them and adjust according to your needs.

This should create a jobber.sqlite file next to config.py where jobs will be stored (all in ~/.jobber). Now you can create pipelines that will be managed with a python script.

To stop the jobber daemon, run following command:

```
$ jobber_server -stop
```

You can watch and control your jobs and pipelines present in the database using simple we interface. To launch it type:

```
$ jobber_web
```

or

```
$ jobber_web --ip Your.IP.addres --port YourPort
```

Note: If you would like to run MIRZA-G pipeline locally without DRMAA change executer in config.py file from “drmaa” to “local”

1.1.4 Python

Install python modules:

- Jobber (see upper paragraph)

- drmaa (if you are going to submit it to the cluster)
- statsmodels
- pandas
- BioPython
- dendropy
- numpy
- scipy

1.2 Download

The pipeline is available as a git repository on GitHub:

```
git clone https://github.com/guma44/MIRZAG.git
```

By default we provide 3'UTR sequences without alignments. If you would like run MIRZA-G with conservation you need to download alignments to this particular 3'UTR set.

```
wget http://www.clipz.unibas.ch/public/mirza/MIRZAG_alignments.tar.gz
```

You can also download whole package including alignments from out website:

```
wget http://www.clipz.unibas.ch/public/mirza/pipeline_MIRZAG.tar.gz
```

CHAPTER 2

Usage

2.1 Basic usage

Command to launch the pipeline is as follows:

```
python MIRZA_G_pipeline.py run --config congig.ini --name-suffix name_of_the_run
```

All parameters for the script:

The main script launching MIRZA-G analysis pipeline

```
usage: MIRZA-G [-h] {run,clean} ...
```

Sub-commands:

run Run a pipeline

```
usage: MIRZA-G run [-h] [-v] --config CONFIG [--name-suffix NAME_SUFFIX]
                   [--protocol {seed,scan}] [--calulate-bls]
                   [--modules [MODULES [MODULES ...]]]
```

Options:

-v=False, --verbose=False Be loud!

--config Config file

--name-suffix=test_run Suffix to add to pipeline name in order to easily differentiate between different run, defaults to test_run

--protocol=seed Protocol of MIRZA-G, defaults to seed

Possible choices: seed, scan

--calulate-bls=False NOT AVAILABLE: Calculate Branch Length Score (conservation)

--modules A list of modules to load (if HPC or environment requires)

clean Clean after previous run

```
usage: MIRZA-G clean [-h] [-v] [-y]
```

Options:

-v=False, --verbose=False Be loud!

-y=False, --yes=False Force deletion of files.

2.2 Preparing config file

Copy config_example.ini from MIRZA-G directory to your working directory (directory where you want to perform calculation, WD):

```
cd Your/Working/Direcory
cp Path/To/MIRZA-G/config_example.ini config.ini
```

Set all the necessary paths in your config.ini file as indicated in the comments inside the file. The most importand are:

- **motifs**: "Path/To/miRNAs.fa" - abs path to an input fasta file with mi/siRNA sequences of length 21 or more
- **seqs**: "Path/To/MIRZA-G/data/UTR_Sequences.fa" - abs path to a fasta file with the UTR sequences from which the coordinate file will be generated (you can use 3'UTR sequences in the MIRZA-G/data directory, for this file there are also alignments for conservation precalculated)
- **mirza_binary**: "MIRZA" - path to MIRZA binary (or how you invoke it in the bash)
- **contrafold_binary**: "contrafold" - path to CONTRAfold binary (or how you invoke in the bash)

Models paths:

- **model_with_bls**: "Path/To/MIRZA-G/data/glm-with-bls.bin" - abs path to the model with BLS (you can find it in the pipeline/data directory)
- **model_without_bls**: "Path/To/MIRZA-G/data/glm-without-bls.bin" - same as before

Additionally when you would like to calculate with evolutionary conservation you have to make sure that the variable run_only_

- **phylogenetic_tree**: "Path/To/MIRZA-G/data/human_tree.nh" - abspath to provided phylogenetic tree
- **alignment_directory: "Path/To/MIRZA-G/data/HumanAlignments/" - abspath to provided human alignments direct**
  this directory is already in the MIRZA-G directory. If you downloaded from GitHub you have to download it additionally.

If you would like to run it on cluster follow instructions in the configuration file and ask your admin what parameters you need to set up before (like DRMAA path, modules necessary, queues names etc.). All these parameters can be set up in config.ini.

To run it locally it takes ~70 to 90 seconds for one miRNA without conservation calculation and ~170 seconds with calculation (This might be substantial amount of time (up to half an hour per miRNA) for worse processors).

2.3 Example

To test the pipeline go to the tests directory and run:

```
cd Path/To/MIRZA-G/tests
bash rg_run_test.sh help
```

Note: Usage: rg_run_test.sh clean/run [MIRZA/binary/path] ['CONTRAfold/binary/path']

And if you have installed MIRZA and CONTRAfold to default locations (MIRZA and contrafold) run:

```
bash rg_run_test.sh run
```

Otherwise provide paths to **BOTH** of them:

```
bash rg_run_test.sh run Path/To/MIRZA/binary Path/To/CONTRAfold/binary
```

CHAPTER 3

Pipeline flow

3.1 1. Prepare miRNAs for pipeline

Prepare miRNA fasta file for MIRZA i.e. for each miRNA sequence check if it is 21 nucleotide long and if not eliminate it. It also replaces all u or U into T. In the same time it splits miRNAs into separate files.

```
usage: rg_prepare_mirnas_for_mirza_and_split [-h] [-v] --input INPUT
                                             [--output-dir OUTPUT_DIR]
```

Options:

-v=False, --verbose=False Be loud!

--input Input miRNA file in fasta format.

--output-dir=Output Directory for split files, defaults to Output

3.2 2. Chunk UTRs

Take UTRs and generate fragments by sliding window that can be fed into MIRZA

```
usage: rg_generate_utr_chunks [-h] [-v] [--input INPUT] --output-dir
                              OUTPUT_DIR [--part-size PART_SIZE]
                              [--window-size WINDOW_SIZE]
                              [--slide-size SLIDE_SIZE]
```

Options:

-v=False, --verbose=False Be loud!

--input=<open file '<stdin>', mode 'r' at 0x7fc466cc80c0> Input file in fasta format. Defaults to sys.stdin.

--output-dir Output directory for split files

--part-size=40000 Number of sequences per part, defaults to 40000

--window-size=50 Length of the window for MIRZA, defaults to 50

--slide-size=20 Size of the window slide, defaults to 20

3.3 3. Generate miRNA expressions

Generate expressions of miRNAs. This file is required by MIRZA and is composed just from the miRNA ID and its expression. Here we set up expression for each miRNA to 1.

There is no special script dedicated to this function. It is just the bash command:

```
cat input | ruby -ne 'puts "#{$_.rstrip()[1..-1]}\t1" if $_.start_with?(">")' > output
```

3.4 4a. Run MIRZA analysis

If the option "scan" as a miRNA target search is chosen putative targets are selected based on MIRZA interaction energy. Thus, this is the part of the pipeline scans the 3'UTRs with MIRZA to find it.

This script generates the jobs of the pipeline that will do the analysis on the split files.

```
usage: run_mirza_scan [-h] [-v] --config CONFIG --group-id GROUP_ID
                      --input-dir INPUT_DIR --working-dir WORKING_DIR
```

Options:

-v=False, --verbose=False Be loud!

--config Config file

--group-id Group Id

--input-dir Input and output directory

--working-dir Working directory of the pipeline. Required because this file is launched from ~

3.4.1 I. Calculate coordinates with MIRZA

Here the MIRZA algorithm is used to calculate the energy between miRNA and the 3'UTR fragments generated before. MIRZA is launched from the command:

```
MIRZA expressions.tab mrnas.fa mirnas.fa 50 noupdate
```

And the result is piped to the analysis script:

Take MIRZA output and arrange it in proper way

```
usage: rg_extract_data_from_mirza_output [-h] [-v] [--output OUTPUT] --seqs
                                         SEQS --threshold THRESHOLD
                                         [--context CONTEXT]
```

Options:

-v=False, --verbose=False Be loud!

--output Output file in Tab format.

--seqs UTR sequences in fasta format

--threshold Threshold for the score

--context=50 Context for sequence to print, defaults to 50

3.4.2 II. Merge and filter results

The results are merged with bash command and duplicate results resulting from eg. overlapping 3'UTR fragments or transcripts from the same gene.

Filter duplicates in coordinates by id, miRNA and sequence

```
usage: rg_filter_duplicates_from_scan [-h] [-v] --coords COORDS --split-by
                                      SPLIT_BY --index-after-split
                                      INDEX_AFTER_SPLIT --output OUTPUT
```

Options:

-v=False, --verbose=False Be loud!

--coords Coordinates

--split-by Split id by the string

--index-after-split After split take this column as new id, 0 based

--output Output name

3.5 4b. Run seed scan

This part is launched if the "seed" options is chosen when starting the pipeline. The putative miRNA targets are found by simple seed match.

Count miRNA seed in the provided sequences according to chosen definition

```
usage: rg_count_miRNA_seeds_and_filter_duplicates [-h] [-v] [--motifs MOTIFS]
                                                  [--seqs SEQS]
                                                  [--how {ElMMo,TargetScan,6-mer}]
                                                  [--output OUTPUT]
                                                  [--context CONTEXT]
                                                  --split-by SPLIT_BY
                                                  --index-after-split
                                                  INDEX_AFTER_SPLIT
```

Options:

-v=False, --verbose=False Be loud!

--motifs miRNA/siRNA sequences to use when scanning

--seqs Sequences for scaning eg. 3' UTRs

--how=TargetScan What definition for seed to use

Possible choices: ElMMo, TargetScan, 6-mer

--output=coords.tab Name of the output file , defaults to coords.tab

--context=50 Context for sequence to print, defaults to 50

--split-by Split id by the string

--index-after-split After split take this column as new id, 0 based

3.6 5. Run features analysis

This is the main part of the pipeline where all the features are calculated.

3.6.1 I. Calculate MIRZA

Calculate MIRZA interaction energy and MIRZA-based Branch Length Score (conservation) for the provided coordinates of putative targets.

```
usage: rg_calculate_MIRZA [-h] [-v] [--onlyMIRZA {yes,no}] [--seq SEQ]
                          [--out OUT] [--coords COORDS] [--motifs MOTIFS]
                          [--tree TREE] [--mln-dir MLN_DIR] [--threshold THR]
                          [--contextLen CONTEXTLEN] [--mirzabin MIRZABIN]
                          [--reforg REFORG]
```

Options:

-v=False, --verbose=False Be loud!

--onlyMIRZA=no Calculate only MIRZA score for given coordinates

Possible choices: yes, no

--seq=seqs.fa Fasta with mRNA sequences

--out=output.tab, -o=output.tab Output table

--coords=coords.tab File with target sites positions, miRNA and target gene ID

--motifs= Fasta file with miRNA sequences

--tree= Phylogenetic tree of the species used in alignment file

--mln-dir= Directory with multiple alignment files

--threshold=20.0 Threshold for MIRZA score

--contextLen=50 Length of the context sequence serounding binding site

--mirzabin= Path to the MIRZA binary

--reforg=hg19 Reference organism to which alignments are performed: default: hg19

3.6.2 II. Calculate accessibility with CONTRAfold

A CONTRAfold algorithm is used to calculate accessibility of target site for miRNA.

```
usage: rg_calculate_contrafold [-h] [-v] [--seq SEQ] [--out OUT]
                               [--coords COORDS] [--contextLen_L CONTEXTLEN_L]
                               [--contextLen_U CONTEXTLEN_U]
                               [--context CONTEXT] [--contrabin CONTRABIN]
```

Options:

-v=False, --verbose=False Be loud!

--seq=seqs.fa Fasta file with mRNA sequences , defaults to seqs.fa

--out=output.tab.gz output file, defaults to output.tab.gz

--coords=coords.tab file with target sites positions, miRNA and target gene ID

--contextLen_L=0 length of the context sequence downstream binding site to be unwinded

--contextLen_U=0 length of the context sequence upstream binding site to be unwinded

--context=50 length of the context of the seed to be checked

--contrabin=contrafold Path to CONTRAfold binary

3.6.3 III. Calculate flanks composition

Calculate flanks composition using provided coordinates.

```
usage: rg_calculate_flanks_composition [-h] [-v] [--seq SEQ] [--out OUT]
                                       [--coords COORDS]
                                       [--contextLen CONTEXTLEN]
```

Options:

-v=False, --verbose=False Be loud!

--seq=seqs.fa fasta with mRNA sequences

--out=output.tab output table

--coords=coords.tab file with target sites positions, miRNA and target gene ID

--contextLen=50 length of the context sequence serounding binding site

3.6.4 IV. Calculate distance to the boundary

Calculate distance to the boundary based on provided coordinates for targets.

```
usage: rg_calculate_distance [-h] [-v] [--seq SEQ] [--out OUT]
                             [--coords COORDS] [--contextLen CONTEXTLEN]
```

Options:

-v=False, --verbose=False Be loud!

--seq=seqs.fa fasta with mRNA sequences

--out=output.tab, -o=output.tab output table

--coords=coords.tab file with target sites positions, miRNA and target gene ID

--contextLen=50 length of the context sequence serounding binding site

3.7 6. Merge and add probabilities

Merge all results into one features table

```
usage: rg_merge_results_add_probability_and_calculate_per_gene_score
       [-h] [-v] [--inputs INPUTS] [--coords COORDS] [--model-bls MODEL_BLS]
       [--model-nobls MODEL_NOBLS] [--output OUTPUT] [--only-mirza {yes,no}]
       [--threshold THRESHOLD] [--split-by SPLIT_BY] [--column COLUMN]
       [--name NAME]
```

Options:

-v=False, --verbose=False	Be loud!
--inputs	Coma-separated list of input paths
--coords	Cooridinate file used in the beginning
--model-bls	Path to model with branch length score
--model-nobls	Path to model without branch length score
--output	Output file name
--only-mirza	Calculate only MIRZA and DON'T calculate MIRZA BLS Possible choices: yes, no
--threshold=0.12	Threshold for summing, defaults to 0.12
--split-by=NONE	If the header of fasta has multiple annotations eg. transcript_id|entrez_id|wikiname split it and take only one, defaults to NONE
--column=0	0 based column number to take after splitting, defaults to 0
--name=GeneID	Name of the id eg. gene, transcript etc , defaults to GeneID

3.8 7. Collect results

In the end all the results are collected with bash command:

```
zcat output_dir/\*.score > cwd/mirza_g_results_protocol.tab
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search