imGLAD Documentation

Release 1.0

Juan C. Castro

Mar 10, 2017

CHAPTER

ONE

TABLE OF CONTENTS:

1.1 Getting started

1.1.1 What is imGLAD?

imGLAD is a computational tool for detection of bacterial genomes in metagenomic datasets. imGLAD has two steps fitting a model and predicting data. The fitting step simulates metagenomic datasets and randoly spikes them with the target genome. This is used to create a logistic model to discriminate positive samples from negative samples. The prediction step uses the model paramters to estimate the likelihood of presence of the target genome in a given metagenomic sample.

1.1.2 Obtaining imGLAD

Clone the git repository

```
$> git clone https://github.com/jccastrog/imGLAD
```

You can also download the zip file from the GitHub site https://github.com/jccastrog/imGLAD

1.1.3 System requirements

Python 2.7 or higher.

ART 2.5.8 or higher.

Either BLAST 2.2.28 (or higher) or BLAT (any version)

Python requirements

numpy, scipy, biopython, gzip, screed, statsmodel (optional)

1.2 Model fitting

imGLAD attempts to solve a binary classification problem. The positive examples in this case are all the samples were the target genome is present. The negative examples are all the samples were the negative genome is absent. `fitModel.py` generates a series of metagenomic datasets, 50% of this metagenomic datasets will contain reads from the target genome, these are labeled as positive samples. The remaining 50% will **not** contain any reads from the target genome. These sets of samples are used to separate positive from negative samples using a logistic model.

1.2.1 General options

For a target genome a logistic model is built. The target genome is provided in a Fasta file. The associated options are.

-t The target genome to be detected in FASTA format.

-sp The species of the genome written as binomial name in quotations. (e.g. "Escherichia coli")

The option `-t` refers to the file of the target genome. The option `-sp` refers to the species name, this field is used to avoid the inclusion of closely related gnoes in the training set (see below). If the species name is unknown this field can be left blank `''`.

1.2.2 The training dataset

Metagenomic datasets are generated by random generation of reads from the selected genomes. The associated options are.

-l GENOMES The genomes list to create the training dataset

-s TRAIN_SIZE Number of genomes included in the training dataset (Incompatible with -l). (default : 200)

-e TRAINING_EXAMPLES Number of training examples (metagenomic datasets) used to train the model by default 200 (100 positive and 100 negative examples. (default : 100)

-j PLATFORM The sequencing platform used to generate reads in the datasets. (default : illumina)

-r NUM_READS Number of reads per training example (Metagenomic dataset simulated). (default : 1000000)

-d READ_LENGTH Average read length for the simulated datasets. (default : 150)

1.2.3 Using your own list of genomes

If you need a simulated metagenome with a specific set of genomes from NCBI you can provide imGLAD with a specific list of the accesion numbers for the genomes. imGLAD will download the genomes in the list and create metagenomic datasets with only these genomes in order to train the model. The target model needs to be provided in a separate file.

Example 1

A genome list could be the following.

```
GCA_000148745.1
GCA_000521945.1
GCA_000492815.1
GCA_001444575.1
GCA_000007845.1
```

Which will include genomes from:

```
Pseudomonas aeruginosa 39016
Citrobacter freundii UCI 32
Klebsiella oxytoca MGH 42
Clostridium sporogenes
Bacillus anthracis str. Ames
```

A simple run of `fitModel.py` could be:

```
$>./fitModel.py -t genome.fa -sp 'Escherichia coli' -l genomes.txt
```

This will create a model for *Escherichia coli*, from the target `genome.fa` and the genome list `genomes.txt`.

1.2.4 Using random genomes from NCBI

If you don't have any preference regarding the genomes used to build the training set, imGLAD can built a model with a random number of genomes from NCBI. The user can provide the number of genomes to be used.

Example 2

```
$>./fitModel.py -t genome.fa -sp 'Escherichia coli' -s 100
```

This will create a model for *Escherichia coli*, from the target `genome.fa` using 100 randomly selected genomes from NCBI as background metagenomic signal.

1.2.5 Output files

```
parameters.txt
trainingGenomes.txt
detectionLimit.txt
```

After running `fitModel.py` 3 files will be created in your working directory. a `parameters.txt` file will contain the values for the regresssion parameters of the model, this file should be provided to `probEstimate.py` as an input under the option `-p`. Additionally a file `trainingGenomes.txt` is also created, this file will contain a list of the files used to create the training set. This should give you an idea of the complexity and diversity under which the model was created. Finally a `detectionLimit.txt` file, this file will include a calculation for the detection limit with a 95% confidence.

1.2.6 General use examples

Example 3

```
$>./fitModel.py -t genome.fa -sp 'Escherichia coli' -e 100
```

Specifying `-e` instructs `fitModel.py` to create a given number of training metagenomic datasets, the value specified with `-e` represents the number of positive and negative metagenomic datasets (with and withoureads from the target genome respectively). The default is to create 100 of each.

NOTE: A small numnber of training examples will take less computational resources (disk space and running time) but will likely yield inaccurate results. Perfect separation errors can occur as a consequence of small values of `-e`. This means that the positive and negative datasets are too far apart for imGLAD to establish a detection limit. If you get this error you should increase the training dataset size by increasing the value of `-e`

Example 4

```
$>./fitModel.py -t genome.fa -sp 'Escherichia coli' -j illumina
```

Specifying `-j` instructs `fitModel.py` to use a specific platform for read generation in this case Illumina. Although solid and Roche 454 technologies are also available.

Example 5

```
$>./fitModel.py -t genome.fa -sp 'Escherichia coli' -r 1000000
```

Specifying `-r` instructs `fitModel.py` to generate a particular number of reads in each training example. The default is 1 millon.

Example 6

```
$>./fitModel.py -t genome.fa -sp 'Escherichia coli' -d 150
```

Specifying `-d` designates the length of the reads in the training datasets.

1.3 Probability estimation

`probEstimate.py` estimates the likelihood of presence of your target genome in a metagenomic sample (one or more).

1.3.1 Getting started

First you need to run `fitModel.py` and get the parameters file it generatesi (`parameters.txt`).

Before running `probEstimate.py` you should also run a Blastn or Blat of your metagenomic sample against the target genome you created the model with. The output should be stored in tabular format.

1.3.2 General options

Required arguments

-t TARGET	Subject sequences (ref) in FastA format.
-m MAP	One or more Tabular BLAST files of reads vs genes (or contigs).
-p PARAM	Parameters file obtained from fitModel.py

1.3.3 Single mode

The `parameters.txt` file will save parameters for a model based on sequencing breath only and a model based o sequeincing breadth and depth. Single mode will use sequencing breadth parameters to estimate the likelihood of presence.

Example 1

```
$>./probEstimate.py -t genome.fa -m blastTab_01.tbl -p parameters.txt -l
single
```

The above line will estimate the likelihood of presence of the target genome in the metagenomic sample, based on sequencing breadth, the results will be displayed in the screen like.

Filename	Sequencing breadth	Sequencing depth	P-value
file01.tsv	float	float	float

You can save these results by redirecting the output to a file.

```
$>./probEstimate.py -t genome.fa -m blastTab_01.tbl -p parameters.txt -l
single > results.tsv
```

1.3.4 General mode

`fitModel.py` estimates parameters for the model based on sequencing depth as well. If general mode is used both variables will be used to calculate the precense probability.

Example 2

```
$>./probEstimate.py -t genome.fa -m blastTab_01.tbl -p parameters.txt -l
general
```

Will instruct `probEstimate.py` to include sequencing depth in the calculation of likelihood for presence.

NOTE: Using general mode can lead to false positive identification due to high mapping of conserved regions, plasmids, or mobile elements.