SigProfilerExtractor

Version

September 14, 2018

1 Introduction

The objective of the document is to introduce the users with the SigProfilerExtractor framework. SigProfilerExtrator is a python based package which generate the mutational signatures from mutation catalogues of cancer genomes [ref]. It is able to take input of the mutational catalogues from various input formats. Below we will discuss a step by step procedure which will direct the user to extract

2 Prerequisites

The framework requires access the to BASH along with following softwares:

☑Python ______ version 3.4 or newer
☑wget ______ version 1.19
☑Pandas (Python Library) ______ version (any)
☑Numpy (Python Library) ______ version (any)
☑Scikit-Learn (Python Library) ______ version (any)
☑Nimfa (Python Module) ______ version (1.3.4 or newer)

By default, the installation process will save the FASTA files for all chromosomes for the default genome assemblies (GRCh37, GRCh38, mm10). As a result, XXX Gb of storage must be available for the downloads.

3 Insatallation

After downloading the SigProfilerExtractor folder, users have to enter into the directory and type the following script in bash:

$bash installer.sh

That command will download the required data and make the framework to use. The download process will take few hours based on the internet speed.

4 Signature generation

The SigProfilerExtractor offers three different piplines to generate and analyze the mutational signatures from mutational catalogues of cancer genome. Three different python files are dedicated to excecute each of the pipelines. For each of the piplines, the results (the signatures and the statistics) will be generated in the output folder. We strongly recommend to empty the output folder before each run.

4.1 Pipeline one: Varient caller data to signature generation

This pipline starts from reading the VCF files to generate the mutational signatures. In the begining, the user need the create a project folder according the following steps:

1. Create a new folder for each project/job that you run within the SigProfilerMatrixGenerator/references/vcf_files/ folder. Use a unique name for each project/job.
2. Separate your INDEL mutations from your SNV mutations if they are present in the same files, and create a folder for each mutation type (ex: SigProfilerMatrixGenerator/references/vcf_files/[project]/SNV/ or SigProfilerMatrixGenerator/references/vcf_files/[project]/INDEL/)
3. Place your vcf files within these new folders (either SigProfilerMatrixGenerator/references/vcf_files/[project]/SNV or SigProfilerMatrixGenerator/references/vcf_files/[project]/INDEL/)

This pipeline currently supports maf, vcf, and simple text file formats as the variant caller data. The user must provide variant data adhering to one of these three formats. If the user's files are in vcf format, each sample must be saved as a separate file.

Once the project/job folder is successfully created, the user is ready the run the vcf_to_sigpro_results.py file within the source folder to generate the signatures. To learn the usage of arguments vcf_to_sigpro_results.py, please run the following command within the source folder:

$python3 vcf_to_sigpro_results.py -h

Example code:

$python3 vcf_to_sigpro_results.py project1 GRCh37 1 25 500

Where, "project1" is the name of the project folder that contains the variable calling data, GRCh37 is the reference genome, 1 is the minimum signature, 25 is the maximum signature and 500 is the number of iteratrions. All there arguments are required. Please check with the -h argument to see the optional arguments.

Once the excecution is finished successfully, the results should to generated in the output folder. For each context of mutations (6, 12, 96, 192, 1536, 3072, and DINUC), one file containing process data for different sets of signature, one file containing the exposure data for different sets of signature, one file containing the statistics data for different sets of signatures and one file as python object containing the data and statistics for each sets of signatures are generated.

4.2 Pipeline two: matlab data to signature generation

This pipeline starts from loading the mutational catalogue from a matlab object to generate the mutational signatures. To start the pipeline, first the user has to the place matlab object file in the input folder. Thereafter, the matobj_to_sigpro_results.py file has to excecuted. To learn the usage of arguments matobj_to_sigpro_results.py, please run the following command within the source folder:

$python3 matobj_to_sigpro_results.py -h

Example code:

$python3 matobj_to_sigpro_results.py 21_breast_WGS_substitutions.mat 1 25 500

Where, 21_breast_WGS_substitutions.mat is the name of the matlab object file that contains the mutational catalogue, 1 is the minimum signature, 25 is the maximum signature and 500 is the number of iteratrions. All there arguments are required. Please check with the -h argument to see the optional arguments.

Once the excecution is finished successfully, the results should to generated in the output folder. For that matlab object file, one file containing process data for different sets of signature, one file containing the exposure data for different sets of signature, one file containing the statistics data for different sets of signatures and one file as python object containing the data and statistics for each sets of signatures are generated.

4.3 Pipeline three: plain text mutational catalogue data to signature generation

This pipeline starts from loading the mutational catalogue from a matlab object to generate the mutational signatures. To start the pipeline, first the user has to the place matlab object file in the input folder. Thereafter, the text_to_sigpro_results.py file has to excecuted. To learn the usage of arguments text_to_sigpro_results.py, please run the following command within the source folder:

```
$python3 text_to_sigpro_results.py -h
```

Example code:

```
$python3 text_to_sigpro_results.py 21_breast_WGS_substitutions.txt 1 25 500
```

Where, 21_breast_WGS_substitutions.txt is the name of the matlab object file that contains the mutational catalogue, 1 is the minimum signature, 25 is the maximum signature and 500 is the number of iteratrions. All there arguments are required. Please check with the -h argument to see the optional arguments.

Once the excecution is finished successfully, the results should to generated in the output folder. For that matlab object file, one file containing process data for different sets of signature, one file containing the exposure data for different sets of signature, one file containing the statistics data for different sets of signatures and one file as python object containing the data and statistics for each sets of signatures are generated.

5 Copyright

6 Contact Information

Please contact S M Ashiqul Islam (Mishu) at m0islam@ucsd.edu to report any bug or any querie.