SigProfiler: Framework for Deciphering Mutational Signatures from Mutational Catalogues of Cancer Genomes

Version: 2.5.1
November 10, 2018

INTRODUCTION

The purpose of this document is to provide a brief and essential guide for using SigProfiler (formerly known as the Wellcome Trust Sanger Institute [WTSI]'s framework) for deciphering signatures of mutational processes from catalogues of cancer genomes. Detailed explanation of SigProfiler's theoretical model and computational framework is available in our manuscript entitled "Deciphering signatures of mutational processes operative in human cancer" by Alexandrov *et al.*, Cell Reports, Volume 3, Issue 1, 246-259:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3588146/

Further, several examples and multiple input files are provided to better illustrate how to apply SigProfiler to mutational catalogues of cancer genomes. Please note that SigProfiler allows identifying mutational signatures in both a direct manner and a hierarchical manner (see examples later in the document). The approach underlying the direct manner identification of mutational signatures can be found in the manuscript "Deciphering signatures of mutational processes operative in human cancer", while the approach underlying the hierarchical manner is briefly described in the manuscript "Landscape of somatic mutations in 560 breast cancer whole-genome sequences":

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4910866/

In addition to SigProfiler, a tool for assigning mutational signatures in single samples is also provided. The tool is known as SigProfilerSingleSample and its source code as well as an example for running it can be found in the *tools* folder. The algorithm underlying SigProfilerSingleSample is briefly described in our manuscript entitled "Clock-like mutational processes in human somatic cells" by Alexandrov *et al.*, Nature Genetics, Volume 47, Issue 12, 1402-1407:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4783858/

PREREQUISITES

SigProfiler and all additional tools are written in MATLAB and require the following packages with the specified (or newer) versions:
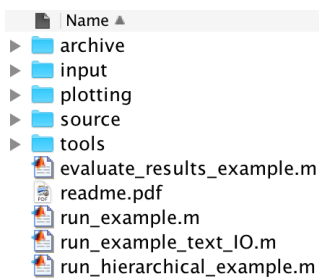
MATLAB	9.3.0.713579 (R2017b)
Parallel Computing Toolbox	Version 6.11 (R2017b)
Global Optimization Toolbox	Version 3.4.3 (R2017b)
Optimization Toolbox	Version 8.0 (R2017b)
Bioinformatics Toolbox	Version 4.9 (R2017b)
Statistics and Machine Learning Toolbox	Version 11.2 (R2017b)

In principle, SigProfiler will work on any operating system that allows installing MATLAB R2017b (or newer). Please note that SigProfiler has been tested with the above listed versions of MATLAB and MATLAB packages on macOS High Sierra (ver. 10.13.4). Further, please note that MATLAB, the parallel computing toolbox, the global optimization toolbox, and the optimization toolbox are essential for running SigProfiler's core functionality. In addition, MATLAB and these three toolboxes are also required for running SigProfilerSingleSample. The other two toolboxes are desirable but the code for deciphering mutational signatures and the code for assigning mutational signatures to an individual sample could be executed without them as other freely available packages have been leveraged in the appropriate places. SigProfiler and SigProfilerSingleSample do not require installation as one can run these programs directly through MATLAB.

Accurately deciphering signatures of mutational processes is computationally intensive. SigProfiler is usually executed on a computational cluster (or a computational farm) with at least 100 nodes. Further, the three provided examples assume that the default parallel cluster has already been preconfigured (please refer to MATLAB's documentation for configuring a default cluster). The code will make use of all available workers for the default cluster.

By default, SigProfiler uses the nonnegative matrix factorization (NMF) solver from (Brunet et al., PNAS, 2004, 12, 4164-4169), which is based on the multiplicative update algorithm (Lee and Seung, 1999, Nature 401, 788-791). However, if the Statistics and Machine Learning Toolbox is available, the provided NMF solver (i.e., nnmf) could be used instead and it will generally produce faster results. Additional freely available NMF solvers based on the multiplicative update algorithm as well as other algorithms are also provided. In principle, for sample datasets, all solvers (with the appropriate options) converge to almost identical solutions and the main difference between the algorithms is the required, for code execution, CPU time and memory.
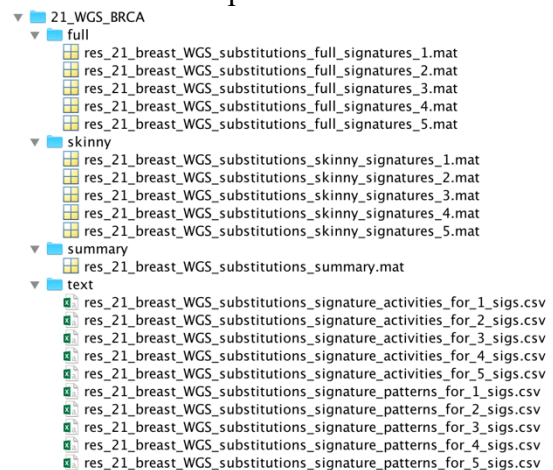
FOLDER STRUCTURE

Name ▲
▶ 📁 archive
▶ 📁 input
▶ 📁 plotting
▶ 📁 source
▶ 📁 tools
📄 evaluate_results_example.m
📄 readme.pdf
📄 run_example.m
📄 run_example_text_IO.m
📄 run_hierarchical_example.m

When first downloaded, SigProfiler contains five folders, four example files, and this readme file.

The source folder includes all code related to deciphering signatures of mutational processes including several nonnegative matrix factorization solvers. The plotting folder contains all source code related to plotting mutational signatures with and without strand bias as well as a plot that could be used for identifying the number of operative mutational signatures. The input folder contains examples of MATLAB (i.e., *.mat) and text input files. These files are used when executed the provided examples, *run_example.m* or *run_example_text_IO.m*. The executed example code will also create an output folder structure (shown on the right) and a temp folder. The output folder structure generates a folder for the analyzed dataset and *full*, *skinny*, *text*, and *summary* subfolders. The subfolders contain the results of executing SigProfiler in either a MATLAB (i.e., *.mat) or a text (i.e., *.csv) format. The result files in the *full* folder contain data for all iterations. These files usually require a lot of storage (i.e., 100+ gigabytes) when examining large datasets. The result files in the *skinny* folder contain data only for the average mutational signatures and the mutations they generate in individual samples. The result file in the *summary* folder provides summary information (i.e., average reconstruction,

```
▼ 📁 21_WGS_BRCA
  ▼ 📁 full
      res_21_breast_WGS_substitutions_full_signatures_1.mat
      res_21_breast_WGS_substitutions_full_signatures_2.mat
      res_21_breast_WGS_substitutions_full_signatures_3.mat
      res_21_breast_WGS_substitutions_full_signatures_4.mat
      res_21_breast_WGS_substitutions_full_signatures_5.mat
  ▼ 📁 skinny
      res_21_breast_WGS_substitutions_skinny_signatures_1.mat
      res_21_breast_WGS_substitutions_skinny_signatures_2.mat
      res_21_breast_WGS_substitutions_skinny_signatures_3.mat
      res_21_breast_WGS_substitutions_skinny_signatures_4.mat
      res_21_breast_WGS_substitutions_skinny_signatures_5.mat
  ▼ 📁 summary
      res_21_breast_WGS_substitutions_summary.mat
  ▼ 📁 text
      res_21_breast_WGS_substitutions_signature_activities_for_1_sigs.csv
      res_21_breast_WGS_substitutions_signature_activities_for_2_sigs.csv
      res_21_breast_WGS_substitutions_signature_activities_for_3_sigs.csv
      res_21_breast_WGS_substitutions_signature_activities_for_4_sigs.csv
      res_21_breast_WGS_substitutions_signature_activities_for_5_sigs.csv
      res_21_breast_WGS_substitutions_signature_patterns_for_1_sigs.csv
      res_21_breast_WGS_substitutions_signature_patterns_for_2_sigs.csv
      res_21_breast_WGS_substitutions_signature_patterns_for_3_sigs.csv
      res_21_breast_WGS_substitutions_signature_patterns_for_4_sigs.csv
      res_21_breast_WGS_substitutions_signature_patterns_for_5_sigs.csv
```

average stability, *etc.*) across the examined dataset. The result files in the *text* contain two types of text matrices (all in a CSV format) – one type of matrices containing the mutational signatures and another type containing the number of mutations attributed to each signature in each sample. The *temp* folder is used to store sub-matrix input files that are subsequently leveraged by the hierarchical example. Please also note that running a hierarchical analysis will generate sub-sub-folders for each layer of the analysis.

INPUT FILE FORMATS

A SigProfiler input file can be either in a MATLAB (i.e., *.mat) or in a text (i.e., CSV) format. A MATLAB input file contains a set of mutational catalogues and metadata information about the cancer type, and the mutational types, subtypes, etc. for which these mutational catalogues have been defined.

Name ▲	Value
{} cancerType	*1x1 cell*
▦ originalGenomes	*96x21 double*
{} sampleNames	*21x1 cell*
{} subtypes	*96x1 cell*
{} types	*96x1 cell*

For example, the provided **21_WTSI_BRCA_whole_genome_substitutions.mat** is shown in the figure above. The file contains the following fields:

- **cancerType:** a cell array of string(s) describing the type of each samples in the file.

- **sampleNames:** a cell array of strings in which each element corresponds to the name of the analyzed sample.

- **types:** a cell array of strings in which each element corresponds to the name of the mutational types for which the catalogues have been defined.

- **subtypes**: a cell array of strings in which each element corresponds to the name of the mutational subtype for which the catalogues have been defined. Note that additional fields could be added if more classes of mutational types are desired (e.g., strand bias).

- **originalGenomes:** an array containing mutational catalogues of cancer genomes with size <samples> by <mutational types> in which each element corresponds to the number of mutations per sample per mutational type and its subtype.

Please note that the MATLAB input file could contain more fields, however, the fields listed above are required for SigProfiler to examine the provided mutational catalogues.

In addition to a MATLAB input file, SigProfiler can be applied to a text file in a comma-separated values (CSV) format. Five examples of CSV input files are provided in the *input/text* folder. Each of these files is a matrix, where each column reflects a sample and each row a mutation type. The numbers in each cell of the matrix correspond to the number of mutations of a particular type found in a specific sample. In principle, the first few columns of the CSV file are used to define the names of the different mutation types in a string format. The five example input CSV files contain 35 whole-genome sequenced biliary adenocarcinomas for respectively: 96, 192, 1536, dinucleotide, and indel mutation types.

DESCRIPTION OF PROVIDED EXAMPLES

The provided examples perform 10 iterations per available core. Please note that there is an expectation of at least 1,000 iterations (*i.e.*, 100 available cores) and this number should be adjusted accordingly to the available nodes, otherwise it is possible that the identified mutational signatures are not accurate and/or not stable. Please note that only one of the examples generates plots, while the remaining three are used to generate output results in a MATLAB (*.mat files) and text format (*.csv). Lastly, note that the estimated time for running the provided examples is based on a MacBook Pro (13-inch, 2017) with

Intel Core i7 (3.5 GHz) and 16 GB of RAM (LPDDR3 2133 MHz). In principle, SigProfiler should be run on a large computational cluster.

run_example.m: This example illustrates deciphering mutational signatures in a direct manner from both: (i) a MATLAB input file containing a set of mutational catalogues derived from 21 breast cancer genomes; and (ii) a MATLAB input file containing a set of mutational catalogues derived from 100 breast cancer exomes with a third mutational subtype (*i.e.*, strand bias). **Estimated runtime:** 30 minutes.

run_example_text_IO.m: This example illustrates deciphering mutational signatures in a direct manner from a text/CSV file containing a set of mutational catalogues derived from 35 whole-genome sequenced biliary adenocarcinomas. **Estimated runtime:** 12 minutes.

run_hierarchical_example.m: This example illustrates deciphering mutational signatures in a hierarchical manner. The example requires first executing *run_example.m* and identifying the optimal number of mutational signatures in the dataset. The parameters of the example are set for identifying a second layer of mutational signatures in 21 breast cancer genomes. Provided that there are enough samples, the example can be easily modified for analyzing deeper hierarchical layers by only changing the parameters. **Estimated runtime:** 18 seconds.

evaluate_results_example.m: This example illustrates identifying the number of mutational processes operative in an examined dataset. The default option show results for the set of 21 breast cancer genomes, while simply changing the file names will allow examining the results for the 100 breast cancer exomes or the 35 whole-genome sequenced biliary adenocarcinomas. This example requires running *run_example.m* or *run_example_text_IO.m*. **Estimated runtime:** 9 seconds.

CONTACT INFORMATION
Please address any queries or bug reports to Ludmil B. Alexandrov at l2alexandrov@ucsd.edu.