mutationSeq_3.0.0

Table of Contents

1. About

mutationSeq is software for somatic SNV detection using next generation sequencing (NGS) data from matched tumour/normal samples. It uses a feature-based classifier trained on validated somatic mutation samples while benefiting from other available information such as base quality, mapping quality, strand bias and tail distance.

2. Download

The package can be downloaded from Shah lab for Computational Cancer Biology homepage
It is available to academics under the terms of the GPL3. Non-academic users should contact Dr. Patrick Rebstein at the BC Cancer Agency Technology Development Office for license agreements.

3. Installation

This package is only supported on linux operating systems.
To install the package, simply extract the downloaded `mutationSeq_3.0.0.tar.gz` file into the desired directory. Using a terminal in linux:

```
cd <$desired_dir>
tar -xvf mutationSeq_3.0.0.tar.gz
```

Note that the `setup.py` provided in the package is for compiling the pybam library (bam/fasta traversal engine) and should not be used for the purpose of installation.

4. Dependencies

`mutationSeq_3.0.0` has the following dependencies:

- python (tested for version 2.7.x) with the following packages installed:
  - numpy (tested for version 1.7.1 and highly recommended to link against BLAS)

    - scipy (tested for version 0.12.0)
    - scikits-learn (tested for version 0.13.1)
- samtools (tested for version 1.2.3+)

Install LAPACK package and make sure to export `LD_LIBRARY_PATH` to the location of the LAPACK/ATALS libraries.

5. Usage

To use the mutationSeq_3.0.0 package, there are two python wrappers around the pybam engine:

1. `train.py`
2. `classify.py`

5.1. Train

`train.py` aims to extract features of the labeled training data and fit them to a model based on the random forest algorithm to avoid over fitting. It is also robust against training sets with unequal distributions of positive and negative samples and supports multi-class classification. Figure 1 shows the schematic of `train.py` implementation.

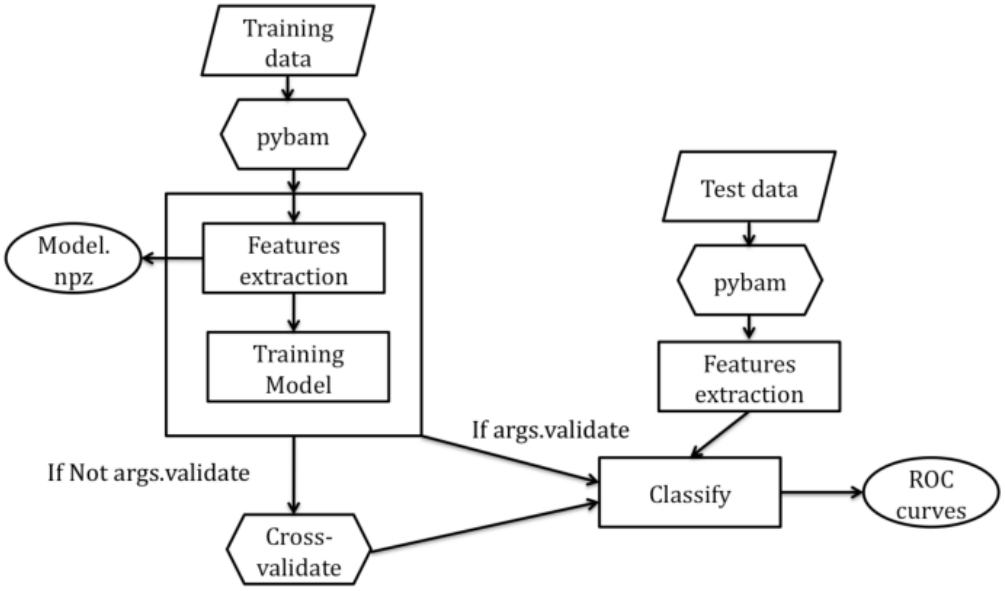

Figure 1. `train.py` implementation

5.2. Classify

Given a matched tumour/normal bam files, **`classify.py`** returns the probability that each candidate site is somatic. It employs the model generated by `train.py`. Figure 2 shows the schematic of `classify.py` implementation.

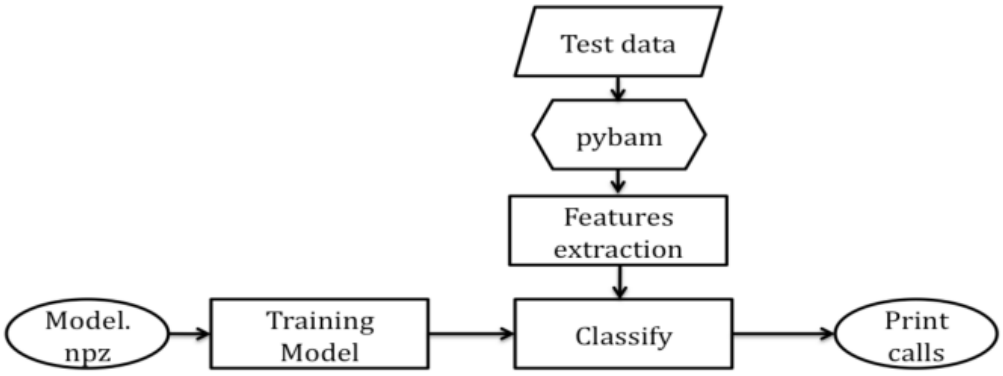

Figure 2. `classify.py` implementation

6. How To Run

The easiest way to run **mutationSeq_3.0.0** to analyse a whole genome or exon capture is to run `classify.py` with the default `model.npz` provided in the package. It is also possible to build a custom model based on new training data by running `train.py` first.

6.1. Running `classify.py`

Using a terminal in linux:

```
cd <$mutationSeq_3.0.0>
python classify.py normal:normal.bam tumour:tumour.bam reference:reference.fasta model:model.npz
[--options]
```

6.1.1. Description

`classify.py` should be run using a python command. It is recommended to use python 2.7 (since python 3's new features have not yet been tested on this version of mutationSeq). A list of colon-delimited sample names should be specified as input arguments as following:

`normal:`	`/path/to/normal_file.bam`
`tumour:`	`/path/to/tumour_file.bam`
`reference:`	`/path/to/reference.fasta`
`model:`	`/path/to/model.npz` (use the `model.npz` provided in the package unless you train a new mode using `train.py`)

The following options are available:

`-h --help`	print usage help - optional
`--config`	specify the `path/to/metadata.config` that is used to add meta information to the output file as in the standard VCF4.1 - mandatory
`--export`	save exported feature vector of test data, `default=None` - optional
`--interval`	specify `[chr]:[start]-[end]` range, `default=None` (will analyse whole genome) - optional
`--normalized`	use this option if using `Nmodel.npz`, otherwise it will extract regular features which match `model.npz`, `default=False`, `action="stotr_true` - optional
`--out`	`/path/to/output.vcf`, `default=None` - mandatory
`--purity`	pass sample purity as an input, `default=70` - optional
`--threshold`	filter out the predicted probabilities greater than or equal to `--threshold`, `default=0.5` - optional
`--verbose`	`default=False`, `action="store_true"` (still under modification) - optional
`--version`	print version of the software - optional

Note that some of these options are mandatory.

6.1.2. Example

This is a typical example that would run `classify.py` on the specified tumour/normal pair bam files using the `human_all.fasta` reference. It would be run for chromosome 1 for positions in the range `[123456, 345678]` and the results are written as `out.vcf` in the specified directory. Note that it would print only those positions whose predicted probabilities are greater than `0.3`.

```
cd <$mutationSeq_3.0.0>
python classify.py normal:/share/…/DAH145N_A12970_3_lanes_dupsFlagged.bam
tumour:/share/…/DAH145_A12958_3_lanes_dupFlagged.bam reference:/share/…/reference/human_all.fasta
model:/share/…/mutationSeq_3.0.0/model.npz --config /share/…/mutationSeq_3.0.0/metadata.config
--interval 1:123456-345678 --threshold 0.3 --out /share/…/out.vcf
```

The *fasta* and *bam* files used as input arguments need to be pre-indexed by samtools.

Use `metadata.config` provided in the package for `--config` option.

6.1.3. Output

The output of `classify.py` is formatted as standard VCF4.1. It consists of two parts: meta information and data lines. Below is a sample meta information from a `classify.py` output file:

```
##fileformat=VCFv4.1
##fileDate=$date of the file
##source=mutationSeq-3.0.0
##reference= $the reference used in the command line
##tumour=$the tumour file used in the command line
##normal=$the normal file used in the command line
##threshold= $ the value specified as threshold in the command line
##INFO=<ID=PR,Number=1,Type=Float,Description="Probability of somatic mutation">
##INFO=<ID=TR,Number=1,Type=String,Description="Count of tumour with reference to REF">
##INFO=<ID=TA,Number=1,Type=String,Description="Count of tumour with reference to ALT">
##INFO=<ID=NR,Number=1,Type=String,Description="Count of normal with reference to REF">
##INFO=<ID=NA,Number=1,Type=String,Description="Count of normal with reference to ALT">
##INFO=<ID=TC,Number=1,Type=String,Description="Tri-nucleotide context">
##FILTER=<ID=threshold,Description="Threshold on probability of positive call">
```

The meta information header is self explanatory. To read more about VCF4.1 please refer to VCF4.1 description.

The data line section of the output looks like the following:

```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO
1       249      .            A         G       0.60          PASS
PR=0.129;TR=62;TA=4;NR=112;NA=0;TC=AAA
1       268      .            A         C       0.65          PASS
PR=0.139;TR=55;TA=5;NR=96;NA=0;TC=TAA
1       274      .            A         G       1.81          PASS
PR=0.341;TR=58;TA=3;NR=95;NA=0;TC=TAA
1       280      .            A         C       1.19          PASS
PR=0.239;TR=52;TA=4;NR=88;NA=1;TC=TAA
```

where `REF` is the reference nucleotide at the position `POS` of chromosome `CHROM`. `ALT` is the nucleotide on the major allele in tumour if different from `REF`, otherwise it is the nucleotide on the minor allele. `QUAL` is Phred_quality score and `FILTER` is `PASS` if the probability of being somatic is greater than `--threshold`. `ID` would be eventually a unique id but not yet implemented.
`INFO` column contains the following information:

- `PR`: predicted probability of being somatic mutation
- `TR`: number of nucleotides in the tumour bam file in the position `POS` that match to the reference nucleotide reported by `REF`
- `TA`: similar to `TR` but matches to the alternative nucleotide reported by `ALT`
- `NR`: similar to `TR` but for the normal bam file
- `NA`: similar to `NR` but matches to the alternative nucleotide reported by `ALT`
- `TC`: Tri-nucleotide context

6.2. Running `train.py`

Using a terminal in linux:

```
cd <$mutationSeq_3.0.0>
python train.py posfile.pos [--options]
```

6.2.1. Description

`train.py` aims to generate a new model using new training data. A position input file, e.g. `posfile.pos`, is required. This file consists of a list of space-delimited columns. The columns are: **chromosome**, **position**, and **label**. The file also contains a header which specifies the following information:

normal	`/path/to/normal_file.bam`
tumour	`/path/to/tumour_file.bam`
reference	`/path/to/reference.fasta`

Figure 3 shows a sample position input file.

```
# normal /share/.../SA216N_HS2208_2_lanes_dupsFlagged.bam
# tumour /share/.../SA216_HS2198_3_lanes_dupsFlagged.bam
# reference /share/lustre/reference/genomes/human_all.fasta
10 89682885 SOMATIC
21 46676181 GERMLINE
17 19127560 WILDTYPE
16 79799603 GERMLINE
1 172684077 SOMATIC
```

Figure 3. sample position input file for running `train.py`

The following options are available:

`-h --help`	print usage help - optional
`--label`	specify the label in training file list, `default="SOMATIC"` - optional
`--normalized`	specify if you want to train with normalized features, `default=False` - optional
`--out`	`/path/to/output.vcf`, `default=None` - mandatory
`--validate`	activate the option of validating the same format file with known labels, `metavar='FILE'`, `nargs='*'`, `default=None` - optional
`--version`	print version of the software - optional

⚠ Note that `--out` option is mandatory.

6.2.2. Example

This is a typical example that would run `train.py` on the specified `posfile.pos`. Likely, a training set will be a list of isolated positions and truth labels across different cases. `train.py` extracts the features for these positions and fit them to a model. Train.py will accept a glob of position files (`*.pos`) that have chromosome, position, and somatic status. For each of these files it expects a header with the normal, tumour, and reference paths.

```
cd <$mutationSeq_3.0.0>
python train.py /share/…/posfile.pos --out /share/…/sample_model
```

6.2.3. Output

`sample_model.npz` will be the main output of train.py saved in the directory specified by `--out` option. This model can be later used for classification using `classify.py`. The cross-validation/ validation results and ROC curves plots are also saved in the specified directory.

7. Reference

If you use this software in your work, please cite the following publication:

- Ding et al. Feature-based classifiers for somatic mutation detection in tumour-normal paired sequencing data. Bioinformatics. 2012 Jan 15;28(2):167-75. doi: 10.1093/bioinformatics/btr629.

8. Contact

For further information please refer to Shah lab for Computational Cancer Biology homepage or contact:

- Jafar Taghiyar <jtaghiyar@bccrc>
- Dr. Sohrab Shah <sshah@bccrc.ca>