MutEnricher: Quick Start Guide

Anthony R. Soltis

June 26, 2018

1. Introduction

MutEnricher is a tool suite that performs somatic mutation recurrence analysis from tumor whole genome sequencing data. It runs in two distinct fashions: 1) *coding* for examination of non-silent somatic mutations in protein coding genes and 2) *noncoding* for enrichment analysis of somatic mutations in non-coding genomic regions. Both analyses are run from user-supplied somatic VCF files with additional input requirements and options.

*****More detailed explanations of run options and outputs are supplied in the accompanying tutorial.*****

2. Downloads, requirements, and installation

The basic steps for downloading and installing MutEnricher are:

1. Obtain MutEnricher code from GitHub:

   ```
   git clone http://github.com/asoltis/MutEnricher.git
   ```

   or

   Directly download repository and unzip file contents.

2. Install the required Python packages (if not already available on your system):

   1. NumPy (http://www.numpy.org/)
   2. SciPy (https://scipy.org/)
   3. Cython (http://docs.cython.org/en/latest/src/quickstart/install.html)
   4. cyvcf2 (http://brentp.github.io/cyvcf2/)
   5. pysam (http://pysam.readthedocs.io/en/latest/)

   The easiest way to obtain these additional packages is to use the Python package manager pip. Python versions >= 2.7.9 include pip by default; otherwise, this package can be installed separately (https://pip.pypa.io/en/stable/installing/). We have explicitly tested our code with Python versions 2.7.12 and 2.7.13.

3. Cythonize the included mathematical functions:

   1. Enter the `math_funcs` sub-directory:
      ```
      cd math_funcs
      ```
   2. Run the following command to cythonize the code:
      ```
      python setup.py build_ext --inplace
      ```

These steps will produce the file `math_funcs.so` in the `math_funcs` directory.

The package should now be successfully installed on your system. The main package driver script is **`mutEnricher.py`**. To see the main help page, use **`python mutEnricher.py -h`**. To see the detailed help pages for the two analysis types, use **`python mutEnricher.py <command> -h`**, where `<command>` is either **`coding`** or **`noncoding`**.

3. Run MutEnricher on example data

We include **example data** for test execution in the `example_data` sub-directory. The sub-directories within are:

1. `annotation_files` – includes an example GTF file and BED file of coding gene promoters.
2. `covariates` – includes example covariate and covariate weights files (for use with covariate clustering options).
3. `precomputed_apcluster` – includes pre-computed affinity propagation clustering results for covariates associated with the provided annotation files.
4. `vcfs` – 100 synthetic somatic mutation files (with tabix-indexed .tbi files)

Additional files include:

- `nonsilent_terms.txt`
- `vcf_files.txt`
- **`quickstart_commands.txt`**
  - This file lists the quick start commands described in this tutorial.

The example somatic VCF files were generated by randomly assigning "somatic mutations" throughout the genome at a global frequency of 2 mutations per megabase, and thus **DO NOT** represent true whole genome somatic mutation data. Included with these randomly distributed "mutations" are known cancer mutations in the coding regions of the genes KRAS and TP53 (coding examples) and in the promoter of the TERT gene (non-coding example). These mutations were included at target cohort frequencies of 30%, 90%, and 40%, respectively, and serve as known true positives for testing.

For the below example runs, the working directory is assumed to be the `example_data` sub-direcotry within the main install directory.

3.1 Quick start <u>coding</u> analysis example runs:

NOTE: unzip example GTF file if not already done!

1. ***Basic run using default "global" background mutation frequency calculation method:***

```
python ../mutEnricher.py coding
annotation_files/ucsc.refFlat.20170829.no_chrMY.gtf vcf_files.txt --anno-type
nonsilent_terms.txt -o test_out_coding --prefix test_global
```

If this executes successfully, you should see three output files in the specified output directory with the supplied prefix. Examine the first 10 lines of "<prefix>_gene_enrichments.txt" file:

```
head test_global_gene_enrichments.txt | cut -f1,3,10
```

```
Gene    num_nonsilent   FDR_BH
TP53    93      8.32e-200
KRAS    29      2.51e-54
MYOF    7       0.281
RPS24   3       0.519
SPG21   3       0.519
CTSH    3       0.519
SSUH2   3       0.519
GJC2    3       0.519
PCDHGA1 4       0.519
```

You can see the genes *TP53* and *KRAS* at the top of the list with very significant associated FDR-corrected p-values. These are expected from the example files.

2. ***Basic run using "local" background mutation frequency calculation method with parallelization:***

```
python ../mutEnricher.py coding
annotation_files/ucsc.refFlat.20170829.no_chrMY.gtf vcf_files.txt --anno-type
nonsilent_terms.txt -o test_out_coding --prefix test_local --use-local -p 5
```

Checking the output from this run is very similar to (1) above, but with different final numbers due to the change in background calculation method (can compare column 8 between outputs):

```
head test_local_gene_enrichments.txt | cut -f1,3,8,10

Gene    num_nonsilent   bg_prob FDR_BH
TP53    93      5.53e-05        6.26e-66
KRAS    29      1.09e-05        2.25e-32
GJC2    3       4.12e-06        1
SPG21   3       6.17e-06        1
NAT8    2       3.52e-06        1
RGPD5   8       2.14e-06        1
EMCN    2       3.35e-06        1
NTM     2       2.11e-06        1
OR13C5  2       3e-06   1
```

3. ***Run with covariate clustering enabled (through pre-computed clusters)***

For the below run, use pre-computed gene covariate clustering results for the background frequency calculation method. We provide two pre-computed sets of clusters for coding genes contained in the supplied GTF annotation file: *all genes* (i.e. clustering performed on all ~19,000 genes contained) and *by contig* (i.e. clustering performed for all genes on each individual chromosome). **NOTE**: the affinity propagation code *does not* need to be available to run with pre-computed clusters.

To execute with pre-computed covariates, use the `--precomputed-covars` option, which accepts the path to the clustering results directory:

```
python ../mutEnricher.py coding
annotation_files/ucsc.refFlat.20170829.no_chrMY.gtf vcf_files.txt
--precomputed-covars
precomputed_apcluster/coding.ucsc.refFlat.20170829.no_chrMY/all_genes/apcluster_ge
```

```
nes
--anno-type nonsilent_terms.txt
-o test_out_coding
--prefix test_precompAP_all_genes
-p 5
```

Anticipated output:

```
head test_precompAP_all_genes_gene_enrichments.txt | cut -f1,3,8,10

Gene    num_nonsilent   bg_prob FDR_BH
TP53    93      2.21e-06        4.33e-193
KRAS    29      2.32e-06        1.12e-51
MYOF    7       1.96e-06        0.38
RGPD5   8       1.05e-06        0.393
RGPD6   6       1.06e-06        0.653
RPS24   3       2.31e-06        0.653
SPG21   3       2.43e-06        0.653
TMEM39A 3       1.63e-06        0.653
CTSH    3       2.55e-06        0.653
```

Again, the expected genes appear as highly significant.

4. ***Run with covariate clustering enabled (enable affinity propagation)***

To execute the analysis with covariate clustering enabled, use the `-c` option with the appropriate covariates table for the genes in the input GTF. Optionally, the user can supply covariate weights with the `-w` option. **NOTE:** *the below example performs covariate clustering by contig (i.e. by chrmosome) using the –`by-contig`; this option speeds the clustering calculations by performing multiple affinity propagation runs for genes on the same chromosome only. If this option is not set, the clustering analysis will be performed on all genes (~19,000 protein coding in the supplied GTF). While this latter option is indeed valid, the run time for this is considerably longer due to the large number of data points. Practically, if the user wishes to use this latter option, he or she may wish to restrict the number of genes considered (with the* `-g` *option) or pre-compute the clusters once and re-use in subsequent runs on the same or other data.*

```
python ../mutEnricher.py coding
annotation_files/ucsc.refFlat.20170829.no_chrMY.gtf vcf_files.txt
-c covariates/ucsc.refFlat.20170829.no_chrMY.covariates.txt
-w covariates/ucsc.refFlat.20170829.no_chrMY.covariate_weights.txt
--by-contig
--anno-type nonsilent_terms.txt
-o test_out_coding
--prefix test_APclust_byContig
-p 10
```

The output from the above execution is:

```
head test_APclust_byContig_gene_enrichments.txt | cut -f1,3,8,10

Gene    num_nonsilent   bg_prob FDR_BH
TP53    93      2.73e-06        1.28e-184
```

```
KRAS    29      2.53e-06        1.46e-50
MYOF    7       2.33e-06        0.796
CTSH    3       2.27e-06        0.796
SSUH2   3       2.02e-06        0.796
MBTPS1  4       1.83e-06        0.796
RPS24   3       3.36e-06        0.796
PALMD   3       2e-06   0.796
TMEM39A 3       2.36e-06        0.796
```

3.2 Quick start non-coding analysis example runs:

The general run scheme for non-coding analysis is similar to what was presented for coding. A major difference is the requirement of an input **BED** file with non-coding regions of interest (as opposed to a GTF of gene models for coding analysis). In addition, as the non-coding analysis does not require definitions for silent versus non-silent mutations, no flag or input file is required for such definitions here.

1. ***Basic run using default "global" background mutation frequency calculation method***

```
python ../mutEnricher.py noncoding
annotation_files/ucsc.refFlat.20170829.promoters_up1kb_down200.no_chrMY.bed
vcf_files.txt -o test_out_noncoding --prefix test_global -p 5
```

Example output of overall regional analysis:

```
head test_global_region_WAP_enrichments.txt | cut -f2,3,12

region_name     num_mutations   FDR_BH
TP53_2  23      0
TERT    43      0
ZSCAN5A_2,ZSCAN5A_5,ZSCAN5A_1,ZSCAN5A_3 3       0.012
CYP4X1_2        3       0.309
CXCR5_2 3       0.309
GBP1    4       0.309
KLHL3_1 4       0.309
GPR137_2,GPR137_1,GPR137_3,BAD_1,BAD_2,GPR137_4 4       0.309
HIF3A_2 3       0.309
```

Note the *TERT* promoter enrichment is expected (and is highly significant). The *TP53* result in this case is due to an overlapping definition of a *TP53* exon with a definition for its promoter (due to varying gene models in the GTF).

Sample output from "hotspot" analysis:

```
head test_global_hotspot.txt | cut -f1,3,10

Hotpsot region_name     BH_qval
chr17:7578555-7578555   TP53_2  1.83e-108
chr5:1295773-1296014    TERT    1.31e-29
chr5:1295047-1295288    TERT    1.31e-29
chr5:1295461-1295611    TERT    9.4e-15
chr5:1295688-1295698    TERT    2.16e-09
chr19:56827049-56827058 ZSCAN5A_2,ZSCAN5A_5,ZSCAN5A_1,ZSCAN5A_3 2.16e-09
```

```
chr5:1294942-1294970    TERT    4.15e-08
```

Note, again, some significant sub-regions ("hotspots") within the *TERT* promoter that show up as significant.

2. ***Basic run using "local" background mutation frequency calculation method with parallelization***

```
python ../mutEnricher.py noncoding
annotation_files/ucsc.refFlat.20170829.promoters_up1kb_down200.no_chrMY.bed
vcf_files.txt
-o test_out_noncoding
--use-local
--prefix test_local
-p 5
```

3. ***Run with covariate clustering enabled (through pre-computed clusters)***

```
python ../mutEnricher.py noncoding
annotation_files/ucsc.refFlat.20170829.promoters_up1kb_down200.no_chrMY.bed
vcf_files.txt
-o test_out_noncoding
--precomputed-covars
precomputed_apcluster/noncoding.ucsc.refFlat.20170829.promoters_up1kb_down200.no_c
hrMY/apcluster_regions
--prefix test_precompAP
-p 5
```

4. ***Run with covariate clustering enabled (enable affinity propagation)***

A subtle differences when running non-coding analysis with covariate clustering:

- The default behavior is to split the analysis by chromosome. This is generally useful, if not necessary, as the potential space of non-coding regions can be much greater than that of coding genes, depending on how the user defines such features.

```
python ../mutEnricher.py noncoding
annotation_files/ucsc.refFlat.20170829.promoters_up1kb_down200.no_chrMY.bed
vcf_files.txt
-o test_out_noncoding
-c
covariates/ucsc.refFlat.20170829.promoters_up1kb_down200.no_chrMY.covariates.txt
-w
covariates/ucsc.refFlat.20170829.promoters_up1kb_down200.no_chrMY.covariate_weight
s.txt
--prefix test_APclust
-p 10
```

3.3. Coding analysis form MAF input

MutEnricher can also perform coding analysis from Mutation Annotation Format (MAF) input files (se specification at https://wiki.nci.nih.gov/display/TCGA/Mutation+Annotation+Format+(MAF)+Specification). For example, the Broad Institute's GDAC Firehose database contains a wide array of somatic mutation data from various cancer studies in MAF format (see https://gdac.broadinstitute.org/).

Nearly all coding analysis options available for VCF inputs are also available for MAF; however, the *local* background method is not available as this requires the ability to scan neighboring regions in and outside of the regions of interest – these often are not included in MAF files.

An example run on a MAF using pre-computed covariate clustering results (e.g. using somatic mutations from GDAC Firehose for a glioblastoma [GBM] study [http://gdac.broadinstitute.org/runs/analyses__2016_01_28/reports/cancer/GBM/MutSigNozzleReportCV/GBM-TP.final_analysis_set.maf]) is:

```
python ../mutEnricher.py coding
annotation_files/ucsc.refFlat.20170829.no_chrMY.gtf -
--maf GBM-TP.final_analysis_set.maf
--exome-only
--min-clust-size 0
--precomputed-covars
precomputed_apcluster/coding.ucsc.refFlat.20170829.no_chrMY/all_genes/apcluster_ge
nes
-o test_out_maf
--prefix GBM
```

A few notes about this run:

- The MAF input is supplied with the `--maf` option.
- Instead of the "vcfs_list.txt" input, a place-holder character (e.g. "-") is used. With the `--maf` option set this input is ignored, but a value is required for run.
- The `--exome-only` flag is set. This tells the program to only cosider *exonic* regions (as opposed to exonic *plus intronic* in normal analyses). This is useful when the input data derives from non whole genome sequencing data (e.g. whole exome sequencing).
- Setting `--min-clust-size` to 0 is appropriate when using covariate clustering data as anything >0 can require local analysis, which is not available with MAF input.

During run, you may receive warnings similar to: "`KLRC3 var chr12 10588555_C_C not in gene limits.`" This generally results from mismatches between the gene coordinates described in the input GTF and those defined in the MAF. MutEnricher simply skips such variants. In addition, gene IDs present in the MAF that are not recognized in the input GTF are skipped and written to an output file called `unrecognized_genes.txt`.

Example output from the above run:

```
head -n20 GBM_gene_enrichments.txt | cut -f 1,3,10

Gene    num_nonsilent   FDR_BH
TP53    97      2.26e-98
PTEN    89      3.05e-82
EGFR    92      5.9e-80
PIK3R1  33      2.1e-17
PIK3CA  31      5.94e-14
NF1     35      3.34e-09
RB1     25      1.52e-07
SPTA1   26      0.00175
IDH1    14      0.00189
```

KEL	15	0.0034
HCN1	12	0.00352
ABCC9	14	0.0251
STAG2	12	0.0363
SEMA3C	11	0.0441
RYR2	30	0.151
CALCR	8	0.151
PDGFRA	13	0.259
SEMA3E	8	0.268
GABRA6	11	0.268