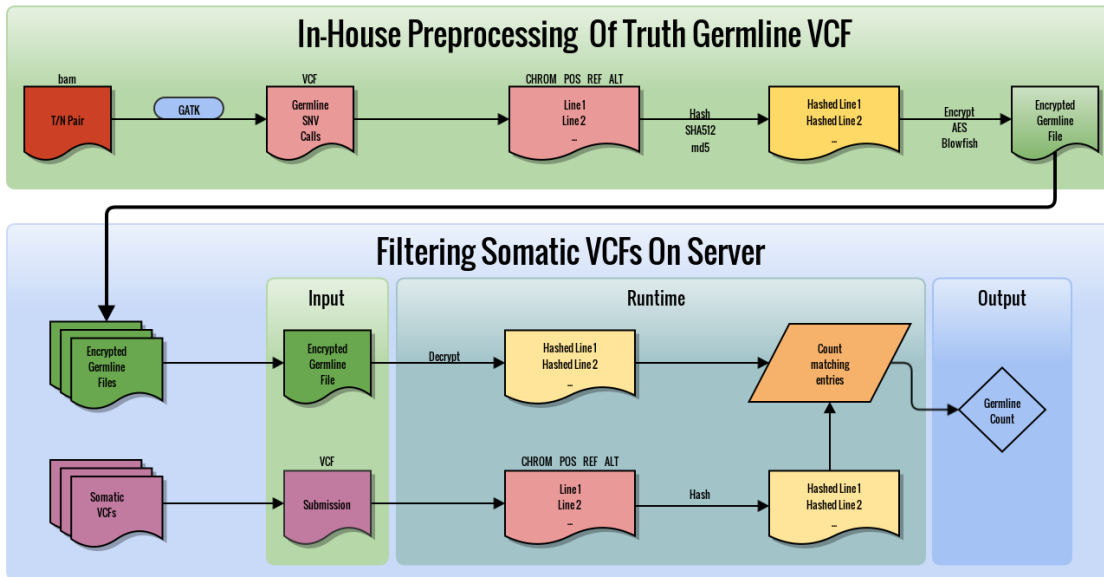


GermlineFilter User Manual

GermlineFilter v1.2 User Manual

Germline Filter Workflow



Overview

Germline Filter is a Python program used in the [ICGC-TCGA DREAM Mutation Calling Challenge](#) meant to contribute to the real data security measures. It takes as input a *preprocessed and encrypted* germline calls file, as generated by GATK, and a somatic SNV vcf file, and it returns a *Pass/Fail* value. It is integrated with the Synapse submission infrastructure, and it is ran against each VCF submitted for the real data SNV subchallenges. If in a submission there are leaked germline calls more than a fixed threshold, that submission is flagged and automatically made private.

Note: the public version is returning the actual *number* of germline calls found in the somatic vcf.

Features

- The most important feature of the GermlineFilter is that the program runs in an encrypted fashion, making it safe for running on any server. All the *filtering steps* described in the flowchart are done at runtime, and at no point data is written on the disk. It has two options:
 - `encrypt_germline_vcf` - Encrypt a truth germline vcf (preprocessing steps in the workflow above)
 - `filter` - Filter a somatic vcf against an encrypted truth germline vcf. This step is done in an encrypted fashion.
 - `get_germline_positions` - Get the actual germline positions called in a somatic vcf. This step is done in an *unencrypted* fashion, against the original truth germline vcf. It should only be run locally or on an encrypted server. The output is written to a tab delimited file.
- Multiple germline vcf's can be preprocessed at the same time, with a common salt file and key file.
- Multiple somatic vcf's corresponding to the same encrypted truth germline file can be filtered simultaneously. This considerably increases the speed versus individual runs.
- The user can choose the encryption protocol (AES, Blowfish); default AES
- The user can choose the hashing protocol (md5 or sha512); default sha512
- Get the actual germline position in a vcf, for plotting and further analysis.

Dependencies

GermlineFilter requires `pycrypto` module. You can install it on your local machine from source code, or if you have `pip` installed, via `pip install pycrypto`. On the cluster, `pycrypto` is already installed and it is available via module `load pycrypto`.

Getting the source code

Soon, the code will be on [PyPI](#). If you have `pip` installed, you would be able to get the Germline Filter via:

```
pip install GermlineFilter
```

Alternatively, a tarball `GermlineFilter-1.2.tar.gz` will be available for download. GermlineFilter can be installed from source via:

Install from source

```
tar -zxvf GermlineFilter-1.2.tar.gz
cd GermlineFilter-1.2
python setup.py build
python setup.py install
```

Running `germline_filter`

To get familiar with the command line arguments needed and their description run the following commands:

Install Germline Filter locally

```
germline_filter --help
germline_filter encrypt_germline_vcf --help
germline_filter filter --help
germline_filter get_germline_positions --help
```

Examples

Encrypted Germline Filtering

Step 1: Encrypt a truth germline vcf

Encrypt a truth SNV germline vcf

```
germline_filter encrypt_germline_vcf --germline <mygermline.vcf> --outdir
<myoutputdir> --saltfile <mygermline.salt> --keyfile <mygermline.key> --encryption
{AES,Blowfish} --hashing {md5,sha512}
```

This creates three (or more - if multiple germline files are encrypted simultaneously) files in `<myoutputdir>`:

```
processed_<encryption>_<hashing>_<mygermline>.vcf.enc
```

```
<mygermline>.key
```

```
<mygermline>.salt
```

Step 2: Filter a somatic SNV vcf file

Filter somatic SNV vcf file

```
germline_filter filter --somatic <mysomatic_1.vcf> <mysomatic_2.vcf> ...  
<mysomatic_n.vcf> --germline  
processed_<myencryptionprotocol>_<myhashingprotocol>_<mygermline>.vcf.enc --saltfile  
<mygermline>.salt --keyfile <mygermline>.key --encryption {AES,Blowfish} --hashing  
{md5,sha512}
```

Remember that if multiple somatic vcf are checked simultaneously, they *must* correspond to the same truth germline vcf.

This will print to stdout the following:

```
somatic.vcf    germline.count  
  
<mysomatic_1.vcf>    <x>  
  
<mysomatic_2.vcf>    <y>  
  
...  
  
<mysomatic_n.vcf>    <z>
```

Get Germline Positions In Somatic vcf Files

Get Germline Positions

```
germline_filter get_germline_positions --truth <mytruthgermline>.vcf --vcfs  
<mysomatic_1>.vcf <mysomatic_2>.vcf ... <mysomatic_n>.vcf --outdir <myoutdir> --sample  
<mysamplename>
```

For every somatic vcf in the command line arguments, the output will be written in <myoutdir>/<mysamplename>/<mysomatic_i>_germline_calls.tsv and it will be of the form:

```
chrom    pos  
8        12416683  
17       21565058  
1        149028657  
11       79197184  
21       45970303  
20       25919675
```

Issues and bugs

Note that the code is still under development, and it still has many limitations. However, if you encounter any issues when trying to access/install/run the code for the above functionality, or if you find any bugs, please don't hesitate to contact me at cristian.caloian@oicr.on.ca or cristian.caloian@gmail.com.