MoCha Oncogenic MOI Annotator (MOMA) Documentation

Release v1.3.20200310-dev1

Dave Sims

Mar 27, 2020

CONTENTS

CHAPTER

ONE

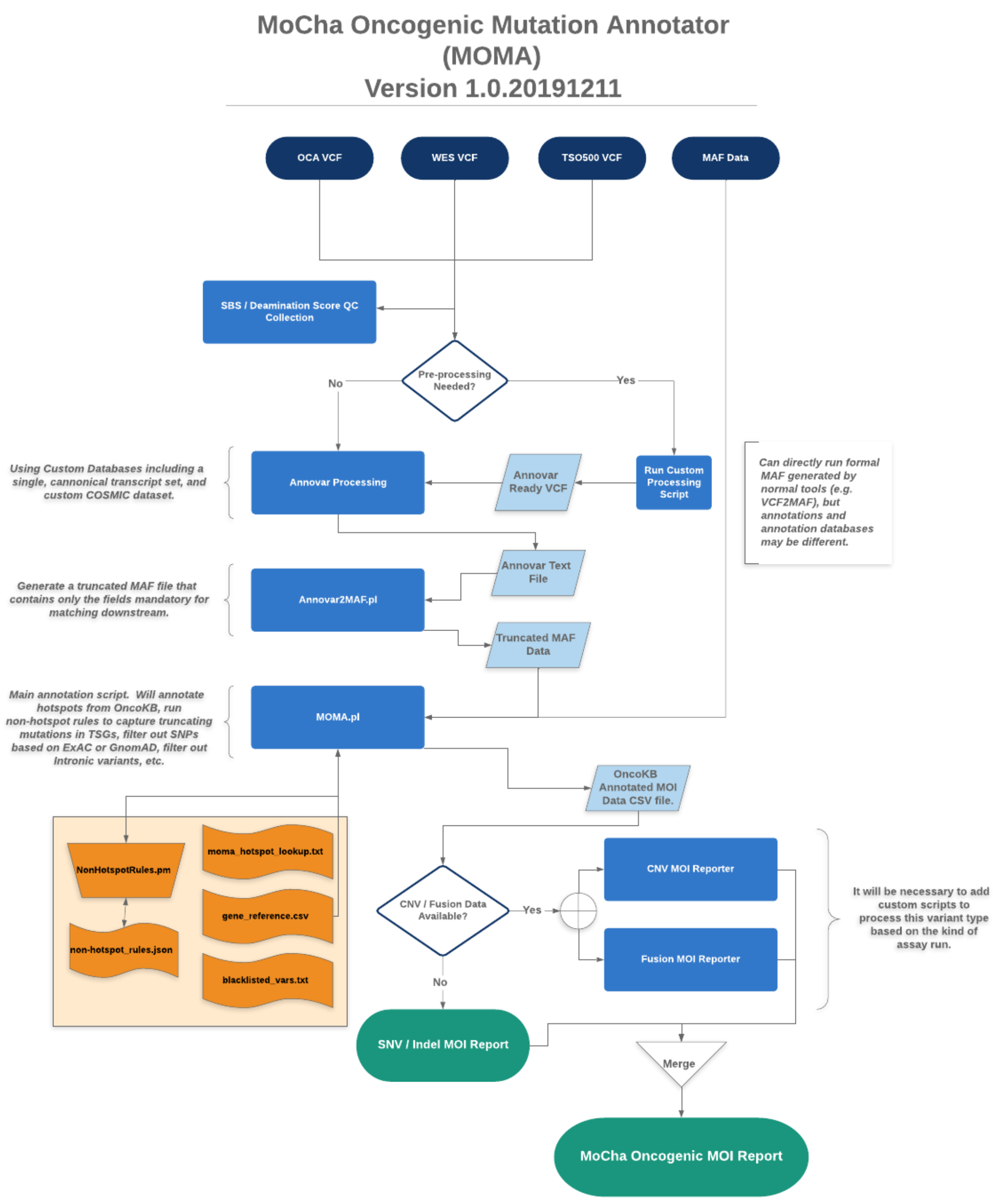
INTRODUCTION

The MoCha Oncogenic MOI Annotator (MOMA) utility is a sequencing platform agnostic tool used to annotate variants from a NGS sequencing assay and classify the variants as Mutations of Interest (MOIs) or Variants of Unknown Significance (VuSes). These classifications are based on annotation data from Annovar, which will indicate a variant's location, functional effect, population frequency information, etc., and mapping them to OncoKB, where the variant's *Oncogenicity* and *Effect* can be determined.

In addition, simple filtering is performed on the output data to remove variants that are not clinically relevant. We remove variants that are above a set population frequency as determined from GnomAD, ExAC, and 1000G, as well as non-coding variants (i.e. intronic) and synonymous variants. The remaining calls, then, are mapped to OncoKB and the variant's Oncogenicity and Oncogenic Effect annotation is added to the variant call if there is a match.

MOMA is designed to be compatible with VCF files from all NGS platforms, either natively, or with the addition of a simple helper script to modify the input to be compatible with the annotation pipeline. MOMA can also run starting from a MAF file. In fact, the first steps of the MOMA pipeline when starting from a VCF file, is to annotate the data and generate a MAF file that the tool will use for downstream processing. As shown below, the spirit of the tool is to be able to accomodate any kind of data, using helper scripts to stage the data in a way that can be easily processed through the rest of the tool

1.1 MOMA Workflow


MoCha Oncogenic Mutation Annotator (MOMA)
Version 1.0.20191211
OCA VCF
WES VCF
TSO500 VCF
MAF Data
SBS / Deamination Score QC Collection
Pre-processing Needed?
No
Yes
Using Custom Databases including a single, cannonical transcript set, and custom COSMIC dataset.
Annovar Processing
Annovar Ready VCF
Run Custom Processing Script
Can directly run formal MAF generated by normal tools (e.g. VCF2MAF), but annotations and annotation databases may be different.
Annovar Text File
Generate a truncated MAF file that contains only the fields mandatory for matching downstream.
Annovar2MAF.pl
Truncated MAF Data
Main annotation script. Will annotate hotspots from OncoKB, run non-hotspot rules to capture truncating mutations in TSGs, filter out SNPs based on ExAC or GnomAD, filter out Intronic variants, etc.
MOMA.pl
OncoKB Annotated MOI Data CSV file.
NonHotspotRules.pm
non-hotspot_rules.json
moma_hotspot_lookup.txt
gene_reference.csv
blacklisted_vars.txt
CNV / Fusion Data Available?
Yes
No
CNV MOI Reporter
Fusion MOI Reporter
It will be necessary to add custom scripts to process this variant type based on the kind of assay run.
SNV / Indel MOI Report
Merge
MoCha Oncogenic MOI Report

Each MOMA run should start with a VCF file from the assay platform of choice. From that VCF, a Single Base Substitution 6 (SBS-6) matrix is generated and the number of base changes corresponding to those 6 categories is tallied, along with computing the number of transitions, transversions, transition to transversion ratio (Ts/Tv) and deamination score. See more about this data in the *SBS-6 Output* for details.

If your pipeline outputs copy number variants (CNVs) and / or translocations (gene fusions), MOMA can possibly handle that as well. As of this version, MOMA will handle these data natively for Oncomine Comprehensive Assay (OCA) and TruSight Oncology 500 ctDNA. As these kinds of data are quite varied in their reporting format, helper scripts will most certainly need to be written to handle those outputs. However, once the data is formatted appropriately with a helper script, the output can be annotated with OncoKB and merged into the final dataset just like the above mentioned platforms.

Note: In order to process these extra variant types, additional input files may be required. See *MOMA Output* for details.

Typical output from a MOMA run will be a final MOMA Report as a CSV file, as well as a directory of some other informative output files. The table below indicates these files and their purpose.

Table 1: **Typical MOMA Output Files**

Output File	**Description**
.log	Log file from the pipline, containing useful information about the run, version numbers for resource files, information about some variants that have been filtered out, and whatnot.
.annovar.txt	Text file output from Annovar that will be used to generate a MAF file.
.annovar.vcf	The VCF version of the Annovar output. Not currently used for anything.
.maf	MAF file of annotated variants. This file is somewhat truncated from the full set of MAF fields, but can still be parsed by many conventional MAF parsing tools.
.moma_report_<date>.csv	The final MOMA report.
.sbs_metrics.csv	Data for the Single Base Substitution 6 (SBS-6) matrix as determined from the VCF. Useful in determining if there is deamination artifacts for example.

More details on the expected output from MOMA can be found in the *MOMA Output* section.

CHAPTER

TWO

INSTALLATION

As this package is a collection of Perl and Python scripts, installation is simple, and requirements are few. The package can be installed on any *nix system with at least 8GB RAM for smaller analyses (e.g. Oncomine Comprehensive Assay) and maybe up to 32GB of RAM for larger, WES analyses.

2.1 Requirements

The following packages and tools are required to run this plugin:

- Python3
- Perl v5.26+
- Annovar
- vcftools
- samtools
- bedtools

Be sure that these elements have been properly installed and are availble in your `$PATH`.

2.1.1 Scripting Languages

All of the code base for this utility is written in either BASH, Perl or Python3. You can probably use whichever system Perl is available to you (tested 5.26.x to 5.31.x) as there are no major, specialized language requirements (see below for module requirements). Please be sure to have Python3 installed as it is required for MOMA, and Python 2.7 will not work. As of this writing Python 2.7 is slated to be deprecated by the end of 2020, and its continued use is discouraged.

Additionally, there are some Python and Perl libraries that should be installed as they are not typically part of the standard installation:

Table 1: **Additional Required Modules for Each Scripting Language**

Language	Package / Library
Perl	Data::Dump Text::CSV Sort::Versions Log::Log4perl
Python3	pysam natsort

Note: The python *natsort* library is required as well, but to help control versioning, a version of this library has been included in the package.

These libraries can be installed using the typical tools / methods normally used to install these components (e.g. `cpan` or its more streamlined cousin `cpanm` for Perl and `pip3` for Python3). Be sure that the modules can be found in your `$PERLLIB` / `$PYTHONPATH`, and can be loaded.

2.1.2 Annovar Variant Annotation Package

In order to determine the coding sequence change, protein change, variant location, population frequency value, etc. for each variant in the VCF file, the data need to be annotated. Annovar was chosen due to its flexibility (the tool uses databases for this task, which can easily be modified and created), speed, and lighter footprint / resources. There certainly are other tools that can be used for the same task (e.g. VEP), and we are not endorsing one tool as being better than any other. For the purposes of MOMA, however, we decided that Annovar would be a good resource for the annotation component during the development of the tool.

In order to ensure that the data are always the same coming from the pipeline, and since Annovar can not be distributed with this package, it must be installed and moved into this package as instructed below. We do not recommend using any other system Annovar at this time, and there has been a check set up to ensure that a local copy of Annovar has been copied into the package.

2.1.2.1 Annovar Package Installation

As indicated above, Annovar is employed by this plugin to do variant annotation, and as such needs to be downloaded and installed along with some databases. You can find the Annovar documentation and installation instructions here

Once downloaded, the Annovar installation should be put into the `lib` dir in the MOMA package without the database files (essentially just the Annovar Perl scripts). We'll store the database files in a different location in the next step.

2.1.2.2 Annovar Database Installation

We use some of the publicly distributed Annovar libraries for this package, along with some custom ones. Since they are far too large to keep within this Github repo, they need to be obtained from a different resource.

Todo: I need to figure out a way to distribute these database files! Maybe I can get a tarball or something of these that can be used?

The following are the databases used by this plugin:

Required Annovar Databases

- **Custom Databases**
  - hg19_trunc_refGene, hg19_trunc_refGeneMrna (custom refgene database).
  - hg19_cosmic89_noEnst (custom COSMICv89 database)
- **Default Databases**
  - hg19_knownGene
  - hg19_avsnp142
  - hg19_dbnsfp35a

- hg19_clinvar_20190305
- hg19_popfreq_all_20150413
- hg19_gnomad_exome

Since the required Annovar databases total more than 40 GB in size they can not be included in this repository. Instead, you must download the databases following the instructions in the Annovar Documentation Once the files are downloaded, they should be decompressed and stored in a directory called `annovar_db` in the `resources` dir of the package (full path: `mocha_oncogenic_moi_annotator/resource/annovar_db/`).

The custom databases, as indicated above, can be obtained from here:

Todo: Maybe I can put these on S3 for download?

2.1.2.3 Human Reference hg19

For some steps of the pipeline a human reference hg19 (GRCh37) FASTA file is required. This file can be obtained from the UCSC Golden Path Repository Once downloaded, place this file in the `resources` directory. To conserve space, this file should be gzipped, and it will be indexed during the first run.

2.2 Setting up MOMA

Todo: Write a setup script that can help automate all of this.

Setting up MOMA is as simple as getting the package, getting Annovar, and getting the necessary resource files (i.e. Annovar database and human reference files), and placing all in the correct location.

1. Clone the MOMA repository from the MOMA github repository.
2. Download the latest version of Annovar from the Annovar source
3. Create a directory called `annovar` in the `lib` dir in the package root. The full path should be:

   ```
   mocha_oncogenic_moi_annotator/lib/annovar/
   ```

4. Move the Annovar perl scripts from Step 3 into this new directory. You'll typically see 6 scripts:
   1. `annotate_variation.pl`
   2. `coding_change.pl`
   3. `convert2annovar.pl`
   4. `retrieve_seq_from_fasta.pl`
   5. `table_annovar.pl`
   6. `variants_reduction.pl`

   We don't need all of the annovar scripts for this package, but there's no reason to exclude any for now.
5. Get the human reference hg19 fasta.gz file and place into the `resources` dir within the package root.
6. Create a new directory in `resources` called `annovar`, get the Annovar database files, and place them into the new directory.

In the end, you should have a directory tree that looks like this:

Listing 1: **Typical MOMA Package Structure**

```
lib
├── annovar
│   ├── annotate_variation.pl
│   ├── coding_change.pl
│   ├── convert2annovar.pl
│   ├── retrieve_seq_from_fasta.pl
│   ├── table_annovar.pl
│   └── variants_reduction.pl
├── bin
│   └── natsort
├── logger.py
├── natsort
│   ├── compat
│   ├── __init__.py
│   ├── __main__.py
│   ├── natsort.py
│   ├── ns_enum.py
│   ├── __pycache__
│   ├── unicode_numbers.py
│   ├── unicode_numeric_hex.py
│   └── utils.py
├── NonHotspotRules.pm
├── __pycache__
│   ├── logger.cpython-36.pyc
│   ├── logger.cpython-37.pyc
│   ├── utils.cpython-36.pyc
│   └── utils.cpython-37.pyc
└── utils.py
LICENSE.txt
MoCha_Oncogenic_Mutation_Annotator.py
moma_plugin.py
resource
├── annovar_db
│   ├── hg19_avsnp142.txt
│   ├── hg19_avsnp142.txt.idx
│   ├── hg19_clinvar_20190305.txt
│   ├── hg19_clinvar_20190305.txt.idx
│   ├── hg19_cosmic89_noEnst.txt
│   ├── hg19_cytoBand.txt
│   ├── hg19_dbnsfp35a.txt
│   ├── hg19_dbnsfp35a.txt.idx
│   ├── hg19_gnomad_exome.txt
│   ├── hg19_gnomad_exome.txt.idx
│   ├── hg19_knownGene.txt
│   ├── hg19_popfreq_all_20150413.txt
│   ├── hg19_popfreq_all_20150413.txt.idx
│   ├── hg19_trunc_refGeneMrna.fa
│   └── hg19_trunc_refGene.txt
├── blacklisted_vars.txt
├── count.txt
├── gene_reference.csv
├── hg19.fasta.gz
├── hg19.fasta.gz.fai
├── hg19.fasta.gz.gzi
```

(continues on next page)

(continued from previous page)

```
├── mocha_tso500_ctdna_hotspots_v1.072018.bed
├── moma_cnv_lookup.tsv
├── moma_fusion_genes.tsv
├── moma_hotspot_lookup.txt
└── non-hotspot_rules.json
run_moma_pipeline.py
scripts
├── annovar2maf.pl
├── annovar_wrapper.sh
├── calc_tmb.pl
├── calc_tstv_deam.py
├── collate_moma_reports.pl
├── get_cnvs.pl
├── get_fusions.pl
├── get_var_counts_from_moma_results.pl
├── moma2rave.py
├── moma.pl
├── simplify_vcf.pl
├── tso500_cnvs.pl
├── tso500_fusions.pl
└── usage.sh
templates
├── barcode_block.html
├── barcode_summary.html
└── progress_block.html
test
├── gen_tests
│   ├── 19-31014-002-Q_cfNA_rep1.clean.maf
│   ├── 19-32096-004-SCRN_cfTNA_rep1.clean.annotated.filtered.maf
│   ├── H7T7_cfTNA_rep3.clean.annotated.filtered.maf
│   ├── H7T7_cfTNA_rep3.clean.maf
│   ├── nhs_tests.json
│   ├── nhs_test.truncmaf
│   ├── sample.truncmaf
│   └── test_nonhs_rules_module.pl
├── ocav3
│   ├── 0CFDXX_IonXpress_079.vcf
│   └── oca.vcf
├── tso500
│   ├── Horizon_2-000_rep1.cnv.fc.txt
│   ├── Horizon_2-000_rep1.fusion.txt
│   ├── Horizon_2-000_rep1.vcf
│   ├── Lovo_nuc.vcf
│   ├── npDNA121_MS_rep1.vcf
│   ├── poolcf88_RDH_rep1.vcf
│   ├── W313718105511_cfNA.vcf
│   ├── W313718105515_cfNA.vcf
│   └── W313718105516_cfNA.vcf
└── wes
    ├── 114434.consensus.vcf
    └── 128128~338-R~L42~WES.merged.vcf
_version.py
```

Note: There are some files and scripts in this package that are not currently used, but are intended for use in Ion Torrent Plugins downstream. They can be ignored for now.

2.3 Running Tests

Included in the package is a set of test VCF files that can be run through MOMA. You can find these located in the `test` directory within the package.

Todo: Set up a simple test harness script to run all tests at once and compare the data.

You can simply attempt to process each platform specific VCF file through MOMA following the usage instructions in the *Running MOMA* section. If these tests can complete successfully, you have a fully working instance, ready to process samples.

CHAPTER

THREE

RUNNING MOMA

Running MOMA is as simple as executing the main pipeline script, `run_moma_pipeline.py`, and telling the utility what the source of the data is, and where the VCF file it should process is. Other options are available for additional data formatting and output, as well as additional filtering. A list of all valid options and parameters can be obtained by running the pipline script with the `-h` or `--help` argument:

Listing 1: **MOMA Help Docs**

```
$ run_moma_pipeline.py -h
usage: run_moma_pipeline.py [-h] -s {oca,wes,tso500,genmaf} [-g <gene>]
                            [--cu FLOAT <amp threshold>]
                            [--cl FLOAT <loss threshold>]
                            [--reads INT <reads>]
                            [--fvaf FLOAT <fusion allele frequency<]
                            [-p <pop_frequency>] [-C] [-F] [-n <sample_name>]
                            [-o <output_directory>] [-k] [-q] [-V]
                            [-r <protocol;treatment_id;specimen_id>] [-v]
                            <VCF File>

MoCha Oncogenic Mutation Annotator Tool (MOMA) This utility will take a VCF
file from MoCha pipelines including the Oncomine Cancer Assay (OCA), TSO500
ctDNA assay, and our current whole exome sequencing (WES) assay, and generate
a report that contains all coding variants that are non-synonymous and not
likely SNPs and adds OncoKB annotations to determine which of them are
oncogenic and not. In addition to the SNVs and Indels, we can also add copy
number variation (CNV) and fusion data to the output in one master report.

positional arguments:
  <VCF File>            VCF file on which to run the analysis. VCF files must
                        be derived from the Ion Torrent TVC plugin.

optional arguments:
  -h, --help            show this help message and exit
  -C, --CNV             Add CNV data to the output. CNV reporting is off by
                        default.
  -F, --Fusion          Add Fusion data to the output. Fusion reporting is off
                        by default.
  -n <sample_name>, --name <sample_name>
                        Sample name to use for output.
  -o <output_directory>, --outdir <output_directory>
                        Directory to which the output data should be written.
                        DEFAULT: <sample_name>_out/
  -k, --keep            Keep all intermediate files and do not delete. Useful
                        for troubleshooting.
  -q, --quiet           Suppress output to the command line. Will still write
```

(continues on next page)

(continued from previous page)

```
                        to log file.
  -V, --Verbose         Output more log messages and general stderr messages.
  -r <protocol;treatment_id;specimen_id>, --rave <protocol;treatment_id;specimen_id>
                        Generate a CSV file that can be uploaded into Rave Web
                        Reporting to support a clinical trial. The protocol
                        number should be "10231" in general, the treatment ID
                        will be the site identifier (e.g. "IA004-0004"), and
                        the specimen ID will be the same as what the lab used
                        for sequencing (e.g. "10231-6D4410K1-1"). Since we are
                        using semicolons to delimit, the whole string must be
                        enclosed in quotes.
  -v, --version         show program's version number and exit

Required Arguments:
  -s {oca,wes,tso500,genmaf}, --source {oca,wes,tso500,genmaf}
                        Type of data being loaded. Will determine what steps
                        and script are necessary to process these data.

Filtering Arguments:
  -g <gene>, --genes <gene>
                        Gene or comma separated list of genes to report. Use
                        the string "all" to remove the gene filter and report
                        data for all genes. DEFAULT: all.
  --cu FLOAT <amp threshold>
                        Threshold for calling amplifications. Will be in
                        ploidy space for Oncomine data, and fold change for
                        TSO500. Only valid with the `-CNV` option selected.
                        Default: None.
  --cl FLOAT <loss threshold>
                        Threshold for calling deletions. Will be in ploidy
                        space for Oncomine data, and fold change for TSO500.
                        Only valid with the `-CNV` option selected. Default:
                        None.
  --reads INT <reads>   Threshold for reporting fusions. This filter arg is
                        used for Oncomine results only, and is nly valid with
                        the `-Fusion` option selected. Default: 250.
  --fvaf FLOAT <fusion allele frequency<
                        Variant Allele Frequency (VAF) threshold for fusion
                        reads. This filter is only applicable to the TSO500
                        assay, which does not report read counts like
                        Oncomine. Default: 0.01.
  -p <pop_frequency>, --popfreq <pop_frequency>
                        Population frequency threshold above which variants
                        will be filtered out. Default: 0.01.
```

3.1 Basic Usage

3.1.1 Required Arguments

Every MOMA run must start with a VCF file from a developed pipeline, and MOMA must be aware of the type of data being submitted using the `--source` (`-s`) argument.

Note: There is an option to load general MAF files, but this functionality has not yet been fully developed and can not be used at this time. This functionality will be available in future releases to expand the data that can be passed into MOMA.

3.1.1.1 Data Source Argument

Depending on the type of data being loaded into MOMA there are different helper scripts that must be invoked, and there are different ways to parse variant calls adequately. For example, the TSO500 ctDNA assay will generate variants with variant allele frequencies (VAFs) < 1%, which is fine. However, sub-1% VAF variant calls, if they should make it into the VCF files, are not generally valid for other platform / assays and will be removed. Therefore, you can not run the pipeline without explicitly telling MOMA what kind of data is to be expected using the `--source` option.

3.1.1.2 MOMA Input Files (Data File Argument)

The other required argument is a valid VCF file that has been generated from either an Ion Torrent or Illumina sequencing assay. As of this writing, we can accomodate data stemming from:

1. Ion Torrent Oncomine Comprehensive Assay
2. TruSight Oncology 500 (TSO500) ctDNA Assay
3. Agilent SureSelect WES Assay using MoCha PDX Pipeline.

It is quite possible to easily adapt to other chemistries / platforms by the addition of a simple helper script or two (see *Technical Details*). Generally, though, these are the main types of NGS data that we would expect to see, and MOMA is perfectly capable of processing any of these.

In the future, we will be able to accomodate any standard MAF file as input, though there are some caveats to be aware of:

- Since the MAF generation will be from something like vcf2maf, there may be different annotation rules and sources in place compared to the standardized set in MOMA. Therefore, we might expect to find some differences in the data simply due to different annotation sources / rules. In general, the core data should be the same.
- Protein and Coding Sequence Change annotations may be different in data stemming from VCF files compared to MAF input. Since we are selecting just a single set of transcripts to use for each gene in MOMA, we can only ever get one annotation for each alteration. However, tools like VCF2MAF will give data for all transcripts, and this can lead to differences in reported protein and coding sequence change annotations.
- Depending on what data sources are used to generate the MAF, there might be differences in population frequency, ClinVar, and other databases. We are locking the analyses to a set of included databases, and there may be differences in the annotation sources coming from other tools.
- Starting from a MAF file will, of course, bypass the variant annotation step since MAF data is already fully annotated. This means that there is no intermediate Annovar data to use as a resource to get more information about a variant call of interest.

In general it's much preferred to submit a VCF file to the MOMA pipeline in order to keep the data harmonized and ensure that the annotation sources and styles are the same across different studies, datasets, etc.

3.1.2 Optional Arguemnts

There are several optional arguments to help customize the output and processing of specimens through MOMA. These can be binned into two groups, *Output and Formatting* and *Filtering Arguments*

3.1.2.1 Output and Formatting

For MOMA output, there are some useful arguments to help format and select certain elements for output. The table below highlights these arguments and how they're intended to help with data formatting and output options.

Table 1: **Output and Formatting Options**

Long Argument	Short Argument	Description
`--name`	`-n`	Custom sample name for the output. If no argument passed, will use the VCF filename as the sample name.
`--output`	`-o`	Directory in which the output data will be written. By default will be `<sample_name>_out`
`--rave`	`-r`	For the NCORP 10231 trial, we need a custom CSV file to be generated and uploaded to the Theradex Rave Web Reporting interface. This option will also generate that file. See the help docs for the specific IDs that are required.
`--keep`	`-k`	Keep all intermediate output files. By default, the temporary intermediate files are cleaned up after the run. Selecting this option will keep those rather than discarding them for troubleshooting.
`--quiet`	`-q`	Do not print log information to the screen. By default all log data is written to a log file as well as to the screen. Selecting this option will only send the log info to a file.
`--Verbose`	`-V`	Opposite of `--quiet`, this will print much more information to the screen and log, files, including debugger information.

Beyond these simple options, there are also some variant type specific options that help with adding more variant type data to the output. In particular, if there is CNV and / or Fusion data present in the assay results and one would like to add these data to the output, the following sections will inform on the process to do so. This is not a standard option at this time because there are many platforms / assays that do not detect and report on these variant types at this time. In addition, the data are not as well standardized in the way they're output, and so coming up with a *one-size-fits-all* solution is really not feasible.

3.1.2.2 CNV Output Arguments

If your pipeline generates Copy Number Variant (CNV) output, we can accomodate this in MOMA with the `-C` or `--CNV` argument. As of this writing, this functionality is only developed for the Oncomine Comprehensive Assay (OCA) and TruSight Oncology 500 (TSO500) data. As mentioned above, different platforms will have slighly different specifications for the input data source.

Ion Torrent CNV Processing

The same VCF used for small variant processing contains CNV data if this panel was run through the Oncomine pipeline in Ion Reporter. Just simply selecting this option is enough for MOMA to add the CNV results to MOMA output.

TSO500 CNV Processing

The CNV data for this platform is output in a different file. From the pipeline, we'll get a VCF called `<sample_name>.vcf` for small variants, and one called `<sample_name>.fc.cnv.txt` for CNVs. If you would like to get these data in the output file, this `fc.cnv.txt` file must be in the same location as the VCF you're processing, and must be named with exactly the same sample name as the small variants VCF and ending with `fc.cnv.txt`.

WES CNV Processing

As of this version, this is not yet supported.

3.1.2.3 Fusion Output Argument

If your pipline generates translocation, gene fusion, or gene rearrangement data, this can be accomodated by MOMA and added to the final report as well. Adding the `-F` or `--Fusion` option will run the appropriate scripts to parse these data, annotate them, and add them to the final set. As of this writing, only the Oncomine Comprehensive Assay (OCA) can be used for this output, and the rest are in development.

Ion Torrent Fusion Processing

As with the CNV data, the fusion data for this platform is included in the Ion Reporter VCF. Simply selecting this option for output is sufficient.

TSO500 Fusion Processing

As of this version, this is not yet supported.

WES Fusion Processing

As of this version, this is not yet supported.

3.1.2.4 Filtering Arguments

There is little need for much manual filtering in this tool. By default, the goal is to simply annotate and report the findings. However, there is some flexibility requirements for things like only reporting on specific genes, changing the population frequency filtering threshold, and things like that. Here are a list of arguments that can be used to filter the output data.

Table 2: **Filtering Options**

Long Argument	Short Argument	Description
`--gene`	`-g`	Only output data for a specific gene or comma separated list of genes (e.g. `gene1,gene2,gene3`). By default data for all genes covered by OncoKB are reported.
`--cu`		Copy gain threshold for CNV output. If parsing Oncomine data, units will be in copies (reference: 2 copies). If using TSO500, the units will be in fold change (reference: 0).
`-cl`		Copy loss threshold for CNV output. Just as with the `--cu` option, will be in copies for Oncomine and fold change for TSO500.
`--reads`		For Oncomine fusions, threshold above which fusions will be reported, in units of read counts. This is not applicable to any other assay.
`-fvaf`		For TSO500 fusions only, allele frequency above which fusions will be reported. This is no applicable to any other assay.
`--popfreq`	`-p`	Threshold above which variants will be filtered if their population frequency max is above.

As mentioned, the more significant thresholding and filtering should probably be done in the variant calling pipeline, and as such the options are limited. But, these metrics should suffice for some level of removing calls from the report if they are really outside of the range we want to report.

3.2 Variant Type Inputs and Processing

3.2.1 SNV / Indel (small variants) Processing

At the very core, MOMA will handle single nucleotide variants (SNVs) and insertion deletion variants (indels), otherwise known as small variants, by default. Without any additional arguments, and by passing an acceptable VCF or MAF file, one will always get small variant output as the default. Copy number variants (CNVs) and fusions are optional outputs at this time since not every panel can generate these variant types.

To generate the most basic MOMA report, we execute the command:

```
run_moma_pipeline.py --source <data_source> <vcf>
```

This will launch the MOMA pipeline on the input VCF file stemming from the data source indicated, and will produce a simple MOMA report containing SNV and Indel (i.e. small variant) data for the sample. All output will be stored in a directory containing the VCF name.

On a small dataset, like OCAv3, the run will take less than 5 minutes, and might generate a dataset that looks like this (assume the sample name is *test*):

Listing 2: **MOMA Output Directory Listing**

```
test_out/
├── moma_reporter_test_20200312.log
```

(continues on next page)

(continued from previous page)

```
├── test.annovar.txt
├── test.annovar.vcf
├── test.maf
├── test_moma_report_20200312.csv
└── test_sbs_metrics.csv
```

More details on the output files are give in the *MOMA Output* section below.

3.2.2 CNV Processing

As mentioned above, if one would like to add the CNV data to the output, this can easily be achieved by using the `--CNV` option. The data can be then parsed and output from either the main VCF file (in the case of OCA) or from a secondary file (in the case of TSO500). The output files will look the same as above, but looking at the main MOMA output report, one will see these data have populated a new section in the report.

As mentioned above, if one is going to try to get CNV data from a TSO500 run, a second file must be loaded. The output from the TSO500 pipeline will generate CNV data in a second file that will be named with the sample name and have the ending `.fc.cnv.txt`. For example, if the VCF is called `sample1.vcf`, the corresponding CNV file should be `sample1.fc.cnv.txt`. If, for some reason, the CNV file does not adopt the sample name and / or format as indicated, MOMA will not be able to read the file. In this case, you should manually name the file to match this structure.

Also, as mentioned earlier, CNV processing can not yet be done on any other panels apart from OCA and TSO500. Other panels are in development.

3.2.3 Fusion Processing

Fusions can also be added to the output if available using the `--Fusion` option as indicated above. Also, just as with the CNV data, Fusion data will be added to a new section in the MOMA report, should this option be chosen.

As of this writing, no fusion output is available for WES data. However, for Oncomine data and TSO500, this can be handled just as with the CNV data. OCA adds fusion calls to the main VCF, while TSO500 has a third file that will contain these data. This file will have the naming format `<sample_name>.fusions.txt`. Just as with the CNV option, if one would like to get these output in the MOMA report, this file must be present in the same location as the VCF file and must have the same naming convention.

3.3 MOMA Output

The main MOMA output is a CSV report called `<sample>_moma_report_<date>.csv`. This report contains each reportable variant (a *MOI* or a *VuS*), one per row, along with the most pertinent information regarding the details of the call. There is a section for each variant type - small variants, CNVs and fusions - the latter two of which will only be generated if requested and possible.

In short, variants will only make it to the output tables if they pass some simple criteria:

1. Small variants are located in coding regions and / or affect splicing (i.e. intronic).
2. Small variants are non-synonymous.
3. Small variants are not found to be in above 1% (adjustable) of any population in the databases.
4. Variants are not below read count, VAF, etc. thresholds or are otherwise blacklisted. See also *MOMA Logging* below for more details.

Note: There are almost no filtering requirements for CNVs and Fusions. We only filter out those calls that are below the reporting threshold of the assay.

After removing calls based on those criteria, the variants are cross-referenced with the OncoKB database, and annotations are added indicating the *Oncogenicity* (generally Oncogenic or Likely Oncogenic) and *Effect* (Gain of Function or Loss of Function) of the call based on whether or not the call maps to an identity or non-hotspot rule (see *MOMA Workflow* and *Technical Details* for details). These annotated variants are considered to be **Mutations of Interest** or **MOIs**. Any variants that have not been filtered out based on the above criteria, but do not have an OncoKB annotation are considered to be **Variants of Unknown Significance** or **VuSes**.

The final report will have the most pertinent information for each MOI or VuS, including positional information, functional information, and clinical information. The following elements are available for each section (i.e. variant type) of the report:

Note: The SNV / Indel data will be consistent across platforms. However, due to the differences in the data collected for CNV and Fusion results, the table elements will be different, depending on the assay run.

Table 3: **Table Elements for The SNV / Indel Section of the MOMA Report**

Header Element	Description
Chr	Chromosome position where variant is located.
Pos	Start position of variant, relative to hg19 reference sequence.
Ref	Reference base at this position.
Alt	Alternative Allele (i.e. mutation) at this position.
VAF	Variant allele frequency. The percent of alternative reads to reference reads for this variant.
Ref_Reads	The number of reference reads at this position.
Alt_Reads	The number of alternative allele reads at this position.
dbSNP_Id	If the variant can be found in the dbSNP database, the ID under which it can be found.
COSMIC_Id	If the variant can be found in the Catalogue of Somatic Mutations in Cancer (COSMIC) database, the ID under which it can be found.
Gene	Official HUGO symbol for the gene in which the variant is located.
Transcript	RefSeq transcript ID used for mapping the coding sequence and amino acid change created by the variant.
CDS	Coding Sequence change created by the alteration in the context of the indicated transcript.
AA	Amino Acid change created by the alteration.
Location	Generic location where the variant resides, usually exon number or intronic.
Function	Functional effect caused by the variant in transcript space.
SIFT	SIFT prediction of variant's effect on protein function.
PolyPhen	PolyPhen prediction of variant's effect on protein function.
Clinvar_Significance	Clinical significance of variant as reported in ClinVar.
Clinvar_Review_Status	Information about the status of peer reviews of the clinical significance of the variant in the ClinVar database.
MOI_Type	If a variant is a MOI, the criteria under which the variant qualifies. More details about this can be found in the *Technical Details* section.
Oncogenicity	If a variant maps to OncoKB, oncogenicity as defined by OncoKB. Usually *Oncogenic* or *Likely Oncogenic*.

Continued on next page

Table 3 – continued from previous page

Header Element	Description
Effect	If the variant maps to OncoKB, effect as defined by OncoKB. Usually *Gain of Function* or *Loss of Function*.

Table 4: **Platform Specific CNV Table Output Fields**

Platform	Field	Description
Oncomine	*Chr*	Chromosome where the copy number even has taken place.
	Start	Start position for the reading. This represents the 5' most position of the first amplicon covering the region.
	End	End position for the reading. This represents the 3' most position of the last amplicon covering the region.
	Gene	HUGO symbol of the gene in this region.
	CN	Copy number. This is the total number of copies of the region of interest detected. Normal reference copies is 2.
	CI_05	5% confidence interval. <more description>
	CI_95	95% confidence interval. <more description>
	MAPD	Median Absolute Pairwise Difference.
	Oncogenicity	If a variant maps to OncoKB and is a MOI, oncogenicity as defined by OncoKB. Usually *Oncogenic* or *Likely Oncogenic*
	Effect	If the variant maps to OncoKB and is a MOI, effect as defined by OncoKB. Usually *Gain of Function* or *Loss of Function*.
TSO500	*<field>*	<description>
	<field>	<description>
WES	*<field>*	<description>
	<field>	<description>

Table 5: **Platform Specific Fusion Table Output Fields**

Platform	Field	Description
Oncomine	*Fusion*	Name of the fusion detected. (e.g. *EML4-ALK*).
	Junction	String indicating the exon of each gene where the fusion takes place. For example, if exon 14 of *EML4* is fused to exon 20 of *ALK*, this will be represented with the string `E14A20`.
	ID	If the fusion has been described in COSMIC or similar, and the ID has been loaded into the fusion reference, this field will contain that ID. Otherwise, it will be empty.
	Driver_Gene	Oncogenic driver gene of the fusion pair.
	Partner_Gene	Partner gene of the fusion pair.
	Read_Counts	Number of reads observed for this even in the specimen.
	Oncogenicity	If a variant maps to OncoKB and is a MOI, oncogenicity as defined by OncoKB. Usually *Oncogenic* or *Likely Oncogenic*
	Effect	If the variant maps to OncoKB and is a MOI, effect as defined by OncoKB. Usually *Gain of Function* or *Loss of Function*.
TSO500	*N/A*	N/A
	N/A	N/A
WES	*N/A*	N/A
	N/A	N/A

3.3.1 SBS-6 Output

In addition to the MOI and VuS data, MOMA will compute the Single Base Substitution 6 (SBS-6) matrix data, and tally the number of variants associated with these 6 possible changes:

1. C>A
2. C>T
3. C>G
4. T>A
5. T>C
6. T>G

Note: Typcially the data all get converted to positive strand space for this computation, which is why there are only 6 possible changes.

Additionally, the tool will compute the number of C>T base changes in the context of a CpG region (e.g. TCG > TTG) and the number of C>T base changes in non-CpG regions. The ratio of CpG changes to non-CpG changes can be used to identify samples that may be heavily deaminated due to exposure to formalin as part of the fixation and embedding process. Samples that are heavily deaminated (i.e. may have been overexposed to formalin or similar) can generate a high number of false positive calls and sequencing artifacts.

Along with comptuting a deamination metric, a ratio of the number of transitions (i.e. C>T, A>G) to the number of transversions (A>C, A>T, G>C, G>T) is computed. This transition / transverion ratio (Ts/Tv) can also be a measure of sample integrity and the likelihood of false positives due to artifact. Ideally, for panels that sequence the coding regions of genes (e.g. WES, amplicon based sequencing), the Ts/Tv ratio will be ~3. For whole genome sequencing (WGS) that number drops a bit closer to 2. If we observe a Ts/Tv ratio that is very much above 3 (Ts/Tv >= 4?), the quality of the specimen, and consequently sequencing data, is suspect.

These data are output in the log file for a quick snapshot (see *MOMA Logging*). If the sample appears to be damaged, either from an elevated deamination metric or a high Ts/Tv ratio, a warning will be output to the log file:

Listing 3: **SBS-6 Warning**

```
2020-03-25 14:51:14  [ WARN ]:   Ts/Tv ratio is high (4.157). Sample may be damaged.
```

Additionally a tab delimited data file called:

```
<sample_name>_sbs_metrics.csv
```

is produced by MOMA outlining these data. In the MOMA output, we can see the counts for each variant type as well as the Ts/Tv ratio and deamination metric.

Listing 4: **SBS-6 Output Example**

```
Sample   Tot_Vars Indels Ts/Tv C>A C>G C>T T>A T>C T>G C>T(CpG) C>T(other) C>T_Ratio
<sample> 166      7      2.614 12  15  57  8   58  9   30       27         0.90
```

Since the results above show a Ts/Tv ratio around the expected value, and the *C>T_Ratio* (deamination metric) is low (a consequence of having a nearly equal distribution of C>T changes at CpG site compared to other sites), the sample is probably good quality and can be used. A poor quality sample might look like this:

Listing 5: **SBS-6 Data for Poor Quality Sample**

```
Sample   Tot_Vars Indels Ts/Tv C>A C>G C>T T>A T>C T>G C>T(CpG) C>T(other) C>T_Ratio
<sample> 650      21     6.862 29  32  442 9   107 10  77          365           4.74
```

In this case, in addition to the large number of variants seen in the sample compared to other samples sequenced with this panel (average number of variants is typically ~200), we see that there are far more C>T changes at non-CpG sites compared to C>T sites, a hallmark of FFPE deamination artifacts.

Note: For now, the output is simple. Later on, it is intended to have OxoG signature data, as well as the full mutational signature data that can be used to assess a multitude of things.

3.4 MOMA Logging

An important set of output data for MOMA is the log file that is generated and stored in the output directory of data. This file (see *Example MOMA Log File Output* below) contains all of the runtime information for a MOMA run, including:

- MOMA version number.
- Version numbers for all databases used in the annotation and reporting.
- SBS-6 information.
- MOMA MOI and VuS counts.
- Information about some variants that have been filtered out. Generally only variants that are blacklisted or below some threshold.

This log file will be stored in the MOMA output directory for each run, and it is recommended to consult this file when trying to determine why a variant may have not been included in the output.

Note that during a typcial MOMA run, these data are also output on the terminal, unless the `--quiet` option was selected. However, there will always be a log file generated, even when `--quiet` is requested.

An example log file is below:

Listing 6: **Example MOMA Log File Output**

```
=============================================================================================
:::::  Running the MoCha Oncogenic Mutation Annotator (MOMA) pipeline - vv1.3.
↪20200310-dev1  :::::
=============================================================================================

2020-03-12 10:39:29  [ INFO ]:   Processing sample: test
2020-03-12 10:39:29  [ INFO ]:   Generating Ts/Tv, Deamination Score, and SBS-6␣
↪information for sample.
2020-03-12 10:39:31  [ INFO ]:   Results from SBS script:
            Found 166 variants in MSN30015_v1:
              SNVs:           159
              Indels:         7
            Computing SBS-6 and variant motifs...Done!
              Transitions:    115
              Transversions: 44
              Ts/Tv:          2.614
```

(continues on next page)

(continued from previous page)

```
              Deam Score:    0.90
2020-03-12 10:39:31  [ INFO ]:  Simplifying the VCF file.
2020-03-12 10:39:31  [ INFO ]:  Done simplifying VCF file. There are 166 variants in␣
↪this specimen.
2020-03-12 10:39:31  [ INFO ]:  Annotating the simplified VCF file with Annovar.
2020-03-12 10:41:14  [ INFO ]:  Running OncoKB Filter rules on data to look for␣
↪oncogenic variants.
2020-03-12 10:41:14  [ INFO ]:  Converting Annovar data to a truncated MAF file.
2020-03-12 10:41:14  [ INFO ]:  Running MOMA
2020-03-12 10:41:15  [ INFO ]:  MOMA Summary Report:
    2020/03/12 10:41:15 [ INFO ]:
    ================================================================================
                        Starting MoCha Oncogenic MOI Annotator Script
    ================================================================================

    2020/03/12 10:41:15 [ INFO ]: Using OncoKB file moma_hotspot_lookup.txt
    2020/03/12 10:41:15 [ INFO ]: OncoKB lookup file version: v4.1.092719.
    2020/03/12 10:41:15 [ INFO ]: Using TSG file 3.1.022820

    2020/03/12 10:41:15 [ INFO ]: Using a population frequency filter exac -> 0.01
    2020/03/12 10:41:15 [ INFO ]: Annotating 'test.maf'...
    2020/03/12 10:41:15 [ INFO ]:
        >>>  Variant Summary  <<<
            Total variants:  21
            Filtered out:    17
            Retained:         4

    2020/03/12 10:41:15 [ INFO ]:
        >>>  MOI Counts  <<<
            CALR C-terminal Truncating Mutations                 : 0
            CCND1 Truncating Mutations                           : 0
            CCND3 Truncating Mutations                           : 0
            CXCR4 Truncating Mutations                           : 0
            EGFR In-frame Deletion in Exon 19                    : 0
            EGFR In-frame Insertion in Exon 20                   : 0
            ERBB2 In-frame Insertion in Exon 20                  : 0
            FLT3 TyrK Domain Mutations                           : 0
            Hotspots                                             : 2
            JAK2 Exon 12 Alterations                             : 0
            KIT Mutations in Exons 9, 11, 13, 14, or 17          : 0
            MED12 Exon 1 or 2 Mutations                          : 0
            MET Exon 14 Splice Variants                          : 0
            MPL Exon 10 Alterations                              : 0
            NOTCH1 Truncating Mutations Upstream TA Domain       : 0
            NOTCH1 Truncating Mutations in the PEST Domain       : 0
            NOTCH2 Truncating Gain of Function Mutations         : 0
            NOTCH2 Truncating Loss of Function Mutations         : 0
            PPM1D Truncating Mutations                           : 0
            TERT Promoter Mutations                              : 0
            TP53 DBD Mutations                                   : 1
            Truncating in TSG                                    : 0
            =======================================================
            Total MOIs                                           : 3


    2020/03/12 10:41:15 [ INFO ]: Finished annotating. Printing results...
    2020/03/12 10:41:15 [ INFO ]: Writing results to /Users/simsdj/Dropbox/git_repos
```

(continues on next page)

(continued from previous page)

```
    /mocha_oncogenic_moi_annotator/test/ocav3/test_out/test.oncokb.maf (filter is
    off)
    2020/03/12 10:41:15 [ INFO ]: Done with /Users/simsdj/Dropbox/git_repos/mocha_on
    cogenic_moi_annotator/test/ocav3/test_out/test.maf!


2020-03-12 10:41:15  [ INFO ]:   Running final filters and preparing data for report.
2020-03-12 10:41:15  [ NOTE ]:   Removing variant TP53:chr17:7578373:T:C:p.D186G␣
↪because it is blacklisted.
2020-03-12 10:41:15  [ INFO ]:   Writing report data to /Users/simsdj/Dropbox/git_
↪repos/mocha_oncogenic_moi_annotator/test/ocav3/test_out/test_mocha_snv_indel_report.
↪csv.
2020-03-12 10:41:15  [ INFO ]:   Generating a CNV report for sample 'test'.

2020-03-12 10:41:15  [ INFO ]:   Writing report data to /Users/simsdj/Dropbox/git_
↪repos/mocha_oncogenic_moi_annotator/test/ocav3/test_out/test_mocha_cnv_report.csv.
2020-03-12 10:41:15  [ INFO ]:   Found 1 copy number events in the specimen.
2020-03-12 10:41:15  [ INFO ]:   Annotating with OncoKB lookup.
2020-03-12 10:41:15  [ INFO ]:   Done with CNV report.
2020-03-12 10:41:15  [ INFO ]:   Generating a Fusions report for sample test. Using␣
↪threshold of 250 reads.
2020-03-12 10:41:15  [ INFO ]:   Writing report data to /Users/simsdj/Dropbox/git_
↪repos/mocha_oncogenic_moi_annotator/test/ocav3/test_out/test_mocha_fusion_report.
↪csv.
2020-03-12 10:41:15  [ INFO ]:   No Fusion events found in this specimen.
2020-03-12 10:41:15  [ INFO ]:   Done with Fusions report.
2020-03-12 10:41:15  [ INFO ]:   Collating all variant data into a single report
2020-03-12 10:41:15  [ INFO ]:   MoCha OncoKB Annotator Pipline completed␣
↪successfully! Data can be found in /Users/simsdj/Dropbox/git_repos/mocha_oncogenic_
↪moi_annotator/test/ocav3/test_out.
```

3.4.1 Variant Counts and MOI Counts Tables

One of the more informative components of the log file output are the *Variant Summary and *MOI Counts* tables. These tables help give a snapshot of the number of variants kept, as well as the binning of the MOIs.

The Variant Counts table indicates the total number of variants that have passed basic filtering (i.e. location and function filters) and will be considered further. In the *Example MOMA Log File Output*, we can see that there were 21 variants that are located in coding regions and are non-synonymous:

Listing 7: **Example Variant Summary Table in Log File**

```
>>>  Variant Summary  <<<
        Total variants:  21
        Filtered out:    17
        Retained:         4
```

From here, MOMA will consider population frequency data for each of the calls and apply any other filters requested. In the example, 17 variants have been removed from the data, leaving 4 to be mapped to OncoKB data for MOI determination.

The MOI counts table breaks down the various MOI rules, most of which are *non-hotspot rules* and gives a tally for each criteria (see *Technical Details* for more information on these rules). In the example, we can see that of the 4 variants above, 3 of them mapped to a MOI rule:

Listing 8: **MOMA MOI Summary Table in Log File**

```
>>>  MOI Counts  <<<
        CALR C-terminal Truncating Mutations                : 0
        CCND1 Truncating Mutations                          : 0
        CCND3 Truncating Mutations                          : 0
        CXCR4 Truncating Mutations                          : 0
        EGFR In-frame Deletion in Exon 19                   : 0
        EGFR In-frame Insertion in Exon 20                  : 0
        ERBB2 In-frame Insertion in Exon 20                 : 0
        FLT3 TyrK Domain Mutations                          : 0
        Hotspots                                            : 2
        JAK2 Exon 12 Alterations                            : 0
        KIT Mutations in Exons 9, 11, 13, 14, or 17         : 0
        MED12 Exon 1 or 2 Mutations                         : 0
        MET Exon 14 Splice Variants                         : 0
        MPL Exon 10 Alterations                             : 0
        NOTCH1 Truncating Mutations Upstream TA Domain      : 0
        NOTCH1 Truncating Mutations in the PEST Domain      : 0
        NOTCH2 Truncating Gain of Function Mutations        : 0
        NOTCH2 Truncating Loss of Function Mutations        : 0
        PPM1D Truncating Mutations                          : 0
        TERT Promoter Mutations                             : 0
        TP53 DBD Mutations                                  : 1
        Truncating in TSG                                   : 0
        =======================================================
        Total MOIs                                          : 3
```

From the table data, 2 of the variants are hotspots, and 1 variant matches the non-hotspot rule, *TP53 DBD Mutations*. One of the 4 variants does not appear to map to any of the MOI rules and would be a VuS.

Note: The final blacklist, VAF, and Coverage filters are run after this step, and so it is possible that the final number of MOIs / VuSes is less than the total computed here.

These data are only intended to be a sneak peek / snapshot of the final report data and should not be considered the final result. In the end, the data that has propagated the MOMA report will be considered the real, final data. But, it's often helpful to get a glimpse at the findings, especially as they relate to the various MOI categories, before the pipeline is finished.

3.5 Troubleshooting

Troubleshooting in MOMA generally starts from the log file or output on the commandline. We can break this up into two concepts:

1. Missing or unexpected variants.
2. Pipeline errors.

For both elements, details can be found in the log file which may help shed some light on what may be going wrong with your run. For example if a variant is missing from the report, it may have been blacklisted, and this information will show up in the log file. The total tally of variants that have been removed from the data due to any kind of filtering is not comprehensive - doing this would lead to a bloated and unuseable log file. But, the critcial and less obvious cases (like blacklisted calls) will make it to the log file output.

Additionally, some of the kept intermediate files may have some extra information that will help determine the source of the issue where missing or unexpected variants is concerned. Further, the addition of the `--keep` option will allow all temporary and intermediate files to be kept, which can help in troubleshooting, and adding the `--Verbose` option will output more debugging information, which could be helpful for determining the source of pipeline errors and related issues.

3.5.1 Missing or Unexpected Variants

As a rule, and since MOMA is a variant annotation and reporting tool, **not a variant calling tool**, variants that do not make it to the report or conversely are added to the report unexpectedly, are this way because of the rules that have been established for reporting (see *Technical Details* for more information). We, therefore, can not say that MOMA *"missed"* calls, but rather, variants did not qualify for our reporting rules. Along those lines, we also can not say that MOMA reported false positives if there are variants that one does not agree are reportable in the final report. It is a simple matter of choice in terms of what's reported and what's not for the most part.

If there are calls that are not present or unexpected, more information can be obtained by looking at the intermediate files. In particular, the Annovar text file (`<sample>.annovar.txt`) or Annovar VCF file (`<sample>.annovar.vcf`) are good places to start since they contain every single variant that was reported in the VCF (i.e. unfiltered data) along with the annotations that are used as a basis for filtering. This information can help one understand why a variant made or was filtered from the list outside of data in the log file, and in a more granular way. The following cas may help to understand how to assess these phenomenon.

3.5.1.1 Case 1: Missing TP53 Variant from MOMA Report

You expected to find a TP53 hotspot MOI in the MOMA report, but it was not found. The first obvious and easy place to check is the log file. You may see that the variant was filtered out because it was either blacklisted or had a VAF / Coverage issue. In the case of blacklisted variant, you might see something like:

Listing 9: **MOMA Blacklisted Variant**

```
2020-03-12 10:41:15  [ NOTE ]:   Removing variant TP53:chr17:7578373:T:C:p.D186G␣
↪because it is blacklisted.
```

In this case, we can see that a the variant call was removed from the MOMA report since it was found on the blacklist, and is likely a SNP that was not found in the population database or a sequencing artifact that is present frequently (see *Technical Details* for more information).

Another entry you might find is that the variant was removed due to insufficient coverage:

Listing 10: **MOMA Insufficient Coverage Filter**

```
2020-03-23 08:58:50  [ NOTE ]:   Removing variant CREBBP:chr16:3900701:C:T because␣
↪Alt reads are less than 25 (alt reads: 20).
```

Or due to insufficient VAF:

Listing 11: **MOMA Insufficient VAF Filter**

```
2019-12-11 12:57:46  [ NOTE ]:   Removing variant PIK3CA:chr3:178916930:G:C because␣
↪VAF is <1% and this is not ctDNA assay data.
```

In both cases above, the variant was found to have either insufficient coverage or insufficient VAF and was removed from the report.

Note: Insufficient VAF, as the log message indicates, is primarily only for non-ctDNA assays. If we're running a ctDNA assay and the VAF is < 1%, the call will not be filtered out.

If you've checked the log file and there is no indication of why the variant was not present, then checking the Annovar data is the next step. In the Annovar data, there will be ~115 columns of annotation data for each call. In particular, we would be interested in looking at the following columns:

Table 6: **Important Annovar Columns for Filtered Variant Checking**

Element	Impact
Func.trunc_refGene	If the variant is in intronic or UTR region, it will be filtered out.
ExonicFunc.trunc_refGene	If the variant is synonymous, it will be filtered out.
PopFreqMax	If the variant has a value > 1%, it will be filtered out. Note that we don't use this specific column, but rather use the specific database (i.e. GnomAD) to get the population frequency data. But, this is a quick place to check if any population data is >1%, after which one can drill into which database flagged the call.

There may be some other elements of note, but these are the primary key ones to look at to understand why a variant didn't make it to the report, and how one might go about determining the cause from the intermediate files available.

3.5.2 Pipeline Errors

MOMA in general has relatively comprehensive error handling and checking code built in, which will be helpful in the event that the pipeline halts due to an error. These error messages may be specific:

```
2020-03-24 09:54:38  [ ERROR ]:  Data type "wes" does not appear to be correct. Check␣
↪the source is correct for this kind of VCF
```

In this case it's clear to see that we chose a bad option for running the pipeline and mis-identified the data source (i.e. `--source`) for he VCF as being `wes` when it's probably not based on data elements within the VCF.

In some cases, due to the complexity of all of the scripts and components of this pipeline, a more generic error message indicating that the pipeline failed and which command caused it to fail will be produced. For example, we can see from the log message below that there was a problem running Annovar:

Listing 12: **Example Pipeline Error Log Entry**

```
2020-03-24 10:02:59  [ ERROR ]:  An error has occurred while trying to annotate VCF␣
↪with Annovar
Traceback (most recent call last):
  File "/Users/simsdj/Dropbox/git_repos/mocha_oncogenic_moi_annotator/run_moma_
↪pipeline.py", line 991, in <module>
    args.keep, args.noanno, args.rave)
  File "/Users/simsdj/Dropbox/git_repos/mocha_oncogenic_moi_annotator/run_moma_
↪pipeline.py", line 851, in main
    annovar_file = run_annovar(simple_vcf, sample_name)
  File "/Users/simsdj/Dropbox/git_repos/mocha_oncogenic_moi_annotator/run_moma_
↪pipeline.py", line 266, in run_annovar
    status = run(cmd, 'annotate VCF with Annovar', silent=not verbose)
  File "/Users/simsdj/Dropbox/git_repos/mocha_oncogenic_moi_annotator/run_moma_
↪pipeline.py", line 339, in run
    raise subprocess.CalledProcessError(proc.returncode, ' '.join(proc.args))
subprocess.CalledProcessError: Command '/Users/simsdj/Dropbox/git_repos/mocha_
↪oncogenic_moi_annotator/scripts/annovar_wrapper.sh /Users/simsdj/Dropbox/git_repos/
↪mocha_oncogenic_moi_annotator/test/ocav3/MSN30015_v1_out/oca_simple.vcf /Users/
↪simsdj/Dropbox/git_repos/mocha_oncogenic_moi_annotator/test/ocav3/MSN30015_v1_out/
↪MSN30015_v1' returned non-zero exit status 2.
```

(continues on next page)

(continued from previous page)

What we can see here is that there was an error while trying to run Annovar from the main error line at the top. Scanning through the stack trace, we can see which specific command within the pipeline threw the error, including the arguments passed to the command. In this case, the erroneous command is:

```
Users/simsdj/Dropbox/git_repos/mocha_oncogenic_moi_annotator/scripts/annovar_wrapper.
↪sh /Users/simsdj/Dropbox/git_repos/mocha_oncogenic_moi_annotator/test/ocav3/
↪MSN30015_v1_out/oca_simple.vcf /Users/simsdj/Dropbox/git_repos/mocha_oncogenic_moi_
↪annotator/test/ocav3/MSN30015_v1_out/MSN30015_v1
```

To figure out what might be going wrong, the easiest thing to do is to try to execute that command on the commandline to see what the error the individual script throws. Executing that command we can clearly see what the problem was:

```
Error: the required database file /tmp/hg19_trunc_refGene.txt does not exist.
```

So, it looks like the path for one of the Annovar database elements is wrong. This error was forced as an example (the paths should be correct for this by default), but it does demonstrate that all of the sub-scripts within the package have their own error handling messages, and it becomes easy to figure out what might have gone awry during the run.

3.5.3 Other Troubleshooting Strategies

There are probably not many cases that can't be investigated and solved via one of the two mechanisms above. However, in the event that the problem persists and / or a cause is still not clear, there are an additional two features that can help.

3.5.3.1 Verbose Logging

If one were to run with the `--Verbose` option to MOMA, one would find that there are a lot more messages and debugging information displayed. By comparison with the *Example MOMA Log File Output* above, we can see that a log file generated with the `--Verbose` is much more detailed.

Note: Not showing the full log file here as it is quite long.

From this extended log file, it is possible to see why a call may have been eliminated from the report or why there may have been some odd piece of data that can not be otherwise explained.

3.5.3.2 Keep All Output

By default, many of the scratch and temporary files that are generated from the various scripts in the pipeline are cleaned up at the end of the run. These files are generally not useful and clutter the output. However, in the case of troubleshooting, it is possible that one or more of the intermediate files can help track down an issue. By running the pipeline with the `--keep` option, no cleanup is done at the end of the run, and all intermediate files, no matter how trivial, are kept.

These files can be helpful, for example, to trace a variant's path through MOMA to pinpoint where and why it was filtered out. Generally, this will be due to a rule or filter designed specifically to remove the call. However, if there is a bug (e.g. a type or unexpected variant annotation term from Annovar), it can be seen in these data, and the root cause can be determined and solved.

3.5.4 Summary

MOMA is intended to have a rich set of help documents, debugging and error messages, log file entry information, and temporary / intermediate files that can be used to ensure the data is accurate and things are being reported as intended. Further, if there are unexpected results or pipeline bugs / issues - whether they be code errors or user errors - MOMA will have the data necessarily to adequately troubleshoot and solve the issue.

CHAPTER

FOUR

TECHNICAL DETAILS

This section to be written.

INDEX