Chapter 1

VMT Reference Manual

1.1 Introduction

This program is an easy-to-use tool to identify causal variants for Mendelian diseases using NGS data. It can handle the data for thousands of individuals. Users can obtain the results with a few command line.

Web resource: please visit www.statgen.us/VMT/view for more information including download & installation instructions, software updates and supports from VMT user forum.

1.2 VMT Program Command Options

To display the command interface:

```
vmt -h
```

VMT interface

```
usage: vmt [-h] [-v]
       {init,import,annotate,filter,choose,output,show,remove,compare} ...
A Mendelian diseases causal variants identification tool for next-generation sequencing data.
optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
subcommands:
  {init,import,annotate,filter,choose,output,show,remove,compare}
    init                Create a new project
    import              Import variants and sample genotype from vcf files.
    annotate            Download or build an annotation database and use it.
    filter              Filter selected variants according to specified
                        conditions and save to a table.
    choose              Choose possible causal variants according to certain
                        MOI.
    output              Output variants and annotation fields.
    show                Display content of a project.
    remove              Remove content of a project.
    compare             Compare variants in variant tables.
Use 'vmt subcommand -h' for details about each command.
Copyright (c) 2015 Hang Dai <hang.dai@bcm.edu>, Gao Wang <wangow@gmail.com>, Di Zhang <di.zhang@bcm.edu>
Home page: www.statgen.us/VMT/view
```

1.2.1 Create project

This command is used to create a new project under the current directory. Only one project is allowed in a directory, when a new project is created, old project with any name will be overwritten.

To display the subcommand interface:

```
vmt init -h
```

VMT init interface

```
usage: vmt init [-h] project
Create a new project in the current directory. If another project already
exists in this directory, that project will be overwritten.
positional arguments:
  project     Project name. This will create a new *.proj file under the
              current directory. Only one project is allowed in a directory.
optional arguments:
  -h, --help  show this help message and exit
```

- `project` **[required positional argument]**

The name of the project.

Two files `project.log` and `project.proj` will be created after the command.

1.2.2 Import data

This command is used to import variants and sample information to the project.

To display the subcommand interface:

```
vmt import -h
```

VMT import interface

```
usage: vmt import [-h] [--format format_file] [--build build] --sample_info
                  sample_information_file
                  vcf_files [vcf_files ...]
Import variants and sample genotype from vcf files. Import sample information.
VMT also automatically annotate variants using ANNOVAR, refGene, dbNSFP and
ExAC.
positional arguments:
  vcf_files             A list of vcf files to be imported. Gzipped files are
                        acceptable.
optional arguments:
  -h, --help            show this help message and exit
  --format format_file  Format of input vcf file. If unspecified, the default
                        vcf.fmt will be used.
  --build build         Reference genome build version. If unspecified, the
                        default version hg19 will be used.
  --sample_info sample_information_file
                        Tab or space delimited file containing sample
                        information. This is required for VMT. The first
                        column must be 'sample_name', with sample names in the
                        vcf file. The file must contain another three columns:
                        'family_id', with the family ID of each sample;
                        'affection_status', either 2 (affected) or 1
                        (unaffected); 'unaffected_parental_genotypes', either
                        'yes', such as that in a nuclear family with genotypes
                        available for unaffected parents and affected child
                        (children), or 'no'. The file may contain other
                        information, such as phenotype information for
                        samples.
```

- `vcf_files` **[required positional argument]**

`vcf` files or gzipped or bgzipped `vcf` files are acceptable.

- `--format` **[default to `vcf.fmt` prepared for user]**

Format of input vcf file. If unspecified, the default vcf.fmt will be used. For most users, this optional argument can be left unspecified.

- `--build` **[default to hg19]**

Reference genome build version. If unspecified, the default version hg19 will be used.

- `--sample_info` **[required]**

Tab or space delimited file containing sample information. The first column must be 'sample_name', with sample names in the vcf file. The file must contain another three columns: 'family_id', with the family ID of each sample; 'affection_status', either 2 (affected) or 1 (unaffected); 'unaffected_parental_genotypes', either 'yes', such as that in a nuclear family with genotypes available for unaffected parents and affected child (children), or 'no'. The file may contain other information, such as phenotype information for samples.

Below is an example for such file. Family HI1 has one affected individual sequenced; family HI2 has 3 affected individuals sequenced; while family HI3 has 2 affected children and their unaffected parents sequenced:

sample_info.txt

```
sample_name        family_id        affection_status        unaffected_parental_genotypes
100000        HI1        2        no
100001        HI2        2        no
100002        HI2        2        no
100003        HI2        2        no
100004        HI3        1        yes
100005        HI3        1        yes
100006        HI3        2        yes
100007        HI3        2        yes
```

1.2.3 Annotation

This command is used to annotate variants.

To display the subcommand interface:

```
vmt annotate -h
```

VMT annotate interface

```
usage: vmt annotate [-h]
                    [--database database | --source source_files [source_files ...]]
                    [--format description_file]
                    [--linked_by fields [fields ...]] [--vmt_annotation]
Download an annotation database, or build it from source file; then link it to
the project.
optional arguments:
  -h, --help            show this help message and exit
  --database database   Use an annotation database (*.DB or *.DB.gz) if
                        available. If the database is not available, download
```

```
                      or build the database if a description file (*.ann) is
                      available. If the description file is not available,
                      download a description file and the corresponding
                      database. If the corresponding database still cannot
                      be downloaded, try to download the source file and
                      build the database. For available database, use 'vmt
                      show annotations' for details. This argument is
                      mutually exclusive with '--source'.
  --source source_files [source_files ...]
                      A list of source files to be used to build a local
                      annotation database. This argument must be used in
                      combination with '--format' argument, which specifying
                      the description file (*.ann). If specified, local
                      annotation database will be compiled from these source
                      files and used. This argument is mutually exclusive
                      with '--database'.
  --format description_file
                      Description file (*.ann). This argument must be used
                      in combination with '--source' argument, which
                      specifying a list of source files to be used to build
                      a local annotation database.
  --linked_by fields [fields ...]
                      A list of fields that are used to link the annotation
                      database to tables in the existing project. This
                      parameter is required only for 'field' type of
                      annotation databases linking to fields of existing
                      tables.
  --vmt_annotation    Annotate variants using VMT annotation if the option
                      string '--vmt_annotation' is present. A field
                      'VMT_annotation' will be added. For non-synonymous and
                      splice-site SNVs, 'VMT_annotation' will be
                      'damaging_SNV' if 20 percent of available prediction
                      algorithms among 9 prediction algorithms (SIFT,
                      Polyphen2, LRT, MutationTaster, MutationAssessor,
                      FATHMM, MetaLR, MetaSVM, PROVEAN) consider the SNV as
                      damaging; for INDELs, 'VMT_annotation' will be
                      'damaging_by_MT' if MutationTaster considers it as
                      damaging or "nondamaging_by_MT" if MutationTaster
                      considers it as non-damaging. In order to let VMT
                      annotate INDELs, first a vcf file with name
                      'indel.vcf' in the project directory should be
                      uploaded to MutationTaster QueryEngine
                      <http://www.mutationtaster.org/StartQueryEngine.html>;
                      then follow the steps instructed by MutationTaster and
                      download the '*.zip' file; finally retrieve the
                      '*.tsv' file in the '*.zip' file and rename it to
                      'indel.tsv' and put it into the project directory. If
                      no file with name 'indel.tsv' is provided, or the file
                      with name 'indel.tsv' is not downloaded from
                      MutationTaster, then the INDELs will not be annotated
                      by VMT annotation.
```

- `--database` **[optional]**

Use an annotation database. For available database, use 'vmt show annotations' for details. For the first time VMT user, there is no available database stored on disk, so it may take some time to download the database. This argument is mutually exclusive with '–source'.

- `--source` **[optional]**

Specify a list of source files to be used to compile a local annotation database. A description file `*.ann` must also be specified by '–format' argument. This argument is mutually exclusive with '–database'.

- `--format` **[optional]**

This argument, specifying description file `*.ann`, must be used together with '–source' argument, which specifying a list of source files to be used to build a local annotation database.

- `--linked_by` **[optional]**

A list of fields that are used to link the annotation database to tables in the existing project. This parameter is required only for 'field' type of annotation databases linking to fields of existing tables.

- `--vmt_annotation` **[default to disabled]**

When this switch is turned on, variants will be annotated by using VMT annotation. A field 'VMT_annotation' will be added.

For non-synonymous and splice-site SNVs, 'VMT_annotation' will be 'damaging_SNV' if 20 percent of available prediction algorithms among 9 prediction algorithms (SIFT, Polyphen2, LRT, MutationTaster, MutationAssessor, FATHMM, MetaLR, MetaSVM, PROVEAN) consider the SNV as damaging.

For INDELs, 'VMT_annotation' will be 'damaging_by_MT' if MutationTaster considers it as damaging or 'nondamaging_by_MT' if MutationTaster considers it as non-damaging.

How to annotate INDELs

In order to let VMT annotate INDELs, first a vcf file with name `indel.vcf` in the project directory should be uploaded to MutationTaster QueryEngine [1]; then follow the steps instructed by MutationTaster and download the `*.zip` file; finally retrieve the `*.tsv` file in the `*.zip` file and rename it to `indel.tsv` and put it into the project directory. If no file with name `indel.tsv` is provided, or the file with name `indel.tsv` is not downloaded from MutationTaster, then the INDELs will not be annotated by VMT annotation.

1.2.4 Filtering

This command is used to filter selected variants according to specified conditions, which are arbitrary SQL expressions [2] that involves fields in the project.

To display the subcommand interface:

```
vmt filter -h
```

VMT filter interface

```
usage: vmt filter [-h] --to_table table [--pop_maf [conditions]]
                  [--pos [conditions]] [--damaging [conditions]]
                  [--other [conditions]] [--filter_pass]
                  [--samples [conditions]]
                  table
Filter selected variants according to specified conditions: properties
(variant and annotation fields) and membership (samples) of variants.
Conditions are arbitrary SQL expressions that involves fields in the project.
Please refer <http://www.sqlite.org/lang_expr.html> for syntax.
positional arguments:
  table                 Select a variant table.
optional arguments:
  -h, --help            show this help message and exit
  --to_table table      Save to a variant table.
According to properties:
```

```
  Filter variants according to annotation fields. For available fields to
  use, type 'vmt show fields'.
  --pop_maf [conditions]
                        Filter variants according to population MAF, such as
                        that in 1000G, ESP and ExAC. If the option string '--
                        pop_maf' is present but not followed by a command line
                        argument, the default threshold of ExAC MAF<=0.001 in all
                        populations will be used.
  --pos [conditions]    Filter variants according to position. This is often
                        useful when disease loci were identified by linkage
                        analysis and homozygosity mapping.
  --damaging [conditions]
                        Filter variants according to bioinformatics tools. If
                        the option string '--damaging' is present but not
                        followed by a command line argument, the default
                        setting, damaging considered by VMT_annotation and
                        CADD PHRED>=10, will be used. Under such
                        circumstances, the variants must already be annotated
                        by VMT_annotation. If not, use 'vmt annotate
                        --vmt_annotation' to annotate. Use 'vmt annotate -h'
                        for details of VMT_annotation.
  --other [conditions]  Filter variants according to other conditions.
  --filter_pass         Filter variants according to 'FILTER' column in vcf
                        file. If the option string '--filter_pass' is present,
                        then variants that do not pass filter will be filtered
                        out.
According to membership:
  Select variants according to sample genotypes. For available fields to
  use, type 'vmt show samples'.
  --samples [conditions]
                        Only variants based on certain sample genotypes will
                        be select.
```

- `from_table` **[required positional argument]**

Select a variant table. If not sure the table names, use 'vmt show tables' to see table names.

- `--to_table` **[required]**

Output variants to a table.

- `--pop_maf` **[optional]**

Filter variants according to population MAF, such as that in 1000G, ESP and ExAC. If the option string '–pop_maf' is present but not followed by a command line argument, the default threshold of ExAC MAF$\leqslant$0.001 in all populations will be used.

Filter according to population MAF

e.g. `vmt filter variant --pop_maf 'ExAC.All_MAF<=0.005' --to_table variant_rare` If not sure which fields to use, use '`vmt show fields`' to see fields names.

- `--pos` **[optional]**

Filter variants according to position. This is often useful when disease loci were identified by linkage analysis and homozygosity mapping.

Filter according to position

e.g. `vmt filter variant_rare --pos ''chr='6' and pos>1000000 and pos<2000000'' --to_table variant_rare_locus`

- `--damaging` **[optional]**

Filter variants according to bioinformatics tools. If the option string '–damaging' is present but not followed by a command line argument, the default setting, damaging considered by VMT_annotation and CADD PHRED$\leqslant$10, will be used. Under such circumstances, the variants must already be annotated by VMT_annotation. If not, use 'vmt annotate –vmt_annotation' to annotate. Use 'vmt annotate -h' for details of VMT_annotation.

Filter according to deleteriousness

e.g. `vmt filter variant_rare_locus --damaging ''dbNSFP.SIFT_pred='D' '' --to_table variant_rare_locus_damaging` If not sure which fields to use, use '`vmt show fields`' to see fields names.

- `--other` **[optional]**

Filter variants according to other conditions.

Filter according to various conditions

e.g. We want to select all the GJB2 variants in the data: `vmt filter variant --damaging ''refGene.name2='GJB2' '' --to_table variant_GJB2`. If not sure which fields to use, use '`vmt show fields`' to see fields names.

- `--filter_pass` **[default to disable]**

Filter variants according to 'FILTER' column in vcf file. When this switch is turned on, then variants that do not pass filter will be filtered out.

- `--samples` **[optional]**

This is used to select variants based on certain sample genotypes will be select.

Filter according to sample genotypes

e.g. We want to select all the variants in samples of family HI2: `vmt filter variant --samples ''family_id='HI2' '' --to_table variant_HI2` If not sure which fields to use, use '`vmt show samples`' to see fields names.

1.2.5 Choose variants according to MOI

This command is used to choose possible causal variants for families according to MOI specified by users. After running the command, files containing potential causal variants, with name 'family_id_table_moi.tsv' can be found in the directory.

To display the subcommand interface:

```
vmt choose -h
```

VMT choose interface

```
usage: vmt choose [-h] --family_id file --moi {AD,AR,De_Novo} [--multihit]
                  table
Choose possible causal variants for families according to MOI specified by
users, also output variants.
positional arguments:
  table                 Select a variant table where possible causal variants
                        will be chosen from.
optional arguments:
  -h, --help            show this help message and exit
  --family_id file      File containing ID of families to be analyzed. The
                        file has no header, each line can only contain the
                        family_id.
  --moi {AD,AR,De_Novo}
                        Mode of inheritance. Can be chosen from 'AD', 'AR' and
                        'De_Novo'.
  --multihit            If the option string '--multihit' is present, a
                        'candidate_genes.tsv' file will be generated, showing
                        the number of times a gene were 'hit', that is, in how
                        many families the variants were identified according
                        to certain MOI in a gene. The file also shows the
                        families in which the gene were 'hit'. This is useful
                        when multiple families with same or similar phenotypes
                        were analyzed at a time, or when the disease has low
                        genetic heterogeneity.
```

- `table` **[required positional argument]**

Select a variant table where possible causal variants will be chosen from.

- `--family_id` **[required]**

File containing ID of families to be analyzed. The file has no header, each line can only contain the family_id. Below is an example:

family_id.txt

```
HI1
HI2
HI3
```

- `--moi` **[required]**

Mode of inheritance. Can be chosen from 'AD', 'AR' and 'De_Novo'. 'AD' will select shared heterozygous variants among affected individuals in the family; 'De_Novo' will select De Novo variants when genotypes of unaffected parents are available; 'AR' will select shared homozygous variants and potential shared compound heterozygous variants among affected individuals in the family, when genotypes of unaffected parents are available, segregation test will also be performed.

- `--multihit` **[default to disable]**

If this switch is turned on, a `candidate_genes.tsv` file will be generated, showing the number of times a gene were 'hit', that is, in how many families the variants were identified according to certain MOI in a gene. This is useful when multiple families with same or similar phenotypes were analyzed at a time, or when the disease has low genetic heterogeneity. Below is an example of `candidate_genes.tsv`:

candidate_genes.tsv

```
gene        family_id        occurence_number
GJB2        HI1, HI2, HI3        3
ADCY1        HI2, HI3        2
GJB6        HI1        1
```

1.2.6 Other utilities

There are several other utilities to help user. Please use commands below to see the detailed descriptions:

```
vmt output -h
```

VMT output interface

```
usage: vmt output [-h] [--header header [header ...]] --fields fields
                  [fields ...] [--group_by fields [fields ...]]
                  [--delimiter delimiter] [--order_by fields [fields ...]]
                  table
Output variants and annotation fields in specified format.
positional arguments:
  table                 Output variants in selected table.
optional arguments:
  -h, --help            show this help message and exit
  --header header [header ...]
                        A complete header or a list of names that will be
                        joined by a delimiter specified by '--delimiter'. If
                        unspecified, a default header will be derived from
                        field names.
  --fields fields [fields ...]
                        A list of fields that will be outputted.
  --group_by fields [fields ...]
                        SQL aggregating functions
                        <http://www.sqlite.org/lang_aggfunc.html> can be used
                        to output summary statistics of fields. For example,
                        you can use function count(*) to count the number of
                        records, These operations can be applied to groups of
                        variants defined by option --group_by. e.g. "vmt
                        output variant --fields chr 'count()' --group_by chr"
                        shows how many variants are there on each chromosome.
                        e.g. "vmt output variant --fields refGene.name2
                        'count()' --group_by refGene.name2" shows how many
                        variants are there in each gene.
  --delimiter delimiter
                        Delimiter use to separate columns of output. Default
                        is '\t', for a csv output, use ','
  --order_by fields [fields ...]
                        Order output by specified fields in ascending order,
                        or descending order if field name is followed by DESC
                        (e.g. --order_by 'pos DESC').
```

```
vmt compare -h
```

VMT compare interface

```
usage: vmt compare [-h] [--union table [table ...]]
                   [--intersection table [table ...]]
                   [--difference table [table ...]]
                   [--expression expression [expression ...]]
                   tables [tables ...]
Get the difference, intersection and union of variant tables.
positional arguments:
  tables                Variant tables to compare. Wildcard characters * and ?
                        can be used to specify multiple tables.
optional arguments:
  -h, --help            show this help message and exit
  --union table [table ...]
                        Save variants with TYPE to a table with name specified
```

```
                        here, in the TYPE of any of the tables (T1 | T2 | T3
                        ...). An optional message can be added to describe the
                        table. (e.g. vmt compare table1 table2 table3 --union
                        table4 'variants in any of table1, table2 and table3')
  --intersection table [table ...]
                        Save variants with TYPE to a table with name specified
                        here, in the TYPE of all the tables (T1 & T2 & T3
                        ...). An optional message can be added to describe the
                        table. (e.g. vmt compare table1 table2 table3
                        --intersection table4 'variants in all of table1,
                        table2 and table3')
  --difference table [table ...]
                        Save variants with TYPE to a table with name specified
                        here, in the TYPE of the first, but not in the TYPE of
                        others (T1 - T2 - T3...). An optional message can be
                        added to describe the table. (e.g. vmt compare table1
                        table2 table3 --difference table4 'variants in table1
                        but not in table2 or table3')
  --expression expression [expression ...]
                        Evaluate a set expression with table names. Operators
                        | (or), & (and), - (difference) and ^ (A or B but not
                        both) are allowed. The variants will be saved to a
                        table with name assigned before the expression. (e.g.
                        --expression 'E=(A|D)-(B&C)'). This is commonly useful
                        when complex comparisons between tables are needed.
```

```
vmt show -h
```

VMT show interface

```
usage: vmt show [-h] [--item items [items ...]]
                {project,tables,table,samples,fields,annotations,annotation,phenotypes,formats,format}
Output information of all system and project related items, such as tables,
samples, phenotypes, annotation databases and fields.
positional arguments:
  {project,tables,table,samples,fields,annotations,annotation,phenotypes,formats,format}
                        Type of information to display, which can be 'project'
                        for summary of a project, 'tables' for all variant
                        tables, 'table TBL' for details of a specific table
                        TBL, 'samples [COND]' for sample name, files from
                        which samples are imported, and associated phenotypes
                        of all or selected samples, 'fields' for fields from
                        variant tables and all used annotation databases,
                        'annotations' for a list of all available annotation
                        databases, 'annotation ANN' for details about
                        annotation database ANN, 'phenotypes [P1 P2...]' for
                        all or specified phenotypes of samples, 'formats' for
                        all supported import and export formats, 'format FMT'
                        for details of format FMT.
optional arguments:
  -h, --help            show this help message and exit
  --item items [items ...]
                        Items to display, which can be, for example, name of
                        table for type 'table' ('vmt show table --item
                        variant'), conditions to select samples for type
                        'samples' ('vmt show samples --item
                        "family_id='family_1'"'), name of an annotation
                        database for type 'annotation' ('vmt show annotation
                        --item dbNSFP'), a list of phenotypes for type
                        'phenotypes' ('vmt show phenotypes --item family_id'),
                        name of a format for type 'format' ('vmt show format
                        --item vcf').
```

```
vmt remove -h
```

VMT output interface

```
usage: vmt remove [-h] [--item items [items ...]] {tables,fields,annotations}
Remove from the current project various items such as variants genotypes, and
annotation fields.
```

```
positional arguments:
  {tables,fields,annotations}
                        Type of items to be removed.
optional arguments:
  -h, --help            show this help message and exit
  --item items [items ...]
                        Items to be removed, which should be, for 'tables'
                        names of one or more variant tables, for 'fields' name
                        of fields to be removed, for 'annotations' names of
                        annotation databases. Removal of annotation databases
                        will stop using these databases in the project, but
                        will not delete them from disk.
```

References

1. [MutationTaster QueryEngine]
   http://www.mutationtaster.org/StartQueryEngine.html
2. [SQL expressions]
   http://www.sqlite.org/lang_expr.html