Exomiser Documentation

Release 13.1.0

Jules Jacobsen, Damian Smedley, Peter Robinson

Jul 29, 2022

Setup and Install

1 Introduction **1**

1.1 Installation . . . 1
- 1.1.1 Software and Hardware requirements . . . 1
- 1.1.2 Pre-built Binaries . . . 2
- 1.1.3 Windows install . . . 2
- 1.1.4 Linux install . . . 2
- 1.1.5 Genomiser / REMM data . . . 3
- 1.1.6 CADD data . . . 3
- 1.1.7 Configuring the application.properties . . . 4
- 1.1.8 Troubleshooting . . . 5

1.2 Running Exomiser . . . 6
- 1.2.1 Usage . . . 6
- 1.2.2 Want help? . . . 6
- 1.2.3 Running from alternate directory . . . 6
- 1.2.4 Troubleshooting . . . 7

1.3 Input Files and Options . . . 7
- 1.3.1 Sample, vcf, assembly, ped . . . 7
- 1.3.2 Preset . . . 8
- 1.3.3 Analysis . . . 8
- 1.3.4 Output . . . 9
- 1.3.5 Batch . . . 9

1.4 Analysis/Job File Configuration . . . 10
- 1.4.1 Job . . . 10
- 1.4.2 Sample . . . 14
- 1.4.3 Preset . . . 14
- 1.4.4 Analysis . . . 15
- 1.4.5 Output options . . . 21

1.5 Structural Variant Prioritisation . . . 23
- 1.5.1 Prioritisation Overview . . . 23
- 1.5.2 Multiple Gene Overlaps . . . 23
- 1.5.3 SV Similarity . . . 23

1.6 Interpreting the Results . . . 24
- 1.6.1 HTML . . . 24
- 1.6.2 JSON . . . 24
- 1.6.3 TSV GENES . . . 24
- 1.6.4 TSV VARIANTS . . . 25

1.6.5 VCF . . . 26
1.7 ACMG Assignment . . . 27
1.7.1 Computational and Predictive Data . . . 28
1.7.2 Segregation Data . . . 28
1.7.3 De novo Data . . . 28
1.7.4 Population Data . . . 29
1.7.5 Allelic Data . . . 29
1.7.6 Phenotype . . . 29
1.7.7 Clinical . . . 29
1.7.8 Transcript Selection . . . 29
1.8 Publications . . . 32
1.8.1 Collaborations . . . 33

CHAPTER 1

Introduction

The Exomiser is a Java program, developed as a collaboration between members of the Monarch Initiative for finding potential disease-causing variants in whole-exome or whole-genome sequencing data. It is available for use by all as an offline command-line tool, or on the web as a demo application/teaching tool.

It requires a VCF file and a set of phenotypes encoded using the Human Phenotype Ontology (HPO) from which it will annotate, filter and prioritise likely causative variants. The program does this based on user-defined criteria such as a variant's predicted pathogenicity, frequency of occurrence in a population and also how closely the given phenotype matches the known phenotype of diseased genes from human and model organism data.

The functional annotation of variants is handled by the Jannovar library and uses any of UCSC, RefSeq or Ensembl KnownGene transcript definitions and hg19 or hg38 genomic coordinates.

Variants are prioritised according to user-defined criteria on variant frequency, pathogenicity, quality, inheritance pattern, and model organism phenotype data. Predicted pathogenicity data is extracted from the dbNSFP resource. Variant frequency data is taken from the 1000 Genomes, ESP, TOPMed, UK10K, ExAC and gnomAD datasets. Subsets of these frequency and pathogenicity data can be defined to further tune the analysis. Cross-species phenotype comparisons come from our PhenoDigm tool powered by the OWLTools OWLSim algorithm.

1.1 Installation

1.1.1 Software and Hardware requirements

- Minimum 8/16GB RAM (For an exome analysis of a 30,000 variant sample 4GB RAM should suffice. For a genome analysis of a 4,400,000 variant sample 12GB RAM should suffice.)
- Any 64-bit operating system
- Java 11 or above
- At least 100GB free disk space (SSD preferred for best performance)
- An internet connection is not required to run the Exomiser, although network access will be required if accessing a networked database (optional).

- By default the Exomiser is completely self-contained and is able to run on standard consumer laptops.

1.1.2 Pre-built Binaries

Note: This is the recommended way of installing for normal users.

Pre-built binaries are available from The Monarch Initiative or from the Exomiser repository on GitHub.

Exomiser requires 2-3 data files to be available as well - one for the phenotype data and one for each genome assembly required.

1.1.3 Windows install

1. Install 7-Zip for unzipping the archive files. The built-in archiving software has issues extracting the zip files.
2. Download the data and distribution files from https://data.monarchinitiative.org/exomiser/latest
3. Extract the distribution files by right-clicking exomiser-cli-13.1.0-distribution.zip and selecting 7-Zip > Extract Here
4. Extract the data files (e.g. 2109_phenotype.zip, 2109_hg19.zip) by right-clicking the archive and selecting 7-Zip > Extract files... into the exomiser data directory. By default exomiser expects this to be 'exomiser-cli-13.1.0/data', but this can be changed in the `application.properties`
5. cd exomiser-cli-13.1.0
6. java -Xmx4g -jar exomiser-cli-13.1.0.jar –analysis examples/test-analysis-exome.yml

1.1.4 Linux install

The following shell script should work-

```
# download the distribution (won't take long)
wget https://data.monarchinitiative.org/exomiser/latest/exomiser-cli-13.1.0-
↪distribution.zip
# download the data (this is ~80GB and will take a while). If you only␣
↪require a single assembly, only download the relevant file.
wget https://data.monarchinitiative.org/exomiser/latest/2202_hg19.zip
wget https://data.monarchinitiative.org/exomiser/latest/2202_hg38.zip
wget https://data.monarchinitiative.org/exomiser/latest/2202_phenotype.zip

# unzip the distribution and data files - this will create a directory called␣
↪'exomiser-cli-13.1.0' in the current working directory
unzip exomiser-cli-13.1.0-distribution.zip
unzip 2202_*.zip -d exomiser-cli-13.1.0/data

# Check the application.properties are pointing to the correct versions
# exomiser.hg19.data-version=2202
# exomiser.hg38.data-version=2202
# exomiser.phenotype.data-version=2202

# run a test exome analysis
cd exomiser-cli-13.1.0
```

```
java -Xmx4g -jar exomiser-cli-13.1.0.jar --analysis examples/test-analysis-
↪exome.yml
```

This script will download, verify and extract the exomiser files and then run the analysis contained in the file 'test-analysis-exome.yml' from the examples sub-directory. This contains a known pathogenic missense variant in the FGFR2 gene.

1.1.5 Genomiser / REMM data

In order to run the Genomiser you will also need to download the REMM data file from [here](https://zenodo.org/record/4768448). Once downloaded you'll need to add the path to the ReMM.v0.3.1.tsv.gz file to the `application.properties` file. For example if you downloaded the file to the exomiser data directory you could add the entry like this:

```
exomiser.hg19.remm-path=${exomiser.hg19.data-directory}/ReMM.v0.3.1.tsv.gz
```

If this step is omitted, the application will throw and error and stop any analysis which defines `REMM` in the `pathogenicitySources` section of an analysis yml file.

Having done this, run the analysis like this:

```
java -Xmx6g -jar exomiser-cli-13.1.0.jar --analysis examples/NA19722_601952_
↪AUTOSOMAL_RECESSIVE_POMP_13_29233225_5UTR_38.yml
```

This is an analysis for an autosomal recessive 5'UTR variant located in POMP gene on chromosome 13. The phenotype HPO terms are taken from the clinical synopsis of OMIM #601952 (http://www.omim.org/clinicalSynopsis/601952)

1.1.6 CADD data

In order to use CADD you will need to download the CADD data files separately. These can be accessed from https://cadd.gs.washington.edu/download. Exomiser only requires the file with the score in, not the full annotations. For example, in release v1.4 Exomiser requires both the files *All possible SNVs of GRCh38/hg38* and *80M InDels to initiate a local setup*. Each genome assembly will require the relevant files. The direct links from the US site are shown below and are correct at the time of writing.

```
wget https://krishna.gs.washington.edu/download/CADD/v1.4/GRCh38/whole_genome_SNVs.
↪tsv.gz
wget https://krishna.gs.washington.edu/download/CADD/v1.4/GRCh38/whole_genome_SNVs.
↪tsv.gz.tbi
wget https://krishna.gs.washington.edu/download/CADD/v1.4/GRCh38/InDels.tsv.gz
wget https://krishna.gs.washington.edu/download/CADD/v1.4/GRCh38/InDels.tsv.gz.tbi

wget https://krishna.gs.washington.edu/download/CADD/v1.4/GRCh37/whole_genome_SNVs.
↪tsv.gz
wget https://krishna.gs.washington.edu/download/CADD/v1.4/GRCh37/whole_genome_SNVs.
↪tsv.gz.tbi
wget https://krishna.gs.washington.edu/download/CADD/v1.4/GRCh37/InDels.tsv.gz
wget https://krishna.gs.washington.edu/download/CADD/v1.4/GRCh37/InDels.tsv.gz.tbi
```

Enable Exomiser to use CADD by altering the `application.properties` file to enable these lines and ensure the `cadd.version` property matches the version you downloaded.

```
cadd.version=1.4

exomiser.hg19.cadd-snv-path=${exomiser.data-directory}/cadd/${cadd.version}/hg19/
↪whole_genome_SNVs.tsv.gz
```

(continues on next page)

(continued from previous page)

```
exomiser.hg19.cadd-in-del-path=${exomiser.data-directory}/cadd/${cadd.version}/hg19/
↪InDels.tsv.gz

# and/or for hg38
exomiser.hg38.cadd-snv-path=${exomiser.data-directory}/cadd/${cadd.version}/whole_
↪genome_SNVs.tsv.gz
exomiser.hg38.cadd-in-del-path=${exomiser.data-directory}/cadd/${cadd.version}/InDels.
↪tsv.gz
```

Exomiser will expect the tabix index `.tbi` file to be present in the same directory as the `.tsv.gz` files. To use CADD scores in an analysis, the `pathogenicitySources` should contain the `CADD` property

```
#Possible pathogenicitySources: POLYPHEN, MUTATION_TASTER, SIFT, CADD, REMM
#REMM is trained on non-coding regulatory regions
#*WARNING* if you enable CADD or REMM ensure that you have downloaded and installed␣
↪the CADD/REMM tabix files
#and updated their location in the application.properties. Exomiser will not run␣
↪without this.
pathogenicitySources: [POLYPHEN, MUTATION_TASTER, SIFT, CADD]
```

1.1.7 Configuring the application.properties

Once you have downloaded and unzipped all the data, you will need to edit the exomiser-cli-13.1.0/application.properties file located in the main exomiser-cli directory. This file contains a lot of comments for optional data and assemblies.

If you want to run Exomiser using data from a different release directory edit the line in `application.properties`:

```
exomiser.data-directory=
```

with

```
exomiser.data-directory=/full/path/to/alternative/data/directory
```

For example, assuming you unzipped the contents of the *2202_hg38.zip* data file into */data/exomiser-data*:

```
exomiser.data-directory=/data/exomiser-data
```

where the contents of *exomiser-data* looks something like this:

```
$ tree -L 1 /data/exomiser-data/
    /data/exomiser-data/
    ├── 2202_hg19
    ├── 2202_hg38
    ├── 2202_phenotype
    ├── cadd
    └── remm
```

By default Exomiser will look for data located in the exomiser-cli-13.1.0/data directory.

After defining the a *exomiser.data-directory*, a minimal setup for exome analysis using GRCh37/hg19 would only require the `application.properties` to contain this:

```
### hg19 assembly ###
exomiser.hg19.data-version=2109
exomiser.hg19.variant-white-list-path=2109_hg19_clinvar_whitelist.tsv.gz

### phenotypes ###
exomiser.phenotype.data-version=2109
```

For a GRCh38/hg38 only setup:

```
### hg38 assembly ###
exomiser.hg38.data-version=2109
exomiser.hg38.variant-white-list-path=2109_hg38_clinvar_whitelist.tsv.gz

### phenotypes ###
exomiser.phenotype.data-version=2109
```

Or an install supporting both assemblies:

```
### hg19 assembly ###
exomiser.hg19.data-version=2109
exomiser.hg19.variant-white-list-path=2109_hg19_clinvar_whitelist.tsv.gz

### hg38 assembly ###
exomiser.hg38.data-version=2109
exomiser.hg38.variant-white-list-path=2109_hg38_clinvar_whitelist.tsv.gz

### phenotypes ###
exomiser.phenotype.data-version=2109
```

n.b. each assembly will require approximately 1GB RAM to load. Attempting to analyse a VCF called using an unsupported/unloaded assembly data will result in an unrecoverable error being thrown.

Notice here that we are loading a whitelist created from ClinVar data. Exomiser will consider any variant on the whitelist to be maximally pathogenic, regardless of the underlying data (*e.g.* variant effect, allele frequency, predicted pathogenicity) and always included these in the results.

1.1.8 Troubleshooting

Zip file reported as too big or corrupted

If, when running 'unzip exomiser-cli-13.1.0-distribution.zip', you see the following:

```
error:  Zip file too big (greater than 4294959102 bytes)
Archive:  exomiser-cli-13.1.0-distribution.zip
warning [exomiser-cli-13.1.0-distribution.zip]:  9940454202 extra bytes at␣
↪beginning or within zipfile
  (attempting to process anyway)
error [exomiser-cli-13.1.0-distribution.zip]:  start of central directory not␣
↪found;
  zipfile corrupt.
  (please check that you have transferred or created the zipfile in the
  appropriate BINARY mode and that you have compiled UnZip properly)
```

Check that your unzip version was compiled with LARGE_FILE_SUPPORT and ZIP64_SUPPORT. This is standard with UnZip 6.00 and can be checked by typing:

```
unzip -version
```

This shouldn't be an issue with more recent linux distributions.

1.2 Running Exomiser

1.2.1 Usage

In general we recommend running Exomiser from the install directory. A single sample exome analysis can be run using the following command:

```
# run a test exome analysis
cd exomiser-cli-13.1.0
java -jar exomiser-cli-13.1.0.jar --sample examples/pfeiffer-phenopacket.yml -
↪-vcf examples/Pfeiffer.vcf.gz --assembly hg19
```

This command prioritises variants from the input VCF file, called against the GRCh37/hg19 reference assembly in the context of the sample phenotypes encoded using Human Phenotype Ontology terms contained in the Phenopacket file.

Running a multi-sample VCF for trios also requires a PED file. e.g.

```
# run a test exome family analysis
cd exomiser-cli-13.1.0
java -jar exomiser-cli-13.1.0.jar --sample examples/pfeiffer-family.yml --vcf␣
↪examples/Pfeiffer-quartet.vcf.gz --assembly hg19 --ped examples/Pfeiffer-
↪quartet.ped
```

By default there will be two output files written to the `results` directory using the same filename as the input VCF file but with `_exomiser` appended before a file extension of `.json` or `.html` e.g.

```
ls results/Pfeiffer*
results/Pfeiffer_exomiser.html  results/Pfeiffer_exomiser.json
```

The HTML file is for human use, whilst the JSON file is better read by machines, for instance by using jq. Details on how to interpret the output can be found in the *Interpreting the Results* section.

The `examples` directory contains a selection of single sample exome and genome analysis files, a multisample (family) analysis with an associated pedigree in PED format, and the respective Phenopacket representations of the proband or family.

1.2.2 Want help?

```
java -jar exomiser-cli-13.1.0.jar --help
```

1.2.3 Running from alternate directory

If you're running the exomiser from a different directory to the one the exomiser-cli-13.1.0.jar file is located, you will need to specify the path to the `application.properties` file in the start-up command. For example:

```
java -Xmx4g -jar /full/path/to/your/exomiser-cli/directory/exomiser-cli-13.
↪1.0.jar --analysis /full/path/to/your/exomiser-cli/directory/examples/test-
↪analysis-exome.yml --spring.config.location=/full/path/to/your/exomiser-cli/
↪directory/application.properties
```

n.b. the `spring.config.location` command **must be the last argument in the input commands!**

1.2.4 Troubleshooting

java.lang.UnsupportedClassVersionError:

If you get the following error message:

```
Exception in thread "main" java.lang.UnsupportedClassVersionError:
org/monarchinitiative/exomiser/cli/Main : Unsupported major.minor version
```

or

```
Error: A JNI error has occurred, please check your installation and try again
Exception in thread "main" java.lang.UnsupportedClassVersionError: org/
↪monarchinitiative/exomiser/cli/Main has been
compiled by a more recent version of the Java Runtime (class file version 55.0), this␣
↪version of the Java Runtime
only recognizes class file versions up to 52.0
```

You are running an older unsupported version of Java. Exomiser requires java version 11 or higher. This can be checked by running:

```
$ java -version
```

You should see something like this in response:

```
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
```

Versions lower than 11 (e.g. 1.5, 1.6, 1.7, 1.8, 9, 10) will not run exomiser, so you will need to install the latest java version.

1.3 Input Files and Options

The Exomiser can be run via simply via a yaml analysis file. The extended cli capability was removed in version 10.0.0 as this was less capable than the yaml scripts and only supported hg19 exome analysis. Version 13.0.0 introduced a more flexible input using a GA4GH Phenopacket (v 1.0) for the sample data with the ability to specify the input pedigree, VCF and genome assembly independently; user-specified, preset or default analysis options and a new batch mode.

1.3.1 Sample, vcf, assembly, ped

It is recommended to provide Exomiser with the input sample as a Phenopacket. Exomiser will accept this in either JSON or YAML format. The sample is provided using the `sample` switch and the full path to the phenopacket file:

```
java -jar exomiser-cli-13.1.0.jar --sample path/to/phenopacket.json
```

Should the phenopacket either not specify a VCF file or specifies a file on another filesystem, the VCF file path can be provided/overridden using the `vcf` switch:

```
java -jar exomiser-cli-13.1.0.jar --sample path/to/phenopacket.json --vcf␣
↪path/to/genome.vcf
```

Exomiser will assume that the vcf file was called using the default assembly (GRCh37 / hg19) but this can be specified using the `assembly` switch:

```
java -jar exomiser-cli-13.1.0.jar --sample path/to/phenopacket.json --vcf␣
↪path/to/genome.vcf --assembly hg19
```

or for hg38/ GRCh38:

```
java -jar exomiser-cli-13.1.0.jar --sample path/to/phenopacket.json --vcf␣
↪path/to/genome.vcf --assembly hg38
```

Lastly, when analysing a multi-sample VCF a pedigree is required. This can be provided using a dedicated PED file. This uses the `ped` switch and a full path to the PED file:

```
java -jar exomiser-cli-13.1.0.jar --sample path/to/phenopacket.json --vcf␣
↪path/to/genome.vcf --assembly hg38 --ped path/to/pedigree.ped
```

or the pedigree, proband and family members can be provided as a phenopacket family message which can encode the pedigree.

```
java -jar exomiser-cli-13.1.0.jar --sample path/to/family.json --vcf path/to/
↪genome.vcf --assembly hg38
```

Whatever the input used it is essential that the sample names used for the proband and other family members are consistent between the pedigree and the sample identifiers used in the VCF file. Exomiser will exit with an error explaining that they do not match. Examples of these can be found in the *examples* directory of the installation.

1.3.2 Preset

If no `analysis` is provided and no preset is specified, Exomiser will default to running the `exome` preset analysis. If you want to run Genomiser, which will analyse non-coding regions of a WGS sample use `--preset genome`:

```
java -jar exomiser-cli-13.1.0.jar --sample path/to/phenopacket.json --preset␣
↪genome
```

In order to run a `genome` preset you need to first ensure that the REMM score data has been downloaded for the relevant genome assembly and is enabled in the `application.properties` see the *Genomiser / REMM data* section.

1.3.3 Analysis

Analysis files contain all possible options for running an analysis including the ability to specify variant frequency and pathogenicity data sources and the ability to tweak the order that analysis steps are performed.

See the test-analysis-exome.yml and test-analysis-genome.yml files located in the base install directory for examples. Details can be found in the *Analysis* section.

```
java -jar exomiser-cli-13.1.0.jar --analysis examples/test-analysis-exome.yml
```

These files an also be used to run full-genomes, however they will require substantially more RAM to do so. For example a 4.4 million variant analysis requires approximately 12GB RAM. However, RAM requirements can be greatly reduced by setting the analysisMode option to PASS_ONLY. This will also aid your ability to evaluate the results.

Analyses can be run in batch mode. Simply put the path to each analysis file in the batch file - one file path per line.

```
java -jar exomiser-cli-13.1.0.jar --analysis-batch examples/test-analysis-
↪batch.txt
```

1.3.4 Output

By default Exomiser will write out any result files to the exomiser-cli-13.1.0/results sub-directory of the Exomiser installation directory. Unless specified in the *output.yml* or *outputOptions* section of the analysis YAML file, Exomiser will write out a *.json* and a *.html* file. These are for machine (JSON) and human (HTML) use. The filenames will match the input VCF filename. For example

```
java -jar exomiser-cli-13.1.0.jar --sample examples/pfeiffer-phenopacket.yml -
↪-vcf path/to/manuel.vcf.gz
```

Would result in two files being output with the filename 'manuel_exomiser' and the '.json' and '.html' extensions:

```
exomiser-cli-13.1.0/results/manuel_exomiser.html
exomiser-cli-13.1.0/results/manuel_exomiser.json
```

Users requiring more control over their output can use either the `outputOptions` section of an analysis file or a specific output options yaml file. An example of this can be found in the exomiser-cli-13.1.0/examples/output-options.yml file:

```
---
outputContributingVariantsOnly: false
#numGenes options: 0 = all or specify a limit e.g. 500 for the first 500 results
numGenes: 10
minExomiserGeneScore: 0.7
#outputPrefix options: specify the path/filename without an extension and this will␣
↪be added
# according to the outputFormats option. If unspecified this will default to the␣
↪following:
# {exomiserDir}/results/input-vcf-name-exomiser-results.html
# alternatively, specify a fully qualifed path only. e.g. /users/jules/exomes/analysis
outputPrefix: /results/NA12345/NA12345
#out-format options: HTML, JSON, TSV_GENE, TSV_VARIANT, VCF (default: HTML)
outputFormats: [HTML, JSON, TSV_GENE]
```

This file is passed to Exomiser using the `--output` switch:

```
java -jar exomiser-cli-13.1.0.jar --sample examples/pfeiffer-phenopacket.yml -
↪-vcf path/to/manuel.vcf.gz --output path/to/output-options.yml
```

1.3.5 Batch

The above commands can be added to a batch file for example in the file exomiser-cli-13.1.0/examples/test-analysis-batch-commands.txt

```
#This is an example analysis batch file to be run using the --analysis-batch command
#
#Each line should specify the path of a single analysis file, either relative to the␣
↪directory the exomiser
#is being run from or the full system path. It will run any combination of exomiser␣
↪commands listed using -h or --help.
#
# Original format exomiser analysis containing all the sample and analysis information
--analysis test-analysis-exome.yml
# New preset exome analysis using a v1 phenopacket to submit the phenotype␣
↪information and adding/overriding the VCF input
--preset exome --sample pfeiffer-phenopacket.yml --vcf Pfeiffer.vcf.gz
# Using the default analysis (exome) with a v1 phenopacket containing the phenotype␣
↪information and adding/overriding the VCF input
```

(continues on next page)

(continued from previous page)

```
--sample pfeiffer-phenopacket.yml --vcf Pfeiffer.vcf.gz
# Using a user-defined analysis with a v1 phenopacket containing the phenotype␣
↪information and adding/overriding the VCF input
--analysis preset-exome-analysis.yml --sample pfeiffer-phenopacket.yml --vcf Pfeiffer.
↪vcf.gz
# Using a user-defined analysis with a v1 phenopacket containing the phenotype␣
↪information and adding/overriding the VCF input
--analysis preset-exome-analysis.yml --sample pfeiffer-phenopacket.yml --vcf Pfeiffer.
↪vcf.gz --output output-options.yml
```

then run using the `--batch` command:

```
java -jar exomiser-cli-13.1.0.jar --batch path/to/exomiser-cli-13.1.0/
↪examples/test-analysis-batch-commands.txt
```

The advantage of this is that a single command will be able to analyse many samples in far less time than starting a new JVM for each as there will be no start-up penalty after the initial start and the Java JIT compiler will be able to take advantage of a longer-running process to optimise the runtime code. For maximum throughput on a cluster consider splitting your batch jobs over multiple nodes.

1.4 Analysis/Job File Configuration

Exomiser analyses are defined using a yml format configuration file(s). Examples can be found in the unpacked exomiser-cli-13.1.0/examples/ directory. Here you will find a mixture of samples, jobs, analyses, outputOptions, phenopackets, phenopacket family, vcf and ped files.

1.4.1 Job

An Exomiser *job* contains three sections:

- *sample* or phenopacket
- *preset* or *analysis*
- *outputOptions*

These fields define all the input data, analysis steps and output requirements for a complete Exomiser analysis. They can be used as a record of how a sample was analysed. Alternatively the various sections can all be defined at runtime using the CLI arguments as explained in the *Input Files and Options* section.

which defines run mode, *filters* and *prioritisers*, and the *output options* section that defines the output format, output file and number of results that should be returned. Each of these sections can be defined independently on the command line or provided as a single file as shown in the *job* section.

```
## Exomiser Job Template.
# The job is split into three sections:
#    sample: describes the proband
#    analysis: details the steps exomiser will take to analyse the sample
#    outputOptions: specifies what files to output and where with filtering options␣
↪for the number of genes and the exomiser score
#
# These can be input separately on the command line e.g. --sample sample.yml --
↪analysis analysis.yml
# In this example the 'sample' is replaced with a 'phenopacket' see https://
↪phenopacket-schema.readthedocs.io/en/1.0.0/phenopacket.html
```

(continues on next page)

(continued from previous page)

```
---
phenopacket:
  id: manuel
  subject:
    id: manuel
    sex: MALE
  phenotypicFeatures:
    - type:
        id: HP:0001159
        label: Syndactyly
    - type:
        id: HP:0000486
        label: Strabismus
    - type:
        id: HP:0000327
        label: Hypoplasia of the maxilla
    - type:
        id: HP:0000520
        label: Proptosis
    - type:
        id: HP:0000316
        label: Hypertelorism
    - type:
        id: HP:0000244
        label: Brachyturricephaly
  htsFiles:
    - uri: examples/Pfeiffer.vcf
      htsFormat: VCF
      genomeAssembly: hg19
  metaData:
    created: '2019-11-12T13:47:51.948Z'
    createdBy: julesj
    resources:
      - id: hp
        name: human phenotype ontology
        url: http://purl.obolibrary.org/obo/hp.owl
        version: hp/releases/2019-11-08
        namespacePrefix: HP
        iriPrefix: 'http://purl.obolibrary.org/obo/HP_'
    phenopacketSchemaVersion: 1.0
analysis:
  #FULL or PASS_ONLY
  analysisMode: PASS_ONLY
  # In cases where you do not want any cut-offs applied an empty map should be used e.
↪g. inheritanceModes: {}
  # These are the default settings, with values representing the maximum minor allele␣
↪frequency in percent (%) permitted for an
  # allele to be considered as a causative candidate under that mode of inheritance.
  # If you just want to analyse a sample under a single inheritance mode, delete/
↪comment-out the others. For AUTOSOMAL_RECESSIVE
  # or X_RECESSIVE ensure *both* relevant HOM_ALT and COMP_HET modes are present.
  inheritanceModes: {
    AUTOSOMAL_DOMINANT: 0.1,
    AUTOSOMAL_RECESSIVE_COMP_HET: 2.0,
    AUTOSOMAL_RECESSIVE_HOM_ALT: 0.1,
    X_DOMINANT: 0.1,
    X_RECESSIVE_COMP_HET: 2.0,
```

(continues on next page)

(continued from previous page)

```
    X_RECESSIVE_HOM_ALT: 0.1,
    MITOCHONDRIAL: 0.2
  }
  #Possible frequencySources:
  #Thousand Genomes project http://www.1000genomes.org/
  #   THOUSAND_GENOMES,
  #ESP project http://evs.gs.washington.edu/EVS/
  #   ESP_AFRICAN_AMERICAN, ESP_EUROPEAN_AMERICAN, ESP_ALL,
  #ExAC project http://exac.broadinstitute.org/about
  #   EXAC_AFRICAN_INC_AFRICAN_AMERICAN, EXAC_AMERICAN,
  #   EXAC_SOUTH_ASIAN, EXAC_EAST_ASIAN,
  #   EXAC_FINNISH, EXAC_NON_FINNISH_EUROPEAN,
  #   EXAC_OTHER
  #Possible frequencySources:
  #Thousand Genomes project - http://www.1000genomes.org/ (THOUSAND_GENOMES)
  #TOPMed - https://www.nhlbi.nih.gov/science/precision-medicine-activities (TOPMED)
  #UK10K - http://www.uk10k.org/ (UK10K)
  #ESP project - http://evs.gs.washington.edu/EVS/ (ESP_)
  #   ESP_AFRICAN_AMERICAN, ESP_EUROPEAN_AMERICAN, ESP_ALL,
  #ExAC project http://exac.broadinstitute.org/about (EXAC_)
  #   EXAC_AFRICAN_INC_AFRICAN_AMERICAN, EXAC_AMERICAN,
  #   EXAC_SOUTH_ASIAN, EXAC_EAST_ASIAN,
  #   EXAC_FINNISH, EXAC_NON_FINNISH_EUROPEAN,
  #   EXAC_OTHER
  #gnomAD - http://gnomad.broadinstitute.org/ (GNOMAD_E, GNOMAD_G)
  frequencySources: [
    THOUSAND_GENOMES,
    TOPMED,
    UK10K,

    ESP_AFRICAN_AMERICAN, ESP_EUROPEAN_AMERICAN, ESP_ALL,

    EXAC_AFRICAN_INC_AFRICAN_AMERICAN, EXAC_AMERICAN,
    EXAC_SOUTH_ASIAN, EXAC_EAST_ASIAN,
    EXAC_FINNISH, EXAC_NON_FINNISH_EUROPEAN,
    EXAC_OTHER,

    GNOMAD_E_AFR,
    GNOMAD_E_AMR,
    #         GNOMAD_E_ASJ,
    GNOMAD_E_EAS,
    GNOMAD_E_FIN,
    GNOMAD_E_NFE,
    GNOMAD_E_OTH,
    GNOMAD_E_SAS,

    GNOMAD_G_AFR,
    GNOMAD_G_AMR,
    #         GNOMAD_G_ASJ,
    GNOMAD_G_EAS,
    GNOMAD_G_FIN,
    GNOMAD_G_NFE,
    GNOMAD_G_OTH,
    GNOMAD_G_SAS
  ]
  # Possible pathogenicitySources: (POLYPHEN, MUTATION_TASTER, SIFT), (REVEL, MVP),␣
↪CADD, REMM
```

(continues on next page)

(continued from previous page)

```
  # REMM is trained on non-coding regulatory regions
  # *WARNING* if you enable CADD or REMM ensure that you have downloaded and␣
↪installed the CADD/REMM tabix files
  # and updated their location in the application.properties. Exomiser will not run␣
↪without this.
  pathogenicitySources: [ REVEL, MVP ]
  #this is the standard exomiser order.
  #all steps are optional
  steps: [
    #hiPhivePrioritiser: {},
    #priorityScoreFilter: {priorityType: HIPHIVE_PRIORITY, minPriorityScore: 0.500},
    #intervalFilter: {interval: 'chr10:123256200-123256300'},
    # or for multiple intervals:
    #intervalFilter: {intervals: ['chr10:123256200-123256300', 'chr10:123256290-
↪123256350']},
    # or using a BED file - NOTE this should be 0-based, Exomiser otherwise uses 1-
↪based coordinates in line with VCF
    #intervalFilter: {bed: /full/path/to/bed_file.bed},
    #genePanelFilter: {geneSymbols: ['FGFR1','FGFR2']},
      failedVariantFilter: { },
    #qualityFilter: {minQuality: 50.0},
      variantEffectFilter: {
        remove: [
            FIVE_PRIME_UTR_EXON_VARIANT,
            FIVE_PRIME_UTR_INTRON_VARIANT,
            THREE_PRIME_UTR_EXON_VARIANT,
            THREE_PRIME_UTR_INTRON_VARIANT,
            NON_CODING_TRANSCRIPT_EXON_VARIANT,
            UPSTREAM_GENE_VARIANT,
            INTERGENIC_VARIANT,
        REGULATORY_REGION_VARIANT,
        CODING_TRANSCRIPT_INTRON_VARIANT,
        NON_CODING_TRANSCRIPT_INTRON_VARIANT,
        DOWNSTREAM_GENE_VARIANT
      ]
    },
    # removes variants represented in the database
    #knownVariantFilter: {},
    frequencyFilter: {maxFrequency: 2.0},
    pathogenicityFilter: {keepNonPathogenic: true},
    # inheritanceFilter and omimPrioritiser should always run AFTER all other filters␣
↪have completed
    # they will analyse genes according to the specified modeOfInheritance above-␣
↪UNDEFINED will not be analysed.
    inheritanceFilter: {},
    # omimPrioritiser isn't mandatory.
    omimPrioritiser: {},
    #priorityScoreFilter: {minPriorityScore: 0.4},
    # Other prioritisers: Only combine omimPrioritiser with one of these.
    # Don't include any if you only want to filter the variants.
    hiPhivePrioritiser: {},
    # or run hiPhive in benchmarking mode:
    #hiPhivePrioritiser: {runParams: 'mouse'},
    #phivePrioritiser: {}
    #phenixPrioritiser: {}
    #exomeWalkerPrioritiser: {seedGeneIds: [11111, 22222, 33333]}
  ]
```

(continues on next page)

(continued from previous page)

```
outputOptions:
  outputContributingVariantsOnly: false
  #numGenes options: 0 = all or specify a limit e.g. 500 for the first 500 results
  numGenes: 0
  #outputPrefix options: specify the path/filename without an extension and this will␣
↪be added
  # according to the outputFormats option. If unspecified this will default to the␣
↪following:
  # {exomiserDir}/results/input-vcf-name-exomiser-results.html
  # alternatively, specify a fully qualifed path only. e.g. /users/jules/exomes/
↪analysis
  outputPrefix: results/Pfeiffer-hiphive-exome
  #out-format options: HTML, JSON, TSV_GENE, TSV_VARIANT, VCF (default: HTML)
  outputFormats: [HTML, JSON, TSV_GENE, TSV_VARIANT, VCF]
```

1.4.2 Sample

It is recommended to provide Exomiser the input sample as in Phenopacket format. Exomiser will accept this in either JSON or YAML format.

probandId:

hpoIds:

Input of the HPO identifiers/terms. You can select them using the HPO browser. Input must be in array format. Terms are comma separated and delimited by single quotes. For example *['HP:0001156', 'HP:0001363', 'HP:0011304', 'HP:0010055']*. It is *critical* that these are as detailed as possible and describe the observed phenotypes as fully and precisely as possible. These are used by the phenotype matching algorithms and will heavily influence the outcome of an analysis.

vcf:

The variant file in VCF format. There can be variants of multiple samples from one family in the file.

ped:

If you have multiple samples as input you have to define the pedigree using the ped format. It is important that you correctly define affected and unaffected individuals. If you use `X_RECESSIVE` as mode of inheritance be sure that the sex is correct (unknown is also fine).

1.4.3 Preset

```
# one of EXOME or GENOME. GENOME will require REMM to be available. Default is EXOME.
preset: EXOME
```

1.4.4 Analysis

The analysis shown below is equivalent to the `--preset exome` argument or adding

```
preset: EXOME
```

to an Exomiser job file. In most cases it is recommended to use the default presets as these have been thoroughly tested on the UK 100,000 genome rare-disease cohort. In case the user requires anything different, it is possible to manually define the data sources and steps in an Exomiser analysis.

```
---
analysisMode: PASS_ONLY
inheritanceModes: {
  AUTOSOMAL_DOMINANT: 0.1,
  AUTOSOMAL_RECESSIVE_HOM_ALT: 0.1,
  AUTOSOMAL_RECESSIVE_COMP_HET: 2.0,
  X_DOMINANT: 0.1,
  X_RECESSIVE_HOM_ALT: 0.1,
  X_RECESSIVE_COMP_HET: 2.0,
  MITOCHONDRIAL: 0.2
}
frequencySources: [
    THOUSAND_GENOMES,
    TOPMED,
    UK10K,

    ESP_AFRICAN_AMERICAN, ESP_EUROPEAN_AMERICAN, ESP_ALL,

    EXAC_AFRICAN_INC_AFRICAN_AMERICAN, EXAC_AMERICAN,
    EXAC_SOUTH_ASIAN, EXAC_EAST_ASIAN,
    EXAC_FINNISH, EXAC_NON_FINNISH_EUROPEAN,
    EXAC_OTHER,

    GNOMAD_E_AFR,
    GNOMAD_E_AMR,
  #         GNOMAD_E_ASJ,
    GNOMAD_E_EAS,
    GNOMAD_E_FIN,
    GNOMAD_E_NFE,
    GNOMAD_E_OTH,
    GNOMAD_E_SAS,

    GNOMAD_G_AFR,
    GNOMAD_G_AMR,
  #         GNOMAD_G_ASJ,
    GNOMAD_G_EAS,
    GNOMAD_G_FIN,
    GNOMAD_G_NFE,
    GNOMAD_G_OTH,
    GNOMAD_G_SAS
]
# Possible pathogenicitySources: (POLYPHEN, MUTATION_TASTER, SIFT), (REVEL, MVP),␣
↪CADD, REMM
# REMM is trained on non-coding regulatory regions
# *WARNING* if you enable CADD or REMM ensure that you have downloaded and installed␣
↪the CADD/REMM tabix files
# and updated their location in the application.properties. Exomiser will not run␣
↪without this.
```

(continues on next page)

(continued from previous page)

```
pathogenicitySources: [ REVEL, MVP ]
#this is the standard exomiser order.
steps: [
    failedVariantFilter: { },
    variantEffectFilter: {
      remove: [
          FIVE_PRIME_UTR_EXON_VARIANT,
          FIVE_PRIME_UTR_INTRON_VARIANT,
          THREE_PRIME_UTR_EXON_VARIANT,
          THREE_PRIME_UTR_INTRON_VARIANT,
          NON_CODING_TRANSCRIPT_EXON_VARIANT,
          NON_CODING_TRANSCRIPT_INTRON_VARIANT,
          CODING_TRANSCRIPT_INTRON_VARIANT,
          UPSTREAM_GENE_VARIANT,
          DOWNSTREAM_GENE_VARIANT,
          INTERGENIC_VARIANT,
          REGULATORY_REGION_VARIANT
      ]
    },
    frequencyFilter: { maxFrequency: 2.0 },
    pathogenicityFilter: { keepNonPathogenic: true },
    inheritanceFilter: { },
    omimPrioritiser: { },
    hiPhivePrioritiser: { }
]
```

analysisMode:

Can be `FULL` or `PASS_ONLY`. We highly recommend `PASS_ONLY` on genomes for because of memory issues. It will only keep variants that passes all filters. `FULL` will keep all variants, using the most memory and requiring the most time, but it can be useful for troubleshooting why a particular variant of interest might not have been returned in the results of an analysis.

```
analysisMode: PASS_ONLY
```

inheritanceModes:

Can be `AUTOSOMAL_DOMINANT`, `AUTOSOMAL_RECESSIVE`, `X_RECESSIVE` or `UNDEFINED`. This is a functionality of Jannovar. See its inheritance documentation for further information.

```
# These are the default settings, with values representing the maximum minor allele␣
↪frequency in percent (%) permitted for an
# allele to be considered as a causative candidate under that mode of inheritance.
# If you just want to analyse a sample under a single inheritance mode, delete/
↪comment-out the others. For AUTOSOMAL_RECESSIVE
# or X_RECESSIVE ensure *both* relevant HOM_ALT and COMP_HET modes are present.
# In cases where you do not want any cut-offs applied an empty map should be used e.g.
↪ inheritanceModes: {}
inheritanceModes: {
        AUTOSOMAL_DOMINANT: 0.1,
        AUTOSOMAL_RECESSIVE_HOM_ALT: 0.1,
        AUTOSOMAL_RECESSIVE_COMP_HET: 2.0,
        X_DOMINANT: 0.1,
```

(continues on next page)

(continued from previous page)

```
        X_RECESSIVE_HOM_ALT: 0.1,
        X_RECESSIVE_COMP_HET: 2.0,
        MITOCHONDRIAL: 0.2
}
```

frequencySources:

Here you can specify which variant frequency databases you want to use. You can add multiple databases using the same array format as the HPO IDs.

The data sources used are from 1000 genomes (via DBSNP), DBSNP, ESP, ExAC, gnomAD exomes and gnomAD genomes, UK10K (via DBSNP), TOPMed (via DBSNP).

DBSNP: `THOUSAND_GENOMES`, `UK10K`, `TOPMED`

ESP: `ESP_AFRICAN_AMERICAN`, `ESP_EUROPEAN_AMERICAN`, `ESP_ALL`

ExAC: `EXAC_AFRICAN_INC_AFRICAN_AMERICAN`, `EXAC_AMERICAN`, `EXAC_SOUTH_ASIAN`, `EXAC_EAST_ASIAN`, `EXAC_FINNISH`, `EXAC_NON_FINNISH_EUROPEAN`, `EXAC_OTHER`

gnomAD exomes: `GNOMAD_E_AFR`, `GNOMAD_E_AMR`, `GNOMAD_E_ASJ`, `GNOMAD_E_EAS`, `GNOMAD_E_FIN`, `GNOMAD_E_NFE`, `GNOMAD_E_OTH`, `GNOMAD_E_SAS`,

gnomAD genomes: `GNOMAD_G_AFR`, `GNOMAD_G_AMR`, `GNOMAD_G_ASJ`, `GNOMAD_G_EAS`, `GNOMAD_G_FIN`, `GNOMAD_G_NFE`, `GNOMAD_G_OTH`, `GNOMAD_G_SAS`

We recommend using all databases if the proband population background is unknown, although removing the `GNOMAD_E_ASJ` and `GNOMAD_G_ASJ`, unless your proband is known to come from an Ashkenazi population e.g.

```
frequencySources: [
  THOUSAND_GENOMES,
  TOPMED,
  UK10K,

  ESP_AFRICAN_AMERICAN, ESP_EUROPEAN_AMERICAN, ESP_ALL,

  EXAC_AFRICAN_INC_AFRICAN_AMERICAN, EXAC_AMERICAN,
  EXAC_SOUTH_ASIAN, EXAC_EAST_ASIAN,
  EXAC_FINNISH, EXAC_NON_FINNISH_EUROPEAN,
  EXAC_OTHER,

  GNOMAD_E_AFR,
  GNOMAD_E_AMR,
  #         GNOMAD_E_ASJ,
  GNOMAD_E_EAS,
  GNOMAD_E_FIN,
  GNOMAD_E_NFE,
  GNOMAD_E_OTH,
  GNOMAD_E_SAS,

  GNOMAD_G_AFR,
  GNOMAD_G_AMR,
  #         GNOMAD_G_ASJ,
  GNOMAD_G_EAS,
  GNOMAD_G_FIN,
  GNOMAD_G_NFE,
  GNOMAD_G_OTH,
```

(continues on next page)

(continued from previous page)

```
  GNOMAD_G_SAS
]
```

pathogenicitySources:

Possible pathogenicitySources: `POLYPHEN`, `MUTATION_TASTER`, `SIFT`, `REVEL`, `MVP`, `CADD`, `REMM`. `REMM` is trained on non-coding regulatory regions. **WARNING** if you enable `CADD`, ensure that you have downloaded and installed the CADD tabix files and updated their location in the `application.properties` (see *CADD data*). Exomiser will not run without this.

We recommend using either `[REVEL, MVP]` **OR** `[POLYPHEN, MUTATION_TASTER, SIFT]` as REVEL and MVP are newer predictors which have been shown to have better performance and are more nuanced. Mixing them with the Polyphen2, MutationTaster or SIFT will give worse performance.

```
pathogenicitySources: [REVEL, MVP, REMM]
```

steps

This section instructs exomiser which analysis steps should be run and with which criteria. **_n.b._ the order in which the steps are declared is important** - exomiser will run them in the order declared, although certain optimisations will happen automatically. We recommend using the [standard settings](../example/test-analysis-genome) for genome wide analysis as these have been optimised for both speed and memory performance. Nonetheless all steps are optional. Being an array of steps, this section must be enclosed in square brackets. Steps are comma separated and written in hash format *name: {options}*. **All steps are optional** - comment them out or delete them if you do not want to use them.

Analysis steps are defined in terms of *variant filters*, *gene filters* or *prioritisers*. The *inheritanceFilter* and *omimPrioritiser* are both somewhat anomalous as they operate on genes but also require the variants to have already been filtered. The optimiser will ensure that these are run at the correct time if they have been incorrectly placed.

Using these it is possible to create artificial exomes, define gene panels or only examine specific regions, for example.

Variant filters

These operate on variants and will produce a pass or fail result for each variant run through the filter.

failedVariantFilter:

Removes variants which are not marked as *PASS* or . in the VCF file. This is a highly recommended filter which will remove a lot of noise.

```
failedVariantFilter: {}
```

qualityFilter:

Removes variants with VCF *QUAL* scores lower than the given *minQuality*.

```
qualityFilter: {minQuality: 50.0}
```

intervalFilter:

Defines an interval of interest. Only variants within this interval will be passed. Currently only single intervals are possible.

```
intervalFilter: {interval: 'chr10:123256200-123256300'}
```

geneIdFilter:

You can define entrez-gene-ids for genes of interest. Only variants associated with these genes will be analyzed.

```
geneIdFilter: {geneIds: [12345, 34567, 98765]}
```

variantEffectFilter:

If you are interested only in specific functional classes of variants you can define a set of classes you want to remove from the output. Variant effects are generated by Jannovar. Jannovar uses Sequence Ontology (SO) terms and are listed in their manual.

```
variantEffectFilter: {remove: [SYNONYMOUS_VARIANT]}
```

regulatoryFeatureFilter:

If included it removes all non-regulatory, non-coding variants over 20Kb from a known gene. Intergenic and upstream variants in known enhancer regions over 20kb from a known gene are associated with genes in their TAD and not effected by this filter. This is an important filter to include when analysing whole-genome data.

```
regulatoryFeatureFilter: {}
```

knownVariantFilter:

Removes variants represented in the databases set in the *frequencySources* section. E.g. if you define

```
frequencySources: [THOUSAND_GENOMES]
```

every variant with an RS number will be removed. Variants without an RSid will be removed/failed if they are represented in any database defined in the *frequencySources* section. We do not recommend this option on recessive diseases.

```
knownVariantFilter: {}
```

frequencyFilter:

Frequency cutoff of a variant **in percent**. Frequencies are derived from the databases defined in the *frequencySources* section. We recommend a value below 5.0 % depending on the disease. Variants will be removed/failed if they have a frequency higher than the stated percentage in any database defined in the *frequencySources* section. _n.b_ Not defining this filter will result in all variants having no frequency data, even if the *frequencySources* contains values.

```
frequencyFilter: {maxFrequency: 1.0}
```

pathogenicityFilter:

Will apply the pathogenicity scores defined in the *pathogenicitySources* section to variants. If the `keepNonPathogenic` field is set to `true` then all variants will be kept. Setting this to `false` will set the filter to fail non-missense variants with pathogenicity scores lower than a score cutoff of 0.5. This filter is meant to be quite permissive and we recommend it be set to true.

```
pathogenicityFilter: {keepNonPathogenic: true}
```

Gene filters

These act at the gene-level and therefore may also refer to the variants associated with the gene. As a rule this is discouraged, although is broken by the inheritanceFiler.

priorityScoreFilter:

Running the prioritizer followed by a priorityScoreFilter will remove genes which are least likely to contribute to the phenotype defined in hpoIds, this will dramatically reduce the time and memory required to analyze a genome. 0.501 is a good compromise to select good phenotype matches and the best protein-protein interactions hits using the hiPhive prioritiser. PriorityType can be one of `HIPHIVE_PRIORITY`, `PHIVE_PRIORITY`, `PHENIX_PRIORITY`, `OMIM_PRIORITY`, `EXOMEWALKER_PRIORITY`.

```
priorityScoreFilter: {priorityType: HIPHIVE_PRIORITY, minPriorityScore: 0.501}
```

inheritanceFilter:

inheritanceFilter and **omimPrioritiser** should always run AFTER all other filters have completed. They will analyze genes according to the specified **modeOfInheritance** above. If set to `UNDEFINED` no filtering will be done. You can read more in the Jannovar inheritance documentation how exactly this filter works.

```
inheritanceFilter: {}
```

Prioritisers

These work on the gene-level and will produce the semantic-similarity scores for how well a gene matches the sample's HPO profile. We recommend using a combination of the *OMIM* and *hiPHIVE* prioritisers **only**. These have been tested on the UK's 100,000 genomes pilot project on rare disease diagnoses. Using other prioritisers may miss diagnoses as the data could be out of date or the algorithm's effectiveness may have been superseded. They are retained for historical interest.

omimPrioritiser:

inheritanceFilter and **omimPrioritiser** should always run AFTER all other filters have completed. Other prioritisers: Only combine **omimPrioritiser** with one of the next filters. The OMIM prioritiser adds known disease information

from OMIM to genes including the inheritance mode and then checks this inheritance mode with the compatible inheritance modes of the gene. Genes with incompatible modes will have their scores halved.

```
omimPrioritiser: {}
```

hiPhivePrioritiser:

Scores genes using phenotype comparisons to human, mouse and fish involving disruption of the gene or nearby genes in the interactome using a random walk.

See the hiPHIVE publication for details.

```
# Using the default
hiPhivePrioritiser: {}
```

is the same as

```
hiPhivePrioritiser: {runParams: 'human, mouse, fish, ppi'}
```

It is possible to only run comparisons agains a given organism/set of organisms `human,mouse,fish` and include/exclude protein-protein interaction proximities `ppi`. e.g. only using human and mouse data -

```
hiPhivePrioritiser: {runParams: 'human,mouse'}
```

phenixPrioritiser:

Scores genes using phenotype comparisons to known human disease genes. See the PhenIX publication for details.

```
phenixPrioritiser: {}
```

phivePrioritiser:

Scores genes using phenotype comparisons to mice with disruption of the gene. This is equivalent to running `hiPhivePrioritiser: {runParams: 'mouse'}`. See the PHIVE publication for details.

```
phivePrioritiser: {}
```

exomeWalkerPrioritiser:

Scores genes by proximity in interactome to the seed genes. Gene identifiers are required to be genbank identifiers. See the ExomeWalker publication for details.

```
exomeWalkerPrioritiser: {seedGeneIds: [11111, 22222, 33333]}
```

1.4.5 Output options

When specified as part of a command-line argument, the file should be a properly formed YAML object:

```
# Exomiser outputOptions
---
outputContributingVariantsOnly: true
numGenes: 20
minExomiserGeneScore: 0.7
outputPrefix: /analyses/sample-12345/sample-12345
outputFormats: [HTML, JSON]
```

When included as part of an exomiser *job* the fields are inlined in the `exomiser-job.yaml` file like so:

```
# other analysis and sample options above
outputOptions:
    outputContributingVariantsOnly: true
    numGenes: 20
    minExomiserGeneScore: 0.7
    outputPrefix: /analyses/sample-12345/sample-12345
    outputFormats: [HTML, JSON]
```

outputContributingVariantsOnly:

Can be `true` or `false`. Setting this to `true` will make the `TSV_VARIANT` and `VCF` output file only contain **PASS** variants which contribute to the gene score. Defaults to `true`.

numGenes:

Maximum number of genes listed in the results. If set to `0` all are listed. In most cases a limit of `50` is more than adequate as typically a good result will be found in the top 5. Not enabled by default.

minExomiserGeneScore

The mimimum gene (combined phenotype and variant) score required to be returned. As a rule of thumb scores >= 0.7 are a good score however, depending on the proband phenotype and the phenotype of best matching condition although it is not a hard-and-fast number. In our testing 0.7 gave the best performance in terms or recall and sensitivity. Not enabled by default.

outputPrefix:

Specify the path/filename without an extension and this will be added according to the *outputFormats* option. If unspecified this will default to the following: `{exomiserDir}/results/input-vcf-name-exomiser-results.html`. Alternatively, specify a fully qualified path only. e.g. `/home/jules/exomes/analysis`.

outputFormats:

Array to define the output formats. can be `[TSV-GENE]`, `[TSV-VARIANT]`, `[VCF]`, `JSON` or `HTML` or any combination like `[TSV-GENE, TSV-VARIANT, VCF]`. JSON is the most informative output option and suitable for use in downstream computational analyses or manually queried using something like jq. The HTML output is most suitable for manual inspection / human use. Default is `[JSON, HTML]`.

1.5 Structural Variant Prioritisation

Exomiser v13.0.0 is capable of jointly prioritising structural and sequence variants from a combined VCF file. For the purposes of the analysis a structural variant is defined as a variant >= 50 nucleotides in length or a symbolic variant (e.g. a variant with a VCF ALT allele of the form <DEL> instead of an actual sequence). There are many, many callers to choose from so performance will depend heavily on these as well as the underlying sequencing technology (e.g. Illumina short-read vs PacBio or Oxford Nanopore long-read sequencing). Exomiser has been tested on Manta and Canvas calls from Illumina short-reads produced for the 100K genomes project, with some compatibility testing against PacBio pbsv.

1.5.1 Prioritisation Overview

Assuming the recommended analysis steps are being used, Exomiser will broadly consider structural variants in a similar manner to sequence variants, with the major difference being that the precise size, position and sequence change is not known, especially for symbolic variants. Briefly, Exomiser performs these steps on each SV:

- Predict variant effects based on overlapping transcripts
- Assign variant to all genes for which it overlaps a transcript
- Assign variant pathogenicity score according to variant effect and known similar ClinVar variants
- Assign variant frequency score according to similar alleles in gnomAD-SV, DECIPHER, dbVar, DGV, GoNL

Following these SV-specific steps, the variant is considered in the same way as a sequence variant and is filtered and prioritised accordingly. This allows compound heterozygous genotypes of structural and sequence variants to be considered during an analysis.

1.5.2 Multiple Gene Overlaps

If a variant overlaps more than one gene, the variant will be associated with all of these genes and reported with the most severe variant effect it is predicted to have on each gene in the results.

For example, if a variant 1-23456-78910-N-<DEL> completely deletes GENE:1 and deletes the first exon of GENE:2, the variant will be reported twice, once for each associated gene. The following pseudo-code tries to illustrate this.

```
GENE1:
    variants:
        - 1-23456-78910-N-<DEL>
            variantEffect: TRANSCRIPT_ABLATION
GENE2:
    variants:
        - 1-23456-78910-N-<DEL>
            variantEffect: EXON_LOSS_VARIANT
```

1.5.3 SV Similarity

Given their imprecise nature, SVs are not looked-up in the database using precise coordinates, instead they are considered 'equal' if their genomic coordinates have a jaccard similarity of 0.8 and they constitute an equivalent broad type - gain, loss, me_gain, me_loss, inversion or complex.

1.6 Interpreting the Results

Depending on the output options provided, Exomiser will write out at least an HTML results file in the *results* sub-directory of the Exomiser installation.

As a general rule all output files contain a ranked list of genes and/or variants with the top-ranked gene/variant displayed first. The exception being the VCF output which, since version 13.1.0, is sorted according to VCF convention and tabix indexed.

Exomiser attempts to predict the variant or variants likely to be causative of a patient's phenotype and does so by associating them with the gene (or genes in the case of large structural variations) they intersect with on the genomic sequence. Variants occurring in intergenic regions are associated to the closest gene and those overlapping two genes are associated with the gene in which they are predicted to have the largest consequence.

Once associated with a gene, Exomiser uses the compatible modes of inheritance for a variant to assess it in the context of any diseases associated with the gene or any mouse knockout models of that gene. These are all bundled together into a *GeneScore* which features filtered variants located in that gene compatible with a given mode of inheritance. After the filtering steps Exomiser ranks these GeneScores according to descending combined score. The results are then written out to the files and formats specified in the output settings.

Prior to 13.1.0 a TSV/VCF file for each mode of inheritance (MOI) was created, which led to misunderstandings and confusion about how to deal with the data from them. The filenames were of the general pattern *output-prefix_MOI.genes.tsv* e.g. *Pfeiffer-hiphive-exome-PASS_ONLY_AD.genes.tsv*, *Pfeiffer-hiphive-exome-PASS_ONLY_AR.genes.tsv*, *Pfeiffer-hiphive-exome-PASS_ONLY_AR.vcf*, *Pfeiffer-hiphive-exome-PASS_ONLY_AR.variants.tsv...* **THESE FILES ARE NOW DEPRECATED AND WILL NOT APPEAR IN THE NEXT VERSION**. Similarly the *variants.tsv* and *vcf* output formats have also been changed and will only feature a single, combined output file of ranked genes/variants e.g. when supplying the *outputPrefix: Pfeiffer-hiphive-exome-PASS_ONLY* and *outputFormats: [TSV_VARIANT, TSV_GENE, VCF]*, the following files will be written out: *Pfeiffer-hiphive-exome-PASS_ONLY.variants.tsv Pfeiffer-hiphive-exome-PASS_ONLY.genes.tsv*, *Pfeiffer-hiphive-exome-PASS_ONLY.vcf*

These formats are detailed below.

1.6.1 HTML

1.6.2 JSON

The JSON file represents the most accurate representation of the data, as it is referenced internally by Exomiser. As such, we don't provide a schema for this, but it has been pretty stable and breaking changes will only occur with major version changes to the software. Minor additions are to be expected for minor releases, as per the SemVer specification.

We recommend using Python or JQ to extract data from this file.

1.6.3 TSV GENES

In the genes.tsv file it is possible for a gene to appear multiple times, depending on the MOI it is compatible with, given the filtered variants. For example in the example below MUC6 is ranked 7th under the AD model and 8th under an AR model.

```
#RANK       ID       GENE_SYMBOL     ENTREZ_GENE_ID  MOI     P-VALUE EXOMISER_GENE_
↪COMBINED_SCORE     EXOMISER_GENE_PHENO_SCORE       EXOMISER_GENE_VARIANT_SCORE     ␣
↪HUMAN_PHENO_SCORE       MOUSE_PHENO_SCORE       FISH_PHENO_SCORE        WALKER_
↪SCORE    PHIVE_ALL_SPECIES_SCORE OMIM_SCORE      MATCHES_CANDIDATE_GENE  HUMAN_
↪PHENO_EVIDENCE    MOUSE_PHENO_EVIDENCE    FISH_PHENO_EVIDENCE     HUMAN_PPI_
↪EVIDENCE      MOUSE_PPI_EVIDENCE      FISH_PPI_EVIDENCE
1   FGFR2_AD        FGFR2   2263    AD      0.0000  0.9981  1.0000  1.0000  0.8808  1.
↪0000  0.0000  0.5095  1.0000  1.0000  0       Jackson-Weiss syndrome (OMIM:123150):␣
↪Brachydactyly (HP:0001156)-Broad hallux (HP:0010055), Craniosynostosis (HP:0001363)-
↪Craniosynostosis (HP:0001363), Broad thumb (HP:0011304)-Broad metatarsal␣
↪(HP:0001783), Broad hallux (HP:0010055)-Broad hallux (HP:0010055),   Brachydactyly␣
↪(HP:0001156)-abnormal sternum morphology (MP:0000157), Craniosynostosis␣
↪(HP:0001363)-premature cranial suture closure (MP:0000081), Broad thumb␣
↪(HP:0011304)-abnormal sternum morphology (MP:0000157), Broad hallux (HP:0010055)-
↪abnormal sternum morphology (MP:0000157),              Proximity to FGF14␣
↪associated with Spinocerebellar ataxia 27 (OMIM:609307): Broad hallux (HP:0010055)-
↪Pes cavus (HP:0001761),   Proximity to FGF14 Brachydactyly (HP:0001156)-abnormal␣
↪digit morphology (MP:0002110), Broad thumb (HP:0011304)-abnormal digit morphology␣
↪(MP:0002110), Broad hallux (HP:0010055)-abnormal digit morphology (MP:0002110),
2   ENPP1_AD        ENPP1   5167    AD      0.0049  0.8690  0.5773  0.9996  0.6972  0.
↪5773  0.5237  0.5066  0.6972  1.0000  0       Autosomal recessive hypophosphatemic␣
↪rickets (ORPHA:289176): Brachydactyly (HP:0001156)-Genu varum (HP:0002970),␣
↪Craniosynostosis (HP:0001363)-Craniosynostosis (HP:0001363), Broad thumb␣
↪(HP:0011304)-Tibial bowing (HP:0002982), Broad hallux (HP:0010055)-Genu varum␣
↪(HP:0002970),   Brachydactyly (HP:0001156)-fused carpal bones (MP:0008915),␣
↪Craniosynostosis (HP:0001363)-abnormal nucleus pulposus morphology (MP:0006392),␣
↪Broad thumb (HP:0011304)-fused carpal bones (MP:0008915), Broad hallux (HP:0010055)-
↪fused carpal bones (MP:0008915),       Craniosynostosis (HP:0001363)-ceratohyal␣
↪cartilage premature perichondral ossification, abnormal (ZP:0012007), Broad thumb␣
↪(HP:0011304)-cleithrum nodular, abnormal (ZP:0006782),       Proximity to PAPSS2␣
↪associated with Brachyolmia 4 with mild epiphyseal and metaphyseal changes␣
↪(OMIM:612847): Brachydactyly (HP:0001156)-Brachydactyly (HP:0001156), Broad thumb␣
↪(HP:0011304)-Brachydactyly (HP:0001156), Broad hallux (HP:0010055)-Brachydactyly␣
↪(HP:0001156),         Proximity to PAPSS2 Brachydactyly (HP:0001156)-abnormal long␣
↪bone epiphyseal plate morphology (MP:0003055), Craniosynostosis (HP:0001363)-domed␣
↪cranium (MP:0000440), Broad thumb (HP:0011304)-abnormal long bone epiphyseal plate␣
↪morphology (MP:0003055), Broad hallux (HP:0010055)-abnormal long bone epiphyseal␣
↪plate morphology (MP:0003055),
//
7   MUC6_AD MUC6    4588    AD      0.0096  0.7532  0.5030  0.9990  0.0000  0.0000  0.
↪0000  0.5030  0.5030  1.0000  0                                       Proximity to␣
↪GKN2 Brachydactyly (HP:0001156)-brachydactyly (MP:0002544), Broad thumb␣
↪(HP:0011304)-brachydactyly (MP:0002544), Broad hallux (HP:0010055)-brachydactyly␣
↪(MP:0002544),
8   MUC6_AR MUC6    4588    AR      0.0096  0.7531  0.5030  0.9990  0.0000  0.0000  0.
↪0000  0.5030  0.5030  1.0000  0                                       Proximity to␣
↪GKN2 Brachydactyly (HP:0001156)-brachydactyly (MP:0002544), Broad thumb␣
↪(HP:0011304)-brachydactyly (MP:0002544), Broad hallux (HP:0010055)-brachydactyly␣
↪(MP:0002544),
```

1.6.4 TSV VARIANTS

In the variants.tsv file it is possible for a variant, like a gene, to appear multiple times, depending on the MOI it is compatible with. For example in the example below MUC6 has two variants ranked 7th under the AD model and two ranked 8th under an AR (compound heterozygous) model. In the AD case the CONTRIBUTING_VARIANT column indicates whether the variant was (1) or wasn't (0) used for calculating the EX-

OMISER_GENE_COMBINED_SCORE and EXOMISER_GENE_VARIANT_SCORE.

```
#RANK       ID       GENE_SYMBOL     ENTREZ_GENE_ID  MOI     P-VALUE EXOMISER_GENE_
↪COMBINED_SCORE    EXOMISER_GENE_PHENO_SCORE       EXOMISER_GENE_VARIANT_SCORE
↪EXOMISER_VARIANT_SCORE  CONTRIBUTING_VARIANT    WHITELIST_VARIANT       VCF_ID  RS_
↪ID   CONTIG  START   END     REF     ALT     CHANGE_LENGTH   QUAL    FILTER
↪GENOTYPE        FUNCTIONAL_CLASS        HGVS    EXOMISER_ACMG_CLASSIFICATION
↪EXOMISER_ACMG_EVIDENCE  EXOMISER_ACMG_DISEASE_ID        EXOMISER_ACMG_DISEASE_NAME
↪    CLINVAR_ALLELE_ID       CLINVAR_PRIMARY_INTERPRETATION  CLINVAR_STAR_RATING
↪GENE_CONSTRAINT_LOEUF   GENE_CONSTRAINT_LOEUF_LOWER     GENE_CONSTRAINT_LOEUF_UPPER
↪    MAX_FREQ_SOURCE MAX_FREQ        ALL_FREQ        MAX_PATH_SOURCE MAX_PATH
↪ALL_PATH
1   10-123256215-T-G_AD     FGFR2   2263    AD      0.0000  0.9981  1.0000  1.0000  1.
↪0000  1       1               rs121918506     10      123256215       123256215
↪  T       G       0       100.0000        PASS    1|0     missense_variant
↪FGFR2:ENST00000346997.2:c.1688A>C:p.(Glu563Ala) LIKELY_PATHOGENIC       PM2,PP3_
↪Strong,PP4,PP5  OMIM:123150     Jackson-Weiss syndrome  28333   LIKELY_PATHOGENIC
↪    1       0.13692 0.074   0.27                            REVEL   0.965   REVEL=0.
↪965,MVP=0.9517972
2   6-132203615-G-A_AD      ENPP1   5167    AD      0.0049  0.8690  0.5773  0.9996  0.
↪9996  1       0               rs770775549     6       132203615       132203615
↪  G       A       0       922.9800        PASS    0/1     splice_donor_variant
↪ENPP1:ENST00000360971.2:c.2230+1G>A:p.? UNCERTAIN_SIGNIFICANCE  PVS1_Strong
↪OMIM:615522     Cole disease            NOT_PROVIDED    0       0.41042 0.292   0.
↪586   GNOMAD_E_SAS    0.0032486517    TOPMED=7.556E-4,EXAC_NON_FINNISH_EUROPEAN=0.
↪0014985314,GNOMAD_E_NFE=0.0017907989,GNOMAD_E_SAS=0.0032486517
//
7   11-1018088-TG-T_AD      MUC6    4588    AD      0.0096  0.7532  0.5030  0.9990  0.
↪9990  1       0               rs765231061     11      1018088 1018089 TG      T
↪  -1      441.8100        PASS    0/1     frameshift_variant
↪MUC6:ENST00000421673.2:c.4712del:p.(Pro1571Hisfs*21)    UNCERTAIN_SIGNIFICANCE
↪                           NOT_PROVIDED    0       0.79622 0.656   0.971   GNOMAD_
↪G_NFE    0.0070363074    GNOMAD_E_AMR=0.0030803352,GNOMAD_G_NFE=0.0070363074
7   11-1018093-G-GT_AD      MUC6    4588    AD      0.0096  0.7532  0.5030  0.9990  0.
↪9989  0       0               rs376177791     11      1018093 1018093 G       GT
↪  1       592.4500        PASS    0/1     frameshift_elongation
↪MUC6:ENST00000421673.2:c.4707dup:p.(Pro1570Thrfs*136)   NOT_AVAILABLE
↪                     NOT_PROVIDED    0       0.79622 0.656   0.971   GNOMAD_G_NFE
↪0.007835763     GNOMAD_G_NFE=0.007835763
8   11-1018088-TG-T_AR      MUC6    4588    AR      0.0096  0.7531  0.5030  0.9990  0.
↪9990  1       0               rs765231061     11      1018088 1018089 TG      T
↪  -1      441.8100        PASS    0/1     frameshift_variant
↪MUC6:ENST00000421673.2:c.4712del:p.(Pro1571Hisfs*21)    UNCERTAIN_SIGNIFICANCE
↪                           NOT_PROVIDED    0       0.79622 0.656   0.971   GNOMAD_
↪G_NFE    0.0070363074    GNOMAD_E_AMR=0.0030803352,GNOMAD_G_NFE=0.0070363074
8   11-1018093-G-GT_AR      MUC6    4588    AR      0.0096  0.7531  0.5030  0.9990  0.
↪9989  1       0               rs376177791     11      1018093 1018093 G       GT
↪  1       592.4500        PASS    0/1     frameshift_elongation
↪MUC6:ENST00000421673.2:c.4707dup:p.(Pro1570Thrfs*136)   UNCERTAIN_SIGNIFICANCE
↪                           NOT_PROVIDED    0       0.79622 0.656   0.971   GNOMAD_
↪G_NFE    0.007835763     GNOMAD_G_NFE=0.007835763
```

1.6.5 VCF

In the VCF file it is possible for a variant, like a gene, to appear multiple times, depending on the MOI it is compatible with. For example in the example below MUC6 has two variants ranked 7th under the AD model and two ranked 8th under an AR (compound heterozygous) model. In the AD case the CONTRIBUTING_VARIANT column indicates

whether the variant was (1) or wasn't (0) used for calculating the EXOMISER_GENE_COMBINED_SCORE and EXOMISER_GENE_VARIANT_SCORE. The `INFO` field with the `ID=Exomiser` describes the internal format of this sub-field. Be aware that for multi-allelic sites, Exomiser will decompose and trim them for the proband sample and this is what will be displayed in the Exomiser `ID` sub-field e.g. `11-1018088-TG-T_AD`.

```
##INFO=<ID=Exomiser,Number=.,Type=String,Description="A pipe-separated set of values␣
↪for the proband allele(s) from the record with one per compatible MOI following the␣
↪format: {RANK|ID|GENE_SYMBOL|ENTREZ_GENE_ID|MOI|P-VALUE|EXOMISER_GENE_COMBINED_
↪SCORE|EXOMISER_GENE_PHENO_SCORE|EXOMISER_GENE_VARIANT_SCORE|EXOMISER_VARIANT_
↪SCORE|CONTRIBUTING_VARIANT|WHITELIST_VARIANT|FUNCTIONAL_CLASS|HGVS|EXOMISER_ACMG_
↪CLASSIFICATION|EXOMISER_ACMG_EVIDENCE|EXOMISER_ACMG_DISEASE_ID|EXOMISER_ACMG_
↪DISEASE_NAME}">
#CHROM      POS     ID      REF     ALT     QUAL    FILTER  INFO    sample
10  123256215       .       T       G       100     PASS    Exomiser={1|10-123256215-
↪T-G_AD|FGFR2|2263|AD|0.0000|0.9981|1.0000|1.0000|1.0000|1|1|missense_
↪variant|FGFR2:ENST00000346997.2:c.1688A>C:p.(Glu563Ala)|LIKELY_PATHOGENIC|PM2,PP3_
↪Strong,PP4,PP5|OMIM:123150|"Jackson-Weiss syndrome"};GENE=FGFR2;INHERITANCE=AD;
↪MIM=101600       GT:DS:PL        1|0:2.000:50,11,0
11  1018088 .       TG      T       441.81  PASS    AC=1;AF=0.50;AN=2;BaseQRankSum=7.
↪677;DP=162;DS;Exomiser={7|11-1018088-TG-T_AD|MUC6|4588|AD|0.0096|0.7532|0.5030|0.
↪9990|0.9990|1|0|frameshift_variant|MUC6:ENST00000421673.2:c.4712del:p.
↪(Pro1571Hisfs*21)|UNCERTAIN_SIGNIFICANCE|||""},{8|11-1018088-TG-T_AR|MUC6|4588|AR|0.
↪0096|0.7531|0.5030|0.9990|0.9990|1|0|frameshift_variant|MUC6:ENST00000421673.2:c.
↪4712del:p.(Pro1571Hisfs*21)|UNCERTAIN_SIGNIFICANCE|||""};FS=25.935;HRun=3;
↪HaplotypeScore=1327.2952;MQ=43.58;MQ0=6;MQRankSum=-5.112;QD=2.31;ReadPosRankSum=2.
↪472;set=variant        GT:AD:DP:GQ:PL  0/1:146,45:162:99:481,0,5488
11  1018093 .       G       GT      592.45  PASS    AC=1;AF=0.50;AN=2;BaseQRankSum=8.
↪019;DP=157;Exomiser={7|11-1018093-G-GT_AD|MUC6|4588|AD|0.0096|0.7532|0.5030|0.
↪9990|0.9989|0|0|frameshift_elongation|MUC6:ENST00000421673.2:c.4707dup:p.
↪(Pro1570Thrfs*136)|NOT_AVAILABLE|||""},{8|11-1018093-G-GT_AR|MUC6|4588|AR|0.0096|0.
↪7531|0.5030|0.9990|0.9989|1|0|frameshift_elongation|MUC6:ENST00000421673.2:c.
↪4707dup:p.(Pro1570Thrfs*136)|UNCERTAIN_SIGNIFICANCE|||""};FS=28.574;HRun=1;
↪HaplotypeScore=1267.6968;MQ=44.06;MQ0=4;MQRankSum=-5.166;QD=3.26;ReadPosRankSum=1.
↪328;set=variant    GT:AD:DP:GQ:PL  0/1:140,42:157:99:631,0,4411
6   132203615       .       G       A       922.98  PASS    AC=1;AF=0.50;AN=2;
↪BaseQRankSum=-0.671;DP=94;Dels=0.00;Exomiser={2|6-132203615-G-A_AD|ENPP1|5167|AD|0.
↪0049|0.8690|0.5773|0.9996|0.9996|1|0|splice_donor_variant|ENPP1:ENST00000360971.2:c.
↪2230+1G>A:p.?|UNCERTAIN_SIGNIFICANCE|PVS1_Strong|OMIM:615522|"Cole disease"};FS=0.
↪805;HRun=0;HaplotypeScore=3.5646;MQ=56.63;MQ0=0;MQRankSum=1.807;QD=9.82;
↪ReadPosRankSum=-0.900;set=variant2   GT:AD:DP:GQ:PL  0/1:53,41:94:99:953,0,1075
```

The VCF file is tabix-indexed and exomiser ranked alleles can be extracted using `grep`. For example, to display the top 5 ranked variants, you would issue the command:

```
zgrep -E '\{[1-5]{1}\|' Pfeiffer-hiphive-exome-PASS_ONLY.vcf.gz
```

1.7 ACMG Assignment

Starting with version 13.1.0, Exomiser performs a partial categorisation of the variants contributing to the gene score for a mode of inheritance using the ACMG/AMP Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology. The criteria are assigned and combined according to the UK ACGS 2020 guidelines.

It is important to be aware that these scores are not a substitute for manual assignment by a qualified clinical geneticist or clinician - The scores displayed utilise the data found in the Exomiser database and are a subset of the possible criteria by which to assess a variant. Nonetheless, in our benchmarking on the returned cases from the 100K Genomes

Project, restricting to variants with these automated P/LP classifications increases precision (positive predictive value) markedly without excluding many real diagnoses. For example, on a cohort of 742 solved cases where the top 5 Exomiser candidates were considered, including the P/LP criteria increased precision 3.8-fold from 15% to 57% with only a small drop in the recall of the diagnoses from 94% to 83%. An even larger 5.7-fold increase of precision from 3% to 17% was observed when considering a larger cohort of 31k cases where only 17% had received a positive diagnosis (again with a modest drop in recall from 91% to 75%).

Exomiser is capable of assigning the following ACMG categories:

1.7.1 Computational and Predictive Data

PVS1

Variants must have a predicted loss of function effect, be in a gene with known disease associations and have a gene constraint LOF O/E < 0.7635 (gnomAD 2.1.1) to suggest that a gene is LoF intolerant. Variants not predicted to lead to NMD (those located in the last exon) will have the modifier downgraded to Strong.

PM4

Stop-loss and in-frame insertions or deletions, not previously assigned a *PVS1* criterion are assigned *PM4*.

PP3 / BP4

If REVEL is chosen as a pathogenicity predictor for missense variants, *PP3* and *BP4* are assigned using the modifiers according to table 2 of Evidence-based calibration of computational tools for missense variant pathogenicity classification and ClinGen recommendations for clinical use of PP3/BP4 criteria. Note that this suggests the use of modifiers up to Strong in the case of pathogenic or Very Strong in the case of benign predictions. Otherwise, an ensemble-based approach will be used for other pathogenicity predictors as per the original 215 guidelines. It should be noted we found better performance using the REVEL-based approach when testing against the 100K genomes data.

1.7.2 Segregation Data

BS4

If a pedigree with two or more members, including the proband is provided, Exomiser will assign *BS4* for variants not segregating with affected members of the family.

1.7.3 De novo Data

PS2

Exomiser assigns the *PS2* criterion for variants compatible with a dominant mode of inheritance, with a pedigree containing at least two ancestors of the proband, none of whom are affected and none of whom share the same allele as the proband.

1.7.4 Population Data

BA1

Given Exomiser will filter out alleles with an allele frequency of >= 2.0%, this is unlikely to be seen. However, alleles with a maximum frequency >= 5.0% in the frequency sources specified will be assigned the *BA1* criterion.

PM2

Alleles not present in the ESP, ExAC and 1000 Genomes data sets (i.e. the allele must be absent from all three) are assigned the *PM2* criterion.

1.7.5 Allelic Data

PM3 / BP2

If Exomiser is provided with a phased VCF and a variant is found to be *in-trans* with a ClinVar Pathogenic variant and associated with a recessive disorder, the *PM3* criterion will be applied. However, in cases where variant is being considered for a recessive disorder and is *in-cis* or a dominant disorder and *in-trans* with another pathogenic variant the *BP2* criterion is applied.

1.7.6 Phenotype

PP4

Given Exomiser's focus on phenotype-driven variant prioritisation, variants in a gene associated with a disorder with a phenotype match score > 0.6 to the patient's phenotype are assigned the *PP4* criterion at the Moderate, rather than Supporting level.

1.7.7 Clinical

PP5 / BP6

If a variant is previously reported as P/LP in ClinVar with a 1-start rating, it will be assigned *PP5*, those with >= 2 stars (multiple submitters, criteria provided, no conflicts / reviewed by expert panel / practice guideline) will be assigned a Strong level. Conversely, if the variant is previously reported as B/LB it will be assigned *BP6* with the same modification criteria. Typically these P/LP variants will be in the Exomiser ClinVar 'whitelist', and will have a very high variant score irrespective of the predicted variant effect and always survive any filtering criteria.

1.7.8 Transcript Selection

Transcripts will be selected using the most deleterious predicted variant effect from Jannovar according to the *transcript-source* property set in the *application.properties*. We recommend using the Ensembl transcript datasource as the Exomiser build uses the GENCODE basic set of transcripts. Future versions should use MANE transcripts.

ACMG assignments will be reported for a variant on a transcript consistent with a particular mode of inheritance in conjunction with a disorder, the assigned criteria with any modifiers and the final classification e.g.

```
1-12335-A-T, NC_000001.10:g.12335A>T, GENE1(ENST12345678):c.2346A>T:p.1234A>-,␣
↪PATHOGENIC, [PVS1, PS1, PP4_Strong], Disease (OMIM:12345), AUTOSOMAL_DOMINANT
```

```
"acmgAssignments": [
  {
    "variantEvaluation": {
      "genomeAssembly": "HG19",
      "contigName": "10",
      "start": 123256215,
      "end": 123256215,
      "ref": "T",
      "alt": "G",
      "type": "SNV",
      "length": 1,
      "phredScore": 100,
      "variantEffect": "MISSENSE_VARIANT",
      "whiteListed": true,
      "filterStatus": "PASSED",
      "contributesToGeneScore": true,
      "variantScore": 1,
      "frequencyScore": 1,
      "pathogenicityScore": 1,
      "predictedPathogenic": true,
      "passedFilterTypes": [
        "FAILED_VARIANT_FILTER",
        "PATHOGENICITY_FILTER",
        "FREQUENCY_FILTER",
        "VARIANT_EFFECT_FILTER",
        "INHERITANCE_FILTER"
      ],
      "frequencyData": {
        "rsId": "rs121918506",
        "score": 1
      },
      "pathogenicityData": {
        "clinVarData": {
          "alleleId": "28333",
          "primaryInterpretation": "LIKELY_PATHOGENIC",
          "reviewStatus": "criteria provided, single submitter"
        },
        "score": 0.965,
        "predictedPathogenicityScores": [
          {
            "source": "REVEL",
            "score": 0.965
          },
          {
            "source": "MVP",
            "score": 0.9517972
          }
        ],
        "mostPathogenicScore": {
          "source": "REVEL",
          "score": 0.965
        }
      },
      "compatibleInheritanceModes": [
        "AUTOSOMAL_DOMINANT"
      ],
      "contributingInheritanceModes": [
```


(continues on next page)

(continued from previous page)


```
          "AUTOSOMAL_DOMINANT"
        ],
        "transcriptAnnotations": [
          {
            "variantEffect": "MISSENSE_VARIANT",
            "geneSymbol": "FGFR2",
            "accession": "ENST00000346997.2",
            "hgvsGenomic": "g.12278533A>C",
            "hgvsCdna": "c.1688A>C",
            "hgvsProtein": "p.(Glu563Ala)",
            "rankType": "EXON",
            "rank": 12,
            "rankTotal": 17
          },
          {
            "variantEffect": "MISSENSE_VARIANT",
            "geneSymbol": "FGFR2",
            "accession": "ENST00000351936.6",
            "hgvsGenomic": "g.12278533A>C",
            "hgvsCdna": "c.1688A>C",
            "hgvsProtein": "p.(Glu563Ala)",
            "rankType": "EXON",
            "rank": 13,
            "rankTotal": 18
          }
        ]
      },
      "geneIdentifier": {
        "geneId": "ENSG00000066468",
        "geneSymbol": "FGFR2",
        "hgncId": "HGNC:3689",
        "hgncSymbol": "FGFR2",
        "entrezId": "2263",
        "ensemblId": "ENSG00000066468",
        "ucscId": "uc057wle.1"
      },
      "modeOfInheritance": "AUTOSOMAL_DOMINANT",
      "disease": {
        "diseaseId": "OMIM:123150",
        "diseaseName": "Jackson-Weiss syndrome",
        "associatedGeneId": 2263,
        "diseaseType": "DISEASE",
        "inheritanceMode": "AUTOSOMAL_DOMINANT",
        "phenotypeIds": [
          "HP:0000006",
          "HP:0000272",
          "HP:0001363",
          "HP:0001783",
          "HP:0004691",
          "HP:0008080",
          "HP:0008122",
          "HP:0010055",
          "HP:0010743",
          "HP:0011800"
        ],
        "id": "OMIM:123150",
        "associatedGeneSymbol": "FGFR2"
```


(continues on next page)

(continued from previous page)

```
      },
      "acmgEvidence": {
        "evidence": {
          "PM2": "MODERATE",
          "PP3": "STRONG",
          "PP4": "SUPPORTING",
          "PP5": "SUPPORTING"
        }
      },
      "acmgClassification": "LIKELY_PATHOGENIC"
    }
  ]
}
```

1.8 Publications

100,000 Genomes Pilot on Rare-Disease Diagnosis in Health Care - Preliminary Report
100,000 Genomes Project Pilot Investigators; Smedley D, . . . Caulfield M.
N Engl J Med. 2021 Nov 11;385(20):1868-1880
PMID:34758253 DOI:10.1056/NEJMoa2035790

An Improved Phenotype-Driven Tool for Rare Mendelian Variant Prioritization: Benchmarking Exomiser on Real Patient Whole-Exome Data.
Cipriani V, Pontikos N, Arno G, Sergouniotis PI, Lenassi E, Thawong P, Danis D, Michaelides M, Webster AR, Moore AT, Robinson PN, Jacobsen JOB, Smedley D.
Genes (Basel). 2020 Apr 23;11(4):460
PMID:32340307 DOI:10.3390/genes11040460

A Whole-Genome Analysis Framework for Effective Identification of Pathogenic Regulatory Variants in Mendelian Disease.
Smedley D, Schubach M, Jacobsen JOB, Köhler S, Zemojtel T, Spielmann M, Jäger M, Hochheiser H, Washington NL, McMurry JA, Haendel MA, Mungall CJ, Lewis SE, Groza T, Valentini G, Robinson PN.
The American Journal of Human Genetics 2016;99;3;595-606
PMID:27569544 DOI:10.1016/j.ajhg.2016.07.005

Phenotype-driven strategies for exome prioritization of human Mendelian disease genes.
Smedley D, Robinson PN.
Genome Medicine 2015, 7(1):81
PMID:26229552 DOI:10.1186/s13073-015-0199-2

Next-generation diagnostics and disease-gene discovery with the Exomiser.
Smedley D, Jacobsen JO, Jäger M, Köhler S, Holtgrewe M et al.
Nature protocols 2015;10;12;2004-15
PMID:26562621 DOI:10.1038/nprot.2015.124

Computational evaluation of exome sequence data using human and model organism phenotypes improves diagnostic efficiency.
Bone WP, Washington NL, Buske OJ, Adams DR, Davis J et al.
Genetics in medicine : official journal of the American College of Medical Genetics 2015
PMID:26562225 DOI:10.1038/gim.2015.137

Walking the interactome for candidate prioritization in exome sequencing studies of Mendelian diseases.
Smedley D, Köhler S, Czeschik JC, Amberger J, Bocchini C et al.
Bioinformatics (Oxford, England) 2014;30;22;3215-22
PMID:25078397 DOI:10.1093/bioinformatics/btu508

Effective diagnosis of genetic disease by computational phenotype analysis of the disease-associated genome.
Zemojtel T, Köhler S, Mackenroth L, Jäger M, Hecht J et al.
Science translational medicine 2014;6;252;252ra123
PMID:25186178 DOI:10.1126/scitranslmed.3009262

Improved exome prioritization of disease genes through cross-species phenotype comparison.
Robinson PN, Köhler S, Oellrich A, Sanger Mouse Genetics Project, Wang K et al.
Genome research 2014;24;2;340-8
PMID:24162188 DOI:10.1101/gr.160325.113

1.8.1 Collaborations

Publications for other projects and resources in which the Exomiser is used as a component:

Defining Disease, Diagnosis, and Translational Medicine within a Homeostatic Perturbation Paradigm: The National Institutes of Health Undiagnosed Diseases Program Experience.
Gall T, Valkanas E, BelloC, Markello T, Adams C, Bone WP, Brandt AJ, Brazill JM, Carmichael L, Davids M, Davis J, Diaz-Perez Z, Draper D, Elson J, Flynn ED, Godfrey R, Groden C, Hsieh CK, Fischer R, Golas GA, Guzman J, Huang Y, Kane MS, Lee E, Li C, Links AE, Maduro V, Malicdan MCV, Malik FS, Nehrebecky M, Park J, Pemberton P, Schaffer K, Simeonov D, Sincan M, Smedley D, Valivullah Z, Wahl C, Washington N, Wolfe LA, Xu K, Zhu Y, Gahl WA, Tifft CJ, Toro C, Adams DR, He M, Robinson PN, Haendel MA, Zhai RG, Boerkoel CF.
Front Med (Lausanne). 2017;4 62
PMID: 28603714 DOI:10.3389/fmed.2017.00062

PhenomeCentral: A Portal for Phenotypic and Genotypic Matchmaking of Patients with Rare Genetic Diseases.
Orion J. Buske, Marta Girdea, Sergiu Dumitriu, Bailey Gallinger, Taila Hartley, Heather Trang, Andriy Misyura, Tal Friedman, Chandree Beaulieu, William P. Bone, Amanda E. Links, Nicole L. Washington, Melissa A. Haendel, Peter N. Robinson, Cornelius F. Boerkoel, David Adams, William A. Gahl, Kym M. Boycott, Michael Brudno.
Human Mutation 2015, 36(10):931-940
PMID: 26251998 DOI:10.1002/humu.22851

Phenopolis: an open platform for harmonization and analysis of genetic and phenotypic data.

Nikolas Pontikos, Jing Yu, Ismail Moghul, Lucy Withington, Fiona Blanco-Kelly, Tom Vulliamy, Tsz Lun, Ernest Wong, Cian Murphy, Valentina Cipriani, Alessia Fiorentino, Gavin Arno, Daniel Greene, Julius OB Jacobsen, Tristan Clark, David S. Gregory, Andrea M. Nemeth, Stephanie Halford, Chris F. Inglehearn, Susan Downes, Graeme C. Black, Andrew R. Webster, Alison J. Hardcastle, UKIRDC, Vincent Plagnol.

Bioinformatics 2017,33(15):2421-2423

PMID: 28334266 DOI:10.1093/bioinformatics/btx147