HOWARD User Guide

Highly Open Workflow for Annotation & Ranking toward genomic variant Discovery

HOWARD annotates and prioritizes genetic variations, calculates and normalizes annotations, translates files in multiple formats (e.g. vcf, tsv, parquet) and generates variants statistics.

HOWARD annotation is mainly based on a build-in Parquet annotation method, and external tools such as BCFTOOLS, ANNOVAR, snpEff, Exomiser and Splice (see docs, automatically downloaded if needed). Parquet annotation uses annotation database in VCF or BED format, in mutliple file format: Parquet/duckdb, VCF, BED, TSV, CSV, TBL, JSON.

HOWARD calculation processes variants information to calculate new information, such as: harmonizes allele frequency (VAF), extracts Nomen (transcript, cNomen, pNomen...) from HGVS fields with an optional list of personalized transcripts, generates VaRank format barcode.

HOWARD prioritization algorithm uses profiles to flag variants (as passed or filtered), calculate a prioritization score, and automatically generate a comment for each variants (example: 'polymorphism identified in dbSNP. associated to Lung Cancer. Found in ClinVar database').Prioritization profiles are defined in a configuration file. A profile is defined as a list of annotation/value, using wildcards and comparison options (contains, lower than, greater than, equal...). Annotations fields may be quality values (usually from callers, such as 'GQ', 'DP') or other annotations fields provided by annotations tools, such as HOWARD itself (example: COSMIC, Clinvar, 1000genomes, PolyPhen, SIFT). Multiple profiles can be used simultaneously, which is useful to define multiple validation/prioritization levels (example: 'standard', 'stringent', 'rare variants', 'low allele frequency').

HOWARD translates VCF format into multiple formats (e.g. VCF, TSV, Parquet), by sorting variants using specific fields (example : 'prioritization score', 'allele frequency', 'gene symbol'), including/excluding annotations/fields, including/excluding variants, adding fixed columns.

HOWARD generates statistics files with a specific algorithm, snpEff and BCFTOOLS.

HOWARD is multithreaded through the number of variants and by database (data-scaling).

Table of Contents

Installation

Python

Quick install

Install HOWARD using Python Pip tool:

```
python -m pip install -e .
```

Run HOWARD for help options:

```
howard --help
```

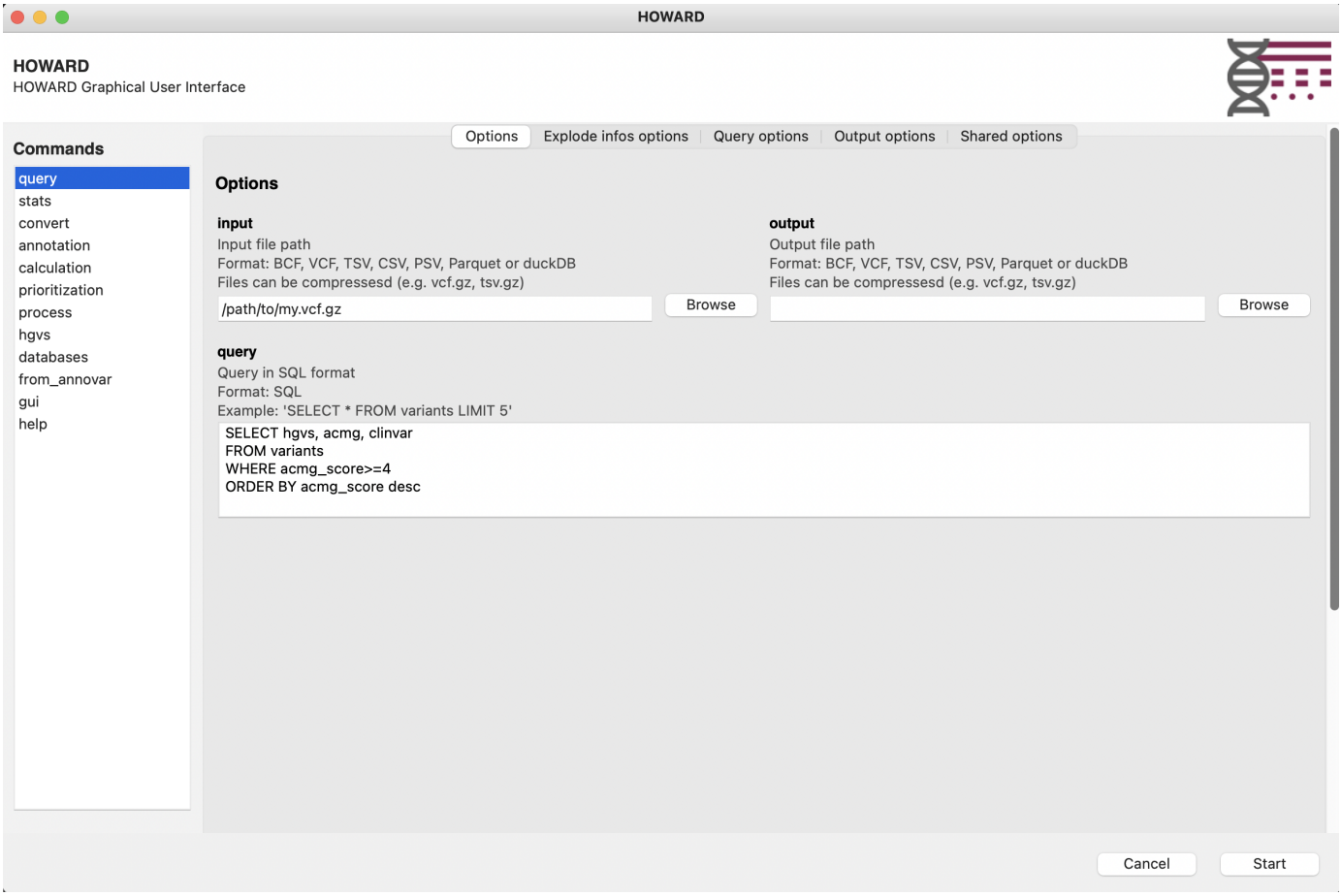
GUI install

Install HOWARD Graphical User Interface using Python Pip tool with supplementary packages:

```
python -m pip install -r requirements-gui.txt
```

Run HOWARD Graphical User Interface as a tool:

```
howard gui
```



Configuration

HOWARD Configuration JSON file defined default configuration regarding resources (e.g. threads, memory), settings (e.g. verbosity, temporary files), default folders (e.g. for databases) and paths to external tools.

See HOWARD Configuration JSON for more information.

Configuration file example:

```
{
  "threads": 8,
  "memory": null,
  "verbosity": "warning",
  "folders": {
    "databases": {
      "genomes": "~/howard/databases/genomes/current",
      "annotations": [
        "~/howard/databases/annotations/current",
        "~/howard/databases/dbnsfp/current",
        "~/howard/databases/dbsnp/current"
      ],
      "parquet": [
        "~/howard/databases/annotations/current"
        ],
      "bcftools": [
        "~/howard/databases/annotations/current"
        ],
      "annovar": [
        "~/howard/databases/annovar/current"
```

```
        ],
        "snpeff": "~/howard/databases/snpeff/current",
        "exomiser": "~/howard/databases/exomiser/current"
      }
    },
    "tools": {
      "bcftools": "bcftools",
      "bgzip": "bgzip",
      "java": "java",
      "snpeff": "~/howard/tools/snpeff/current/bin/snpEff.jar",
      "snpsift": "~/howard/tools/snpeff/current/bin/SnpSift.jar",
      "annovar": "~/howard/tools/annovar/current/bin/table_annovar.pl",
      "exomiser": "~/howard/tools/exomiser/current/bin/exomiser-cli-14.0.0.jar",
      "splice": {
        "docker": {
          "image": "bioinfochrustrasbourg/splice:0.2.1",
          "entrypoint": "/bin/bash",
          "options": null,
          "command": null
        }
      },
      "docker": "docker"
    }
  }
```

External tools

In order to use external tools, mainly for annotation (e.g. Annovar, snpEff, Exomiser, Splice), they need to be installed (see doc of each tools). For Splice tool, a Docker image is automatically downloaded using configuration file.

Docker

Quick Start

In order to build images, launch default setup and create a persitent CLI (Command Line Inferface container), docker-compose command build images and launch services as containers.

```
docker-compose up -d
```

The persitent CLI contains external tools, such as:

External tool	Description
BCFTools	Utilities for variant calling and manipulating VCFs and BCFs
snpEff	Genomic variant annotations, and functional effect prediction toolbox
Annovar	Efficient software tool to utilize update-to-date information to functionally annotate genetic variants
Exomiser	Program that finds potential disease-causing variants from whole-exome or whole-genome sequencing data
Splice	Image to run SPiP and SpliceAI tools in an nextflow pipeline.

Setup container

Docker service HOWARD-setup creates HOWARD image and download useful databases to start with HOWARD tools.

List of databases downloaded in HOWARD setup for hg19 assembly (see Databases section for more information):

- Genome
- Annovar (refGene, COSMIC)
- snpEff
- refSeq

- dbNSFP
- AlphaMissense
- dnSNP

To avoid databases download (see Databases section to download manually), just start Command Line Interface

Command Line Interface

A Command Line Interface container (HOWARD-CLI) is started with host data and databases folders mounted (by default both in ${HOME}/HOWARD folder). To manually start CLI container:

```
docker-compose up -d HOWARD-CLI
```

To use HOWARD tools within HOWARD-CLI container:

```
docker exec -ti HOWARD-CLI bash
howard --help
```

To run a command into HOWARD-CLI container:

```
docker exec HOWARD-CLI <howard command>
```

Docker HOWARD-CLI container (Command Line Interface) can be used to execute commands.

> Example: Query of an existing VCF
>
> ```
> docker exec HOWARD-CLI \
>    howard query \
>       --input=/tool/tests/data/example.vcf.gz \
>       --query='SELECT * FROM variants'
> ```

> Example: VCF annotation using HOWARD-CLI (snpEff and ANNOVAR databases will be automatically downloaded), and query list of genes with HGVS
>
> ```
> docker exec --workdir=/tool HOWARD-CLI \
>    howard process \
>       --config=config/config.json \
>       --param=config/param.json \
>       --input=tests/data/example.vcf.gz \
>       --output=/tmp/example.process.tsv \
>       --explode_infos \
>       --query='SELECT "NOMEN", "PZFlag", "PZScore", "PZComment" FROM variants ORDER BY
> "PZScore" DESC'
> ```

See HOWARD Help for more options.

Let's play within Docker HOWARD-CLI service!

Tests

In order to test HOWARD within Docker, use this command:

```
docker exec -ti HOWARD-CLI bash
cd /tool
# Basic test
```

```
coverage run -m pytest .
# Debug test
coverage run -m pytest . -x -v --log-cli-level=DEBUG --capture=tee-sys
```

Databases

Databases tool

Multiple databases can be automatically downloaded with databases tool, such as:

database	description
Genome	Genome Reference Consortium Human
Annovar	ANNOVAR is an efficient software tool to utilize update-to-date information to functionally annotate genetic variants detected from diverse genomes
snpEff	Genetic variant annotation, and functional effect prediction toolbox
refSeq	A comprehensive, integrated, non-redundant, well-annotated set of reference sequences including genomic, transcript, and protein
dbSNP	dbSNP contains human single nucleotide variations, microsatellites, and small-scale insertions and deletions along with publication, population frequency, molecular consequence, and genomic and RefSeq mapping information for both common variations and clinical mutations
dbNSFP	dbNSFP is a database developed for functional prediction and annotation of all potential non-synonymous single-nucleotide variants (nsSNVs) in the human genome
AlphaMissense	AlphaMissense model implementation
Exomiser	The Exomiser is a Java program that finds potential disease-causing variants from whole-exome or whole-genome sequencing data

> Example: Download Multiple databases in the same time for assembly 'hg19' (can take a while):
>
> ```
> howard databases \
>    --assembly=hg19 \
>    --download-genomes=~/howard/databases/genomes/current --download-genomes-provider=UCSC --download-genomes-contig-regex='chr[0-9XYM]+$' \
>    --download-annovar=~/howard/databases/annovar/current --download-annovar-files='refGene,cosmic70,nci60' \
>    --download-snpeff=~/howard/databases/snpeff/current \
>    --download-refseq=~/howard/databases/refseq/current --download-refseq-format-file='ncbiRefSeq.txt' \
>    --download-dbnsfp=~/howard/databases/dbnsfp/current --download-dbnsfp-release='4.4a' --download-dbnsfp-subdatabases \
>    --download-alphamissense=~/howard/databases/alphamissense/current \
>    --download-exomiser=~/howard/databases/exomiser/current \
>    --download-dbsnp=~/howard/databases/dbsnp/current --download-dbsnp-vcf --threads=8
> ```

See HOWARD Help Databases tool for more information.

Home-made Databases

Databases can be generated using an home-made existing annotation file and HOWARD convert tool. The home-made annotation file need to contain specific fields (depending on the annotation type):

- variant annotation: '#CHROM', 'POS', 'ALT', 'REF'
- region annotation: '#CHROM', 'START', 'STOP'

An home-made existing annotation file can be converted into multiple formats (e.g. Parquet, VCF, TSV), but it's strongly suggested to use Parquet format.

After convertion, the database file is associated with a 'header' file ('.hdr'), in VCF header format, to describe annotations within the database. Use the 'header' file to describe annotation fields/columns present in the existing file. An Home-made annotation file in VCF format which is converted in another format will keep all annotation information from the initial VCF header.

Note that a VCF can be directly used as a database (annotation field information within the header of the VCF file). Also, an home-made existing annotation file can be used as a database, but will not be totaly compliant due to the lack of annotation information ('header' will be generated by default).

See HOWARD Help Convert tool for more information.

Databases from Annovar and Extann

See HOWARD Help Databases tool tool for more information aboout the tool, and HOWARD Parameters Databases help for generate databases annotation file from Annovar databases (from Annovar, under development) and from Gene annotation file (from Extann).

Tools

HOWARD annotates and prioritizes genetic variations, calculates and normalizes annotations, convert on multiple formats, query variations and generates statistics. These tools require options or a Parameters JSON file.

Parameters

HOWARD Parameters JSON file defined parameters to process annotations, prioritization, calculations, convertions and queries. Use this parameters file to configure tools, instead of options or as a main configuration (options will replace parameters in JSON file).

See HOWARD Parameters JSON for more information.

Example: Use parameters JSON file with query tool

```
howard query \
   --input=tests/data/example.vcf.gz \
   --param=config/param.json
```

	#CHROM	POS	REF	ALT	INFO
0	chr1	28736	A	C	CLNSIG=pathogenic
1	chr1	35144	A	C	CLNSIG=non-pathogenic
2	chr1	69101	A	G	DP=50
3	chr1	768251	A	G	None
4	chr1	768252	A	G	None
5	chr1	768253	A	G	None
6	chr7	55249063	G	A	DP=125

Example: Use parameters JSON file with query tool, and add an option to change the query (list of chromosomes)

```
howard query \
   --input=tests/data/example.vcf.gz \
   --param=config/param.json \
   --query="SELECT distinct(\"#CHROM\") as 'chromosomes' FROM variants"
```

	chromosomes
0	chr7
1	chr1

Example: Parameters JSON file with multiple options for tools

```
{
  "annotation": {
    "parquet": {
      "annotations": {
        "tests/databases/annotations/current/hg19/avsnp150.parquet": {
          "INFO": null
        },
        "tests/databases/annotations/current/hg19/dbnsfp42a.parquet": {
          "INFO": null
        },
        "tests/databases/annotations/current/hg19/gnomad211_genome.parquet": {
          "INFO": null
        }
      }
    },
    "bcftools": {
      "annotations": {
        "tests/databases/annotations/current/hg19/cosmic70.vcf.gz": {
          "INFO": null
        }
      }
    },
    "snpeff": {
      "options": "-lof -hgvs -oicr -noShiftHgvs -spliceSiteSize 3 "
    },
    "snpsift": {
      "annotations": {
        "tests/databases/annotations/current/hg19/cosmic70.vcf.gz": {
          "INFO": null
        }
      }
    },
    "annovar": {
      "annotations": {
        "refGene": {
          "INFO": null
        }
      },
      "options": {
        "genebase": "-hgvs -splicing_threshold 3 ",
        "intronhgvs": 10
      }
    }
  },
  "calculation": {
    "calculations": {
      "vartype": null,
      "snpeff_hgvs": null,
      "VAF": "",
      "NOMEN": {
        "options": {
          "hgvs_field": "snpeff_hgvs",
          "transcripts": "tests/data/transcripts.tsv"
        }
      }
    },
    "config/calculations_config.json": "config/calculations_config.json"
  },
  "prioritization": {
    "profiles": ["default", "GERMLINE"],
    "prioritization_config": "config/prioritization_profiles.json",
    "pzfields": ["PZScore", "PZFlag", "PZComment"],
    "prioritization_score_mode": "VaRank"
  },
  "hgvs": {
    "full_format": true,
```

```
    "use_exon": true
  },
  "stats": {
    "stats_md": null,
    "stats_json": null
  },
  "query": {
    "query": "SELECT \"#CHROM\", POS, REF, ALT, INFO FROM variants",
    "query_limit": 10,
    "query_print_mode": "default"
  },
  "explode_infos": {
    "explode_infos": false,
    "explode_infos_prefix": "",
    "explode_infos_fields": null
  },
  "export": {
    "header_in_output": false,
    "parquet_partitions": null,
    "order_by": null
  },
  "threads": 8
}
```


Moreover, a transcripts file can be defined, especially to select NOMEN from a list of HGVS annotation (see Calculation and HGVS and NOMEN from snpEff). This file is a tab-delimited with 'transcript' as first column and 'gene' a second column. For a gene, transcripts of reference are ordered (first is priority, e.g. 'NM_001346897' has prior over 'NM_005228').

Example: Transcripts file in tab-delimited format with column 'transcript' and column 'gene'

```
NR_024540       WASH7P
NR_036266       MIR1302-9
NM_001346897    EGFR
NM_005228       EGFR
```

Stats

Generates statistics on genetic variations, such as number of variants, number of samples, statistics by chromosome, genotypes by samples, annotations. Theses statsitics can be applied to VCF files from all database annotation file formats. Statistics can be wrote into files in Markdown and JSON format (resp. `--stats_md` and `--stats_json` parameter).

See HOWARD Help Stats tool for more information.

Example: Show example VCF statistics and brief overview

```
howard stats \
   --input=tests/data/example.vcf.gz
```

Example: Show example VCF statistics and generate a file in JSON and Markdown formats (extract)

```
howard stats \
   --input=tests/data/example.vcf.gz \
   --stats_json=/tmp/stats.json \
   --stats_md=/tmp/stats.md

cat /tmp/stats.json /tmp/stats.md
```

```
{
    "Infos": {
        "Input file": "tests/data/example.vcf.gz",
        "Number of variants": 7,
        "Number of samples": 4,
        "Number of INFO fields": 5,
        "Number of FORMAT fields": 7
    },
    "Variants": {
        "Number of variants by chromosome": {
            "1": {
                "CHROM": "chr1",
                "count": 6,
                "percent": 0.8571428571428571
            },
            "0": {
                "CHROM": "chr7",
                "count": 1,
                "percent": 0.14285714285714285
            }
        }
    }
}
```


```
...
## Variants
### Number of variants by chromosome
| CHROM   |   count |   percent |
|:--------|--------:|----------:|
| chr1    |       6 |  0.857143 |
| chr7    |       1 |  0.142857 |
### Counts
| Type   |   count |
|:-------|--------:|
| Total  |       7 |
| SNV    |       7 |
| MNV    |       0 |
| InDel  |       0 |
...
```

Example of statistics in Markdown output

- Input file: tests/data/example.vcf.gz
- Number of variants: 7
- Number of samples: 4
- Number of INFO fields: 5
- Number of FORMAT fields: 7

CHROM	count	percent
chr1	6	0.857143
chr7	1	0.142857

Type	count
Total	7
SNV	7
MNV	0
InDel	0

Convert

Convert genetic variations file to another format. Multiple format are available, such as usual and official VCF format, but also other formats such as TSV, CSV, TBL, JSON and Parquet/duckDB. These formats need a header '.hdr' file to take advantage of the power of howard (especially through INFO/tag definition), and using howard convert tool automatically generate header file fo futher use (otherwise, an default '.hdr' file is generated).

Multiple options are available, such as explode VCF INFO/tags (parameter `--explode_infos`, see HOWARD Help query - Explode infos), order by columns, include header within file (only TSV and CSV format), or use partitioning into multiple files within a folder. See HOWARD Help Convert tool for more information.

CSV, TSV and JSON convert

To convert a file (multiple formats) into another flat file, such as CSV (tab-delimiter) and TSV (comma-delimiter), or JSON format, simply name output file with desired extension. Use `.gz` extension to compress file.

Example: Convert VCF into TSV and show output file

```
howard convert \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.tsv
cat /tmp/example.tsv
```

Example: Convert TSV into VCF and show output file

```
howard convert \
   --input=tests/data/example.tsv \
   --output=/tmp/example.vcf
cat /tmp/example.vcf
```

Example: Convert VCF into CSV and compress file

```
howard convert \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.csv.gz
```

Example: Convert VCF into JSON and compress file

```
howard convert \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.json.gz
```

Parquet and duckDB convert

Files can be format into Parquet and duckDB format. For duckDB format, a duckDB database will be created with a `variants` table.

Example: Convert VCF into parquet

```
howard convert \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.parquet
```

Example: Convert VCF into duckDB

```
howard convert \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.duckdb
```

BED convert

To convert into BED format, input file needs mandatory columns chrommosome "#CHROM", and positions columns "START"/"END" or "POS" (corresponfing to regions with a uniq nucleotide). Header will be automatically included to decribe all columns.

Example: Convert BED in TSV format into BED format

```
howard convert \
   --input=tests/data/example.bed.tsv \
   --output=/tmp/example.bed
```

Example: Convert BED in Parquet format into BED format

```
howard convert \
   --input=tests/data/example.bed.parquet \
   --output=/tmp/example.bed
```

HOWARD input file format does not allow BED format. To read a BED file and export into a CSV or Parquet file format, see Query BED format section.

Partitioning

Partitioning (or Hive partitioning) is a partitioning strategy that is used to split a table into multiple files based on partition keys. The files are organized into folders. Within each folder, the partition key has a value that is determined by the name of the folder (see duckDB hive partitioning).

Simply list columns as keys to process partitioning. Use 'None' (string) for NO partition but split parquet files into a folder. The partitioning is available for all format (e.g. Parquet, TSV, JSON, except duckDB format), by naming output file with desired extension. This option is faster parallel writing, but memory consuming, and also is faster reading.

Example: Convert VCF into partitioned Parquet and show tree structure

```
howard convert \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.partitioned.parquet \
   --parquet_partitions="#CHROM"

tree /tmp/example.partitioned.parquet
```

```
/tmp/example.partitioned.parquet
├── #CHROM=chr1
│   └── fe61bf182de640f8840270a527aa1582-0.parquet
└── #CHROM=chr7
    └── fe61bf182de640f8840270a527aa1582-0.parquet
```

Example: Convert VCF into partitioned TSV (compressed) and show tree structure (files are named with `.csv` extension, but are tab-delimited and compressed)

```
howard convert \
   --input=tests/data/example.vcf.gz \
```

```
    --output=/tmp/example.partitioned.tsv.gz \
    --parquet_partitions="#CHROM,REF"

tree /tmp/example.partitioned.tsv.gz
```

```
/tmp/example.partitioned.tsv
├── #CHROM=chr1
│   └── REF=A
│       └── data_0.csv
└── #CHROM=chr7
    └── REF=G
        └── data_0.csv
```

Explode INFO tags

Use `--explode_infos` parameter to extract all INFO tags (i.e. annotations) into columns (see HOWARD Help query - Explode infos).

Example: Convert VCF into TSV, export INFO/tags into columns, and show output file

```
howard convert \
   --input=tests/data/example.vcf.gz \
   --explode_infos \
   --output=/tmp/example.tsv

cut /tmp/example.tsv -f1-4,7,15
```

```
#CHROM  POS       REF  ALT  FILTER  CLNSIG
chr1    28736     A    C    PASS    pathogenic
chr1    35144     A    C    PASS    non-pathogenic
chr1    69101     A    G    PASS
chr1    768251    A    G    PASS
...
```

Query

Query tool provides a simple way to query genetic variations in SQL format. Data are loaded into 'variants' table from various formats (e.g. VCF, TSV, Parquet...). Using `--explode_infos` allows querying on INFO/tag annotations. SQL query can also use external file within the request, such as a Parquet file(s), or TSV files.

See HOWARD Help Query tool for more information.

Variants file

Loading data

Query tool is able to read variants (i.e. VCF) or regions files (i.e. BED) files, in various format (e.g. VCF, BED, Parquet, TSV, JSON), using `--input` parameter. This allows to load data to perfom actions, such as explode VCF INFO/tags (parameter `--explode_infos`, see HOWARD Help query - Explode infos) in columns to be easier querying. Each columns format (e.g. string, integer) are automatically detected to be used in a SQL query.

Example: Select variants in VCF with REF and POS fields filter

```
howard query \
   --input=tests/data/example.vcf.gz \
   --query="SELECT * FROM variants WHERE REF = 'A' AND POS < 100000"
```

Example: Select variants in VCF with INFO Tags criteria filters

```
howard query \
   --input=tests/data/example.vcf.gz \
   --explode_infos \
   --query='SELECT "#CHROM", POS, REF, ALT, DP, CLNSIG, sample2, sample3
            FROM variants
            WHERE DP >= 50 OR CLNSIG NOT NULL
            ORDER BY CLNSIG DESC, DP DESC'
```

Example: Select variants in VCF and generate VCF output with variants

```
howard query \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.filtered.vcf \
   --explode_infos \
   --query='SELECT "#CHROM", POS, REF, ALT, QUAL, FILTER, INFO
            FROM variants
            WHERE DP >= 50 OR CLNSIG NOT NULL'
```

External file

Variants files can be used directly within the query, espacially if they already contain variants information (e.g. "#CHROM", "POS", "REF", "ALT") and annotations as columns.

Example: Query a Parquet file with specific columns (e.g. from VCF convertion to Parquet)

```
howard query \
   --query="SELECT * \
            FROM 'tests/databases/annotations/current/hg19/dbnsfp42a.parquet' \
            WHERE \"INFO/Interpro_domain\" NOT NULL \
            ORDER BY \"INFO/SiPhy_29way_logOdds_rankscore\" DESC"
```

Example: Query multiple Parquet files, merge INFO columns, and extract as TSV (in VCF format)

```
howard query \
   --query="
      SELECT \
         \"#CHROM\" AS \"#CHROM\", \
         POS AS POS, \
         '' AS ID, \
         REF AS REF, \
         ALT AS ALT, \
         '' AS QUAL, \
         '' AS FILTER, \
         STRING_AGG(INFO, ';') AS INFO \
      FROM 'tests/databases/annotations/current/hg19/*.parquet' \
      GROUP BY \"#CHROM\", POS, REF, ALT" \
   --output=/tmp/full_annotation.tsv \
   --include_header
```

Other files

Whatever the external file format, if it is compatible with duckDB, query tool is able to query data (see duckDB Select Statement).

Parquet format

Simply use Parquet file path within the query (as descibe above).

Example: Query a Parquet file with specific columns (e.g. from VCF convertion to Parquet)

```
howard query \
   --query="SELECT * \
            FROM 'tests/databases/annotations/current/hg19/dbnsfp42a.parquet' \
            WHERE \"INFO/Interpro_domain\" NOT NULL \
            ORDER BY \"INFO/SiPhy_29way_logOdds_rankscore\" DESC"
```

CSV format

Use duckDB function `read_csv_auto` to read a TSV file format as a table. See duckDB CSV import for more information.

Example: Query a TSV file

```
howard query \
   --query="SELECT * FROM read_csv_auto('tests/data/transcripts.tsv')"
```

Example: Query a TSV file with columns struct

```
howard query \
   --query="SELECT * FROM read_csv_auto('tests/data/transcripts.tsv', columns=
{'transcript': 'VARCHAR','gene': 'VARCHAR'})"
```

BED format

In order to read a BED file, create a query (using appropiate columns), and export file into a desired format.

Example: Read a BED file and export in Parquet format

```
howard query \
   --query="SELECT * FROM read_csv_auto('tests/data/example.bed', columns={'#CHROM':
'VARCHAR', 'START': 'INTEGER', 'END': 'INTEGER'})" \
   --output=/tmp/example.bed.parquet
```

Example: Convert a BED file in a Parquet format into a BED file format

```
howard convert \
   --input=tests/data/example.bed.parquet \
   --output=/tmp/example.bed

cat /tmp/example.bed
```

```
#CHROM  START   END
chr1    28735   69101
chr1    768250  768253
chr7    55249060    55249069
```

A BED file can be filtered using positions or other columns such as gene names, or transcripts.

Example: Filter a BED using positions

```
howard query \
   --input=tests/data/example.bed.parquet \
```

```
    --query="SELECT * \
             FROM variants \
             WHERE \
                \"#CHROM\" = 'chr1' and \
                ( \
                   (\"START\">28000 and \"END\"<70000) or \
                   (\"START\">760000 and \"END\"<770000) \
                )"
```

```
  #CHROM   START     END
0   chr1   28735   69101
1   chr1  768250  768253
```

Extract variants

In order to extract variants from a VCF file, without annotations and samples, use a query to construct the VCF.

Example: Extract variants onnly

```
howard query \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.vcf.gz \
   --query="SELECT \"#CHROM\", POS, ID, REF, ALT, QUAL, FILTER, '.' AS INFO \
            FROM variants"
```

Annotation

Annotation is mainly based on a build-in Parquet annotation method, using annotation database file (in multiple format such as Parquet, duckdb, VCF, BED, TSV, JSON).

See HOWARD Help Annotation tool for more information.

These annotation databases can be automatically downloaded using HOWARD Databases tool and manually generated using existing annotation files and HOWARD Convert tool. Annotation databases need a header file (`.hdr`) to describe each annotation in the database. However, a default header will be generated if no header file is associated to the annotation database file.

Moreover, some external annotation tools are integrated into HOWARD to extend annotation with their own options and databases.

HOWARD annotation tool can use annotation databases files in 2 differents ways: Quick annotation and Annotation Parameters JSON file.

Quick annotation

Parquet annotation method

Quick annotation allows to annotates by simply listing annotation databases (in multiple format).

Parquet annotation with path

These annotation databases are defined with their full path (e.g. `/full/path/to/my.database.parquet`) or relative path (e.g. `databases/my.database.parquet`). A list (separator `:` or `+`) of annotation databases files can be used.

Example: VCF annotation with full path Parquet databases

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --annotations='/tool/tests/databases/annotations/current/hg19/dbnsfp42a.parquet' \
   --output=/tmp/example.howard.vcf.gz
```

Example: VCF annotation with relative path VCF databases

```
cd /tool
howard annotation \
   --input=tests/data/example.vcf.gz \
   --annotations='tests/databases/annotations/current/hg19/cosmic70.vcf.gz' \
   --output=/tmp/example.howard.vcf.gz
```

Example: VCF annotation with relative path BED databases

```
cd /tool
howard annotation \
   --input=tests/data/example.vcf.gz \
   --annotations='tests/databases/annotations/current/hg19/refGene.bed.gz' \
   --output=/tmp/example.howard.vcf.gz
```

Example: VCF annotation with 3 annotation databases files

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --
annotations='/tool/tests/databases/annotations/current/hg19/dbnsfp42a.parquet+tests/databas
es/annotations/current/hg19/cosmic70.vcf.gz+tests/databases/annotations/current/hg19/refGen
e.bed.gz' \
   --output=/tmp/example.howard.vcf.gz
```

Parquet annotation with annotation folder

If annotation folder is configured in HOWARD Configuration JSON, just mention the annotation database basename file. Annotation database file will be found (depending of the assembly).

Example: VCF annotation with Parquet and VCF databases, with annotation database defined in JSON configuration (as a string)

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --annotations='dbnsfp42a.parquet,cosmic70.vcf.gz' \
   --config='{"folders": {"databases": {"annotations":
["/tool/tests/databases/annotations/current"]}}}' \
   --output=/tmp/example.howard.vcf.gz
```

Full annotation

In order to annotate with all available annotation databases, the keyword `ALL` will auto-detect files in the databases annotation folder. The option `format` (defaut `parquet`) can filter annotation databases by listing (separator `+`) desired formats (such as `parquet`, `vcf`). The option `release` (default `current`) is able to scan annotation databases in one or more specific releases in a list (separator `+`). See HOWARD Configuration JSON - Folders - Databases for more information about databases structure.

Example: VCF annotation with all available database annotation files in Parquet format (within the database annotation folder in configuration):

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --assembly='hg19' \
   --annotations='ALL:format=parquet+vcf:release=current' \
   --config='{"folders": {"databases": {"annotations":
```

```
  ["/tool/tests/databases/annotations/current"]}}}' \
    --output=/tmp/example.howard.tsv
```

External tools annotation

External annotation tools are also available, such as BCFTOOLS, Annovar, snpEff, Exomiser and Splice. Annovar, snpEff and Exomiser databases are automatically downloaded (see HOWARD Help Databases tool). Quick annotation allows to annotates by simply defining external tools keywords.

BCFTools annotation

For BCFTools, use HOWARD keyword `bcftools` and list (separator `:` or `+`) annotation databases with format such as VCF or BED (compressed). More options are available using HOWARD Parameters JSON file.

Example: VCF annotation with Cosmic VCF databases and refGene BED database

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --
annotations='bcftools:tests/databases/annotations/current/hg19/cosmic70.vcf.gz+tests/databa
ses/annotations/current/hg19/refGene.bed.gz' \
   --output=/tmp/example.howard.vcf.gz
```

Annovar annotation

For Annovar tool, use HOWARD keyword `annovar` and mention specific Annovar database keywords (separator `:`). More options are available using HOWARD Parameters JSON file.

Example: VCF annotation with Annovar refGene and cosmic70

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --annotations='annovar:refGene:cosmic70' \
   --output=/tmp/example.howard.tsv
```

snpEff annotation

For snpEff tool, use HOWARD keyword `snpeff`. Options are available for quick annotation with snpEff, see HOWARD Parameters JSON - snpEff for more options.

Example: VCF annotation with snpEff

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --annotations='snpeff' \
   --output=/tmp/example.howard.tsv
```

Exomiser Annotation

For Exomiser tool, use HOWARD keyword `exomiser`. A list of options can be provided as key-value format, such as exomiser release, a preset (pre-configured options), source of transcripts (e.g. 'refseq', 'ucsc'), a list of HPO terms (do not use ':' separator, e.g. '0001156', 'HP0001156', 'hpo0001156'). More options are available using HOWARD Parameters JSON file.

Example: VCF annotation with Exomiser (exome preset, list of HPO terms, transcript as refseq and release 2109)

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --
annotations='exomiser:preset=exome:hpo=0001156+0001363+0011304+0010055:transcript_source=re
fseq:release=2109' \
   --output=/tmp/example.howard.tsv
```

Splice Annotation

For Splice tool, use HOWARD keyword `splice`. A list of options can be provided as key-value format, such as split mode, spliceAI distance, spliceAI mask. More options are available using HOWARD Parameters JSON file.

Example: VCF annotation with Splice (split mode, spliceAI distance, spliceAI mask)

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --annotations='splice:split_mode=one:spliceai_distance=500:spliceai_mask=1' \
   --output=/tmp/example.howard.tsv
```

Annotation combination

Quick annotation allows to combine annotations, from build-in Parquet method and external tools. Simply use a list with a comma separator.

Example: VCF annotation with build-in Parquet method and external tools (Annovar, snpEff and Exomiser)

```
howard annotation \
   --input=tests/data/example.vcf.gz \
   --
annotations='tests/databases/annotations/current/hg19/dbnsfp42a.parquet,bcftools:tests/data
bases/annotations/current/hg19/cosmic70.vcf.gz,annovar:refGene:cosmic70,snpeff,exomiser:pre
set=exome:hpo=0001156+0001363+0011304+0010055' \
   --output=/tmp/example.howard.tsv
```

See HOWARD Help Annotation tool tool for more information.

Annotation parameters

All annotation parameters can be defined in HOWARD Parameters JSON file. All annotations can be combined (bild-in parquet method and external tools annotation), and options can be detailed in a full JSON file format, including selection of annotation database columns (and rename them) and specific options for external tools.

Calculation

Calculation processes variants annotations to generate new annotation, such as: identify variation type (VarType), harmonizes allele frequency (VAF) and calculate statistics (VAF_stats), extracts Nomen (transcript, cNomen, pNomen...) from an HGVS field (e.g. snpEff, Annovar) with an optional list of personalized transcripts, generates VaRank format barcode, identify trio inheritance. These calculations are based on existing annotations of variants (and genotypes).

Calculations are either provided by HOWARD within code, or configured into a JSON file. Calculations are either an inner HOWARD Python code, or a SQL query.

See HOWARD Help Calculation tool tool for more information.

To process a calculation, use is keyword with the `--calculations` parameter.

Example: calculation of the variant type with `vartype` keyword

```
howard calculation \
   --input=tests/data/example.full.vcf \
   --calculations='vartype' \
   --output=/tmp/example.calculation.tsv
```

Available calculations

To list all available calculations, from HOWARD default configuration or with a homemade Calculation configuration JSON file, use the `--show_calculations` parameter.

Example: List of build-in calculation

```
howard calculation \
   --show_calculations
```

```
#[INFO] Start
#[INFO] Available calculation operations:
#[INFO]    BARCODE: BARCODE as VaRank tool
#[INFO]    DP_STATS: Depth (DP) statistics
#[INFO]    FINDBYPIPELINE: Number of pipeline that identify the variant (for multi pipeline
VCF)
#[INFO]    FINDBYSAMPLE: Number of sample that have a genotype for the variant (for multi
sample VCF)
#[INFO]    GENOTYPECONCORDANCE: Concordance of genotype for multi caller VCF
#[INFO]    NOMEN: NOMEN information (e.g. NOMEN, CNOMEN, PNOMEN...) from HGVS nomenclature
field
#[INFO]    SNPEFF_HGVS: HGVS nomenclatures from snpEff annotation
#[INFO]    TRIO: Inheritance for a trio family
#[INFO]    VAF: Variant Allele Frequency (VAF) harmonization
#[INFO]    VAF_STATS: Variant Allele Frequency (VAF) statistics
#[INFO]    VARIANT_ID: Variant ID generated from variant position and type
#[INFO]    VARTYPE: Variant type (e.g. SNV, INDEL, MNV, BND...)
#[INFO] End
```

Calculation configuration JSON file

All calculations are configured in a JSON file. A default configuration is provided with default calculations.

Basically, a calculation is defined by:

- Type: either 'sql' for a SQL query or 'python' for a Python function
- Name/keyword: a keyword that is used with `--show_calculations` parameter (case unsensitive)
- Description: a description of the calculation
- Output column information: Name, type and decription of the new annotation calculated
- Query and fields: an SQL query (for 'sql' type) with parameters such as mandatory INFO fields
- Function name and parameters: a existing Python function and parameters (for 'python' type)

Example: Configuration with calculation of variant type using an SQL query and calculation of variant id using an existing Python function `calculation_variant_id`

```
{
  "VARTYPE": {
    "type": "sql",
    "name": "VARTYPE",
    "description": "Variant type (e.g. SNV, INDEL, MNV, BND...)",
    "available": true,
    "output_column_name": "VARTYPE",
    "output_column_type": "String",
    "output_column_description": "Variant type: SNV if X>Y, MOSAIC if X>Y,Z or X,Y>Z, INDEL
```

```
  if XY>Z or X>YZ",
      "operation_query": [
        "CASE",
        "WHEN \"SVTYPE\" NOT NULL THEN \"SVTYPE\"",
        "WHEN LENGTH(REF) = 1 AND LENGTH(ALT) = 1 THEN 'SNV'",
        "WHEN REF LIKE '%,%' OR ALT LIKE '%,%' THEN 'MOSAIC'",
        "WHEN LENGTH(REF) == LENGTH(ALT) AND LENGTH(REF) > 1 THEN 'MNV'",
        "WHEN LENGTH(REF) <> LENGTH(ALT) THEN 'INDEL'",
        "ELSE 'UNDEFINED'",
        "END"
      ],
      "info_fields": ["SVTYPE"],
      "operation_info": true
    },
    "variant_id": {
      "type": "python",
      "name": "variant_id",
      "description": "Variant ID generated from variant position and type",
      "available": true,
      "function_name": "calculation_variant_id",
      "function_params": []
    }
  }
```

See Calculation configuration JSON file example.

See Calculation configuration JSON file for more information.

Build-in calculations examples

Variant type

Variant type calculation `vartype` detect the type of variant (e.g. SNV, INDEL, MNV). Variant type are claculated with these criteria: SNV if X>Y, MOSAIC if X>Y,Z or X,Y>Z, INDEL if XY>Z or X>YZ.

Example: Identify variant types and generate a table of variant type count

```
howard calculation \
   --input=tests/data/example.full.vcf \
   --calculations='vartype' \
   --output=/tmp/example.calculation.tsv

howard query \
   --input=/tmp/example.calculation.tsv \
   --explode_infos \
   --query='SELECT
               "VARTYPE" AS 'VariantType',
               count(*) AS 'Count'
            FROM variants
            GROUP BY "VARTYPE"
            ORDER BY count DESC'
```

```
  VariantType  Count
0         BND      7
1         DUP      6
2         INS      5
3         SNV      4
4         CNV      3
5         DEL      3
6         INV      3
7      MOSAIC      2
8       INDEL      2
9         MNV      1
```

HGVS and NOMEN from snpEff

NOMEN can be extracted from snpEff annotation (see HOWARD Parameters JSON - snpEff). The first calculation extract list of HGVS annotations from snpEff annotation (`snpeff_hgvs` keyword), the second calculation choose the NOMEN from snpEff HGVS annotations using a list of reference transcripts (`NOMEN` keyword, `--hgvs_field` and `--transcripts` parameters). More options are available (see HOWARD Parameters JSON). See Parameters for more information about list of reference transcripts

Example: Calculate NOMEN by extracting hgvs from snpEff annotation and identifying transcripts from a list

```
howard calculation \
   --input=tests/data/example.ann.vcf.gz \
   --calculations='snpeff_hgvs,NOMEN' \
   --hgvs_field='snpeff_hgvs' \
   --transcripts=tests/data/transcripts.tsv \
   --output=/tmp/example.NOMEN.vcf.gz

howard query \
   --input=/tmp/example.NOMEN.vcf.gz \
   --explode_infos \
   --query="SELECT GNOMEN, NOMEN, snpeff_hgvs \
            FROM variants \
            WHERE GNOMEN='EGFR'"
```

```
  GNOMEN  NOMEN                                          snpeff_hgvs
0   EGFR  EGFR:NM_001346897:exon19:c.2226G>A:p.Gln742Gln
EGFR:NM_005228.5:exon20:c.2361G>A:p.Gln787Gln,...
```

Prioritization

See HOWARD Help Prioritization tool tool for more information.

Prioritization algorithm uses profiles to flag variants (as passed or filtered), calculate a prioritization score, and automatically generate a comment for each variants (example: 'polymorphism identified in dbSNP. associated to Lung Cancer. Found in ClinVar database'). Prioritization profiles are defined in a configuration file in JSON format. A profile is defined as a list of annotation/value, using wildcards and comparison options (contains, lower than, greater than, equal...). Annotations fields may be quality values (usually from callers, such as 'DP') or other annotations fields provided by annotations tools, such as HOWARD itself (example: COSMIC, Clinvar, 1000genomes, PolyPhen, SIFT).

Prioritization options

Multiple profiles can be used simultaneously (`--prioritizations` option), which is useful to define multiple validation/prioritization levels (e.g. 'standard', 'stringent', 'rare variants', 'low allele frequency', 'GERMLINE'). By default, all profiles will be processed. A default profile can be defined with `--default_profile` option (by default, the first profile in list of profiles is selected).

Prioritization score can be calculated following multiple mode. The `HOWARD` mode will increment scores of all passing filters (default). The `VaRank` mode will select the maximum score from all passing filters.

Prioritization fields can be selected from:

- PZScore: calculated score from all passing filters, depending of the mode
- PZFlag: final flag ('PASS' or 'FILTERED'), with strategy that consider a variant is filtered as soon as at least one filter do not pass. By default, the variant is considered as 'PASS' (no filter pass)
- PZComment: concatenation of all passing filter comments
- PZTags: combinason of score, flags and comments in a tags format (e.g. 'PZFlag#PASS|PZScore#15|PZComment#Described on ...')
- PZInfos: information about passing filter criteria

Example: Prioritize variants from criteria on INFO annotations for profiles 'default' and 'GERMLINE' (from 'prioritization_profiles.json' profiles configuration), export prioritization tags, and query variants passing filters

```
howard prioritization \
   --input=tests/data/example.vcf.gz \
   --prioritizations='default,GERMLINE' \
   --prioritization_config=config/prioritization_profiles.json \
   --default_profile='default' \
   --pzfields='PZFlag,PZScore,PZComment,PZTags,PZInfos' \
   --prioritization_score_mode='HOWARD' \
   --output=/tmp/example.prioritized.vcf.gz

howard query \
   --input=/tmp/example.prioritized.vcf.gz \
   --explode_infos \
   --query="SELECT \"#CHROM\", POS, ALT, REF, \"PZFlag\", \"PZScore\", \"PZTags\", \"DP\",
\"CLNSIG\" \
            FROM variants \
            WHERE \"PZScore\" > 0 \
              AND \"PZFlag\" == 'PASS' \
            ORDER BY \"PZScore\" DESC"
```

```
  #CHROM       POS ALT REF    PZFlag  PZScore
PZTags     DP          CLNSIG
0   chr1     28736   C   A      PASS       15  PZFlag#PASS|PZScore#15|PZComment#Described
on ...    NaN      pathogenic
1   chr1     35144   C   A  FILTERED     -100
PZFlag#FILTERED|PZScore#-100|PZComment#Describ...    NaN  non-pathogenic
2   chr1     69101   G   A      PASS        5  PZFlag#PASS|PZScore#5|PZComment#DP higher
than...   50.0            None
3   chr1    768251   G   A      PASS        0
PZFlag#PASS|PZScore#0|PZComment#|PZInfos#    NaN            None
4   chr1    768252   G   A      PASS        0
PZFlag#PASS|PZScore#0|PZComment#|PZInfos#    NaN            None
5   chr1    768253   G   A      PASS        0
PZFlag#PASS|PZScore#0|PZComment#|PZInfos#    NaN            None
6   chr7  55249063   A   G      PASS        5  PZFlag#PASS|PZScore#5|PZComment#DP higher
than...  125.0            None
```

Prioritization query

Prioritization fields can be then easily querying, by filtering on fields and order by fields.

Example: Query variants using prioritization fields

```
howard query \
   --input=/tmp/example.prioritized.vcf.gz \
   --explode_infos \
   --query="SELECT \"#CHROM\", POS, ALT, REF, \"PZFlag\", \"PZScore\", \"PZTags\", \"DP\",
\"CLNSIG\" \
            FROM variants \
            WHERE \"PZScore\" > 0 \
              AND \"PZFlag\" == 'PASS' \
            ORDER BY \"PZScore\" DESC"
```

```
  #CHROM       POS ALT REF PZFlag  PZScore
PZTags     DP      CLNSIG
0   chr1     28736   C   A   PASS       15  PZFlag#PASS|PZScore#15|PZComment#Described on
...    NaN  pathogenic
```

```
1   chr1     69101   G   A   PASS        5  PZFlag#PASS|PZScore#5|PZComment#DP higher than
50   50.0        None
2   chr7  55249063   A   G   PASS        5  PZFlag#PASS|PZScore#5|PZComment#DP higher than
50  125.0        None
```

Example: Query variants with different prioritization flag between profiles

```
howard query \
   --input=/tmp/example.prioritized.vcf.gz \
   --explode_infos \
   --query="SELECT \"#CHROM\", POS, ALT, REF, \"PZFlag_default\", \"PZFlag_GERMLINE\" \
            FROM variants \
            WHERE \"PZFlag_default\" != \"PZFlag_GERMLINE\" \
            ORDER BY \"PZScore\" DESC"
```

```
  #CHROM    POS ALT REF PZFlag_default PZFlag_GERMLINE
0   chr1  35144   C   A       FILTERED            PASS
```

Example: Showing propritization comments of variants, with flags and scores

```
howard query \
   --input=/tmp/example.prioritized.vcf.gz \
   --explode_infos \
   --query="SELECT \"#CHROM\", POS, ALT, REF, \"PZComment\", \"PZFlag\", \"PZScore\" \
            FROM variants WHERE \"PZComment\" IS NOT NULL \
            ORDER BY \"PZScore\" DESC"
```

```
  #CHROM       POS ALT REF                                      PZComment    PZFlag
PZScore
0   chr1     28736   C   A      Described on CLINVAR database as pathogenic      PASS
15
1   chr1     69101   G   A                                  DP higher than 50      PASS
5
2   chr7  55249063   A   G                                  DP higher than 50      PASS
5
3   chr1     35144   C   A  Described on CLINVAR database as non-pathogenic  FILTERED
-100
```

Prioritization profiles are defined in a JSON configuration file. Each profiles are defined as a list of annotation fields with associated filters (type of comparison and threshold, with related score, flag and comment).

Example: Profiles with 2 filters on annotation field 'DP' (threashold 50) and 2 filters on annotation field 'CLNSIG' ("pathogenic" or "non-pathogenic")

```
{
    "default": {
        "DP": [
            {
                "type": "gte",
                "value": "50",
                "score": 5,
                "flag": "PASS",
                "comment": [
                    "DP higher than 50"
                ]
            },
```

```
            {
                "type": "lt",
                "value": "50",
                "score": 0,
                "flag": "FILTERED",
                "comment": [
                    "DP lower than 50"
                ]
            }
        ],
        "CLNSIG": [
            {
                "type": "equals",
                "value": "pathogenic",
                "score": 15,
                "flag": "PASS",
                "comment": [
                    "Described on CLINVAR database as pathogenic"
                ]
            },
            {
                "type": "equals",
                "value": "non-pathogenic",
                "score": -100,
                "flag": "FILTERED",
                "comment": [
                    "Described on CLINVAR database as non-pathogenic"
                ]
            }
        ]
    }
}
```


See HOWARD Help Prioritization Profiles for more options.

HGVS annotation

HOWARD annotates variants with HGVS annotation using HUGO HGVS internation Sequence Variant Nomenclature (http://varnomen.hgvs.org/). Annotation refere to refGene and genome to generate HGVS nomenclature for all available transcripts. This annotation add 'hgvs' field into VCF INFO column of a VCF file.

To enhance the functionality of HGVS tool, several options available. The `--use_gene` option enables the utilization of gene information for the generation of HGVS annotation, providing a concise representation such as 'NM_152232(TAS1R2):c.231T>C'. Alternatively, the `--use_exon` option incorporates exon details into the annotation, as seen in the example 'NM_152232(exon2):c.231T>C', but this is only applicable if 'use_gene' is not enabled. For a more detailed HGVS annotation, activate the `--full_format` option, which generates a comprehensive HGVS annotation, including all available information, such as 'TAS1R2:NM_152232:NP_689418:c.231T>C:p.Cys77Arg'. This is non-standard format that ensures exhaustive data representation. Furthermore, specify the format of amino acid codons with the `--codon_type` option, choosing between single-character ('1'), three-character ('3'), or full-name formats ('FULL'). Additionally, protein-level information can be integrated using the `--use_protein` option, enabling annotations like 'NP_689418:p.Cys77Arg', which can be combined with DNA-level annotations using the `--add_protein` option (e.g. 'NM_152232:c.231T>C,NP_689418:p.Cys77Arg').

All these options can be combined into one quick option `--hgvs`. Simply concatenate options (separator ',') with there value (e.g `--hgvs=use_gene:True` or `--hgvs=use_gene:False,codon_type:FULL`). By default, value option is 'True' (e.g. `--hgvs=use_gene` is equal to `--hgvs=use_gene:True`)

For database management, specify refSeq annotation files with `--refgene` and `--refseqlink` options (or set custom folders with `--refseq-folder`), and genomes folder using and `--genomes-folder` (folders are set by default, see HOWARD Configuration JSON file).

See HOWARD Help Prioritization tool for more options.

> Example: HGVS annotation with quick options

```
howard hgvs \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.process.tsv \
   --hgvs=full_format,use_exon

howard query \
   --input=/tmp/example.process.tsv \
   --explode_infos \
   --query="SELECT hgvs \
            FROM variants "
```

Example: HGVS annotation with separated options

```
howard hgvs \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.process.tsv \
   --full_format \
   --use_exon

howard query \
   --input=/tmp/example.process.tsv \
   --explode_infos \
   --query="SELECT hgvs \
            FROM variants "
```

```
                                                hgvs
0                     WASH7P:NR_024540.1:n.50+585T>G
1     FAM138A:NR_026818.1:exon3:n.597T>G:p.Tyr199Asp
2  OR4F5:NM_001005484.2:NP_001005484.2:exon3:c.74...
3  LINC01128:NR_047526.1:n.287+3767A>G,LINC01128:...
4  LINC01128:NR_047526.1:n.287+3768A>G,LINC01128:...
5  LINC01128:NR_047526.1:n.287+3769A>G,LINC01128:...
6  EGFR:NM_001346897.2:NP_001333826.1:exon19:c.22...
```

Process

HOWARD process tool manage genetic variations to:

- annotates genetic variants with multiple annotation databases/files and tools
- calculates and normalizes annotations
- prioritizes variants with profiles (list of citeria) to calculate scores and flags
- annotates genetic variants with HGVS nomenclature
- translates into various formats
- query genetic variants and annotations
- generates variants statistics

This process tool combines all other tools to pipe them in a uniq command, through available options or a parameters file in JSON format (see HOWARD Parameters JSON file).

See HOWARD Help Process tool tool for more information (under development).

Process with options

Process tool uses quick options for annotation, calculation and prioritization to enrich variant file, and to query variants annotations.

Example: Process command with options (HGVS, annotation, calculation)

```
howard process \
   --input=tests/data/example.vcf.gz \
```

```
   --output=/tmp/example.process.tsv \
   --hgvs=full_format,use_exon \
   --
annotations='tests/databases/annotations/current/hg19/avsnp150.parquet,tests/databases/anno
tations/current/hg19/dbnsfp42a.parquet,tests/databases/annotations/current/hg19/gnomad211_g
enome.parquet' \
   --calculations="NOMEN" \
   --explode_infos \
   --query="SELECT NOMEN, REVEL_score, SIFT_score, AF AS 'gnomad_AF', ClinPred_score,
ClinPred_pred \
            FROM variants"
```

```
                                          NOMEN REVEL_score SIFT_score gnomad_AF
ClinPred_score ClinPred_pred
0                    WASH7P:NR_024540:n.50+585T>G        None       None      None
None          None
1    FAM138A:NR_026818:exon3:n.597T>G:p.Tyr199Asp        None       None      None
None          None
2     OR4F5:NP_001005484:exon3:c.74A>G:p.Glu25Gly       0.076      0.005      None
0.688             D
3               LINC01128:NR_047526:n.287+3767A>G        None       None      None
None          None
4               LINC01128:NR_047526:n.287+3768A>G        None       None      None
None          None
5               LINC01128:NR_047526:n.287+3769A>G        None       None      None
None          None
6  EGFR:NM_001346897:exon19:c.2226G>A:p.Gln742Gln        None       None    0.5029
None          None
```

Example: Full process command with options (HGVS, annotation parquet, snpEff and Annovar, calculation and prioritization)

```
howard process \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.process.tsv \
   --hgvs=full_format,use_exon \
   --
annotations="tests/databases/annotations/current/hg19/avsnp150.parquet,tests/databases/anno
tations/current/hg19/dbnsfp42a.parquet,tests/databases/annotations/current/hg19/gnomad211_g
enome.parquet,bcftools:tests/databases/annotations/current/hg19/cosmic70.vcf.gz,snpeff,anno
var:refGene" \
   --calculations="vartype,snpeff_hgvs,VAF,NOMEN" \
   --prioritizations="default,GERMLINE" \
   --prioritization_config="config/prioritization_profiles.json" \
   --explode_infos \
   --query="SELECT NOMEN, PZFlag, PZScore, snpeff_hgvs \
            FROM variants \
            ORDER BY PZScore DESC"
```

```
                                          NOMEN    PZFlag  PZScore
snpeff_hgvs
0                    WASH7P:NR_024540:n.50+585T>G      PASS       15  MIR1302-
2:NR_036051.1:n.-1630A>C,MIR1302-9:NR_...
1     OR4F5:NP_001005484:exon3:c.74A>G:p.Glu25Gly      PASS        5
OR4F5:NM_001005484.1:exon1:c.11A>G:p.Glu4Gly
2  EGFR:NM_001346897:exon19:c.2226G>A:p.Gln742Gln      PASS        5
EGFR:NM_005228.5:exon20:c.2361G>A:p.Gln787Gln,...
3               LINC01128:NR_047526:n.287+3767A>G      PASS        0
LINC01128:NR_047519.1:exon2:n.287+3767A>G,LINC...
4               LINC01128:NR_047526:n.287+3768A>G      PASS        0
LINC01128:NR_047519.1:exon2:n.287+3768A>G,LINC...
5               LINC01128:NR_047526:n.287+3769A>G      PASS        0
```

```
LINC01128:NR_047519.1:exon2:n.287+3769A>G,LINC...
6    FAM138A:NR_026818:exon3:n.597T>G:p.Tyr199Asp  FILTERED     -100  MIR1302-
2:NR_036051.1:n.*4641A>C,MIR1302-9:NR_...
```

Process with parameters JSON file

In order to fine tune process, all tools can be defined in a HOWARD Parameters JSON. This allows to add specific options, such as selecting specific fields (and rename them) for annotation, defining options for external tools, specifying a list of transcripts of preference for NOMEN calculation. This Parameters JSON file can be combine with options.

Example: Full process command with Parameters JSON file example and a query as option

```
howard process \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.process.tsv \
   --param=config/param.json \
   --explode_infos \
   --query="SELECT NOMEN, PZFlag, PZScore, PZComment \
            FROM variants \
            ORDER BY PZScore DESC"
```

```
                                        NOMEN    PZFlag  PZScore
PZComment
0              WASH7P:NR_024540:exon1:n.50+585T>G      PASS       15.0      Described on
CLINVAR database as pathogenic
1      OR4F5:NM_001005484:exon1:c.11A>G:p.Glu4Gly      PASS        5.0
DP higher than 50
2  EGFR:NM_001346897:exon19:c.2226G>A:p.Gln742Gln      PASS        5.0
DP higher than 50
3         LINC01128:NR_047519:exon2:n.287+3767A>G      PASS        0.0
NaN
4         LINC01128:NR_047519:exon2:n.287+3768A>G      PASS        0.0
NaN
5         LINC01128:NR_047519:exon2:n.287+3769A>G      PASS        0.0
NaN
6                 MIR1302-9:NR_036266:n.*4641A>C  FILTERED        NaN  Described on
CLINVAR database as non-pathogenic
```

```
column -t  /tmp/example.process.tsv
```

```
NOMEN                                          PZFlag    PZScore    PZComment
WASH7P:NR_024540:exon1:n.50+585T>G             PASS      15         Described  on
CLINVAR   database  as              pathogenic
OR4F5:NM_001005484:exon1:c.11A>G:p.Glu4Gly     PASS      5          DP         higher
than      50
EGFR:NM_001346897:exon19:c.2226G>A:p.Gln742Gln  PASS      5          DP         higher
than      50
LINC01128:NR_047519:exon2:n.287+3767A>G        PASS      0
LINC01128:NR_047519:exon2:n.287+3768A>G        PASS      0
LINC01128:NR_047519:exon2:n.287+3769A>G        PASS      0
MIR1302-9:NR_036266:n.*4641A>C                 FILTERED  Described  on         CLINVAR
database  as        non-pathogenic
```

Example: Full process command with Parameters JSON file example and switch off query and generate a VCF file

```
howard process \
   --input=tests/data/example.vcf.gz \
   --output=/tmp/example.process.vcf \
   --param=config/param.json \
   --explode_infos \
   --query=""

howard query \
   --input=/tmp/example.process.vcf \
   --explode_infos \
   --query="SELECT \"#CHROM\", POS, ALT, REF, NOMEN, PZComment, PZFlag, PZScore \
            FROM variants \
            ORDER BY PZScore DESC"
```

```
  #CHROM       POS ALT REF                                          NOMEN
PZComment    PZFlag  PZScore
0   chr1     28736   C   A              WASH7P:NR_024540:exon1:n.50+585T>G      Described
on CLINVAR database as pathogenic      PASS     15.0
1   chr1     69101   G   A      OR4F5:NM_001005484:exon1:c.11A>G:p.Glu4Gly
DP higher than 50      PASS      5.0
2   chr7  55249063   A   G  EGFR:NM_001346897:exon19:c.2226G>A:p.Gln742Gln
DP higher than 50      PASS      5.0
3   chr1    768251   G   A         LINC01128:NR_047519:exon2:n.287+3767A>G
None      PASS      0.0
4   chr1    768252   G   A         LINC01128:NR_047519:exon2:n.287+3768A>G
None      PASS      0.0
5   chr1    768253   G   A         LINC01128:NR_047519:exon2:n.287+3769A>G
None      PASS      0.0
6   chr1     35144   C   A                 MIR1302-9:NR_036266:n.*4641A>C  Described on
CLINVAR database as non-pathogenic  FILTERED      NaN
```