HOWARD Help

Contents

1 Introduction

HOWARD:0.11.0

Highly Open Workflow for Annotation & Ranking toward genomic variant Discovery

HOWARD annotates and prioritizes genetic variations, calculates and normalizes annotations, convert on multiple formats, query variations and generates statistics

Usage examples:

howard process --input=tests/data/example.vcf.gz --output=/tmp/example.annotated.vcf.gz --param=config/param.json

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.vcf.gz --annotations='tests/databases/an

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

howard prioritization --input=tests/data/example.vcf.gz --output=/tmp/example.prioritized.vcf.gz --prioritization_config=config/prioritization_profiles.json --prioritizations='default,GERMLINE'

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'

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

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

2 QUERY tool

Query genetic variations in SQL format. Data can be loaded into 'variants' table from various formats (e.g. VCF, TSV, Parquet...). Using --explode_infos allow query on INFO/tag annotations. SQL query can also use external data within the request, such as a Parquet file(s).

Usage examples:

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

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 DP DESC'

howard query --query="SELECT "#CHROM", POS, REF, ALT, "INFO/Interpro_domain" FROM 'tests/databases/annotations/current/hg19/dbnsfp42a.parquet' WHERE "INFO/Interpro_domain" NOT NULL ORDER BY "INFO/SiPhy_29way_logOdds_rankscore" DESC LIMIT 10"

howard query --explode_infos --explode_infos_prefix='INFO/' --query="SELECT "#CHROM", POS, REF, ALT, STRING_AGG(INFO, ';') AS INFO FROM 'tests/databases/annotations/current/hg19/*.parquet' GROUP BY "#CHROM", POS, REF, ALT" --output=/tmp/full_annotation.tsv && head -n2 /tmp/full_annotation.tsv

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

2.1 Main options

```
    --input=<input>

Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --output=<output>

Output file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.

    --query=<query>

Query in SQL format
(e.g. 'SELECT * FROM variants LIMIT 50').
```

2.2 Explode

```
    --explode_infos
```

```
Explode VCF INFO/Tag into 'variants' table columns.

    --explode_infos_prefix=<explode infos prefix>

Explode VCF INFO/Tag with a specific prefix.

    --explode_infos_fields=<explode infos list> (default: *)

Explode VCF INFO/Tag specific fields/tags.
Keyword `*` specify all available fields, except those already specified.
Pattern (regex) can be used, such as `.*_score` for fields named with '_score' at the end.
Examples:
- 'HGVS,SIFT,Clinvar' (list of fields)
- 'HGVS,*,Clinvar' (list of fields with all other fields at the end)
- 'HGVS,.*_score,Clinvar' (list of 2 fields with all scores in the middle)
- 'HGVS,.*_score,*' (1 field, scores, all other fields)
- 'HGVS,*,.*_score' (1 field, all other fields, all scores)
```

2.3 Query

```
    --query_limit=<query limit> (default: 10)

Limit of number of row for query (only for print result, not output).

    --query_print_mode=<print mode> [None, 'markdown', 'tabulate', 'disabled']

Print mode of query result (only for print result, not output).
Either None (native), 'markdown', 'tabulate' or disabled.
```

2.4 Export

```
    --include_header

Include header (in VCF format) in output file.
Only for compatible formats (tab-delimiter format as TSV or BED).

    --parquet_partitions=<parquet partitions>

Parquet partitioning using hive (available for any format).
This option is faster parallel writing, but memory consuming.
Use 'None' (string) for NO partition but split parquet files into a folder.
Examples: '#CHROM', '#CHROM,REF', 'None'.
```

3 STATS tool

Statistics on genetic variations, such as: number of variants, number of samples, statistics by chromosome, genotypes by samples...

Usage examples:

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

howard stats --input=tests/data/example.vcf.gz --stats_md=/tmp/stats.md

howard stats --input=tests/data/example.vcf.gz --param=config/param.json

3.1 Main options

```
    --input=<input> | required

Input file path.
```

```
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.
```

3.2 Stats

```
    --stats_md=<stats markdown>

Stats Output file in MarkDown format.

    --stats_json=<stats json>

Stats Output file in JSON format.
```

4 CONVERT tool

Convert genetic variations file to another format. Multiple format are available, such as usual and official VCF and BCF format, but also other formats such as TSV, CSV, PSV 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.

Usage examples:

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

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

howard convert --input=tests/data/example.vcf.gz --output=/tmp/example.tsv --explode_infos --explode_infos_fields='CLNSIG,SIFT,DP' --order_by='CLNSIG DESC, DP DESC'

howard convert --input=tests/data/example.vcf.gz --output=/tmp/example.tsv --explode_infos --explode_infos_prefix='INFO/' --explode_infos_fields='CLNSIG,SIFT,DP,*' --order_by='"INFO/CLNSIG" DESC, "INFO/DP" DESC' --include_header

howard convert --input=tests/data/example.vcf.gz --output=/tmp/example.tsv --param=config/param.json

4.1 Main options

```
    --input=<input> | required

Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --output=<output> | required

Output file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.
```

4.2 Explode

```
    --explode_infos

Explode VCF INFO/Tag into 'variants' table columns.

    --explode_infos_prefix=<explode infos prefix>

Explode VCF INFO/Tag with a specific prefix.

    --explode_infos_fields=<explode infos list> (default: *)

Explode VCF INFO/Tag specific fields/tags.
Keyword `*` specify all available fields, except those already specified.
Pattern (regex) can be used, such as `.*_score` for fields named with '_score' at the end.
Examples:
- 'HGVS,SIFT,Clinvar' (list of fields)
- 'HGVS,*,Clinvar' (list of fields with all other fields at the end)
- 'HGVS,.*_score,Clinvar' (list of 2 fields with all scores in the middle)
- 'HGVS,.*_score,*' (1 field, scores, all other fields)
- 'HGVS,*,.*_score' (1 field, all other fields, all scores)
```

4.3 Export

```
    --include_header

Include header (in VCF format) in output file.
Only for compatible formats (tab-delimiter format as TSV or BED).

    --order_by=<order by>

List of columns to sort the result-set in ascending or descending order.
Use SQL format, and keywords ASC (ascending) and DESC (descending).
If a column is not available, order will not be considered.
Order is enable only for compatible format (e.g. TSV, CSV, JSON).
Examples: 'ACMG_score DESC', 'PZFlag DESC, PZScore DESC'.

    --parquet_partitions=<parquet partitions>

Parquet partitioning using hive (available for any format).
This option is faster parallel writing, but memory consuming.
Use 'None' (string) for NO partition but split parquet files into a folder.
Examples: '#CHROM', '#CHROM,REF', 'None'.
```

5 HGVS tool

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.

Usage examples:

howard hgvs --input=tests/data/example.full.vcf --output=/tmp/example.hgvs.vcf

howard hgvs --input=tests/data/example.full.vcf --output=/tmp/example.hgvs.tsv --param=config/param.json

howard hgvs --input=tests/data/example.full.vcf --output=/tmp/example.hgvs.vcf --full_format --use_exon

5.1 Main options

```
    --input=<input> | required
```

```
Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).
```

```
    --output=<output> | required
```

```
Output file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).
```

```
    --param=<param> (default: {})
```

```
Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.
```

```
    --hgvs_options=<HGVS options>
```

```
Quick HGVS annotation options.
This option will skip all other hgvs options.
Examples:
- 'default' (for default options)
- 'full_format' (for full format HGVS annotation)
- 'use_gene=True:add_protein=true:codon_type=FULL'
```

```
    --assembly=<assembly> (default: hg19)
```

```
Genome Assembly (e.g. 'hg19', 'hg38').
```

5.2 HGVS

```
    --use_gene
```

```
Use Gene information to generate HGVS annotation
(e.g. 'NM_152232(TAS1R2):c.231T>C')
```

```
    --use_exon
```

```
Use Exon information to generate HGVS annotation
(e.g. 'NM_152232(exon2):c.231T>C').
Only if 'use_gene' is not enabled.
```

```
    --use_protein
```

```
Use Protein level to generate HGVS annotation
(e.g. 'NP_689418:p.Cys77Arg').
Can be used with 'use_exon' or 'use_gene'.
```

```
    --add_protein
```

```
Add Protein level to DNA HGVS annotation (e.g 'NM_152232:c.231T>C,NP_689418:p.Cys77Arg').
```

```
    --full_format
```

```
Generates HGVS annotation in a full format
by using all information to generates an exhaustive annotation
(non-standard, e.g. 'TAS1R2:NM_152232:NP_689418:c.231T>C:p.Cys77Arg').
Use 'use_exon' to add exon information
(e.g 'TAS1R2:NM_152232:NP_689418:exon2:c.231T>C:p.Cys77Arg').
```

```
    --codon_type=<Codon type> ['1', '3', 'FULL'] (default: 3)
```

```
Amino Acide Codon format type to use to generate HGVS annotation.
Available:
```

```
- '1': codon in 1 character (e.g. 'C', 'R')
- '3': codon in 3 character (e.g. 'Cys', 'Arg')
-'FULL': codon in full name (e.g. 'Cysteine', 'Arginine')
```

```
    --refgene=<refGene>

Path to refGene annotation file.
```

```
    --refseqlink=<refSeqLink>

Path to refSeqLink annotation file.
```

6 ANNOTATION tool

Annotation is mainly based on a build-in Parquet annotation method, and tools such as BCFTOOLS, Annovar and snpEff. It uses available databases (see Annovar and snpEff) and homemade databases. Format of databases are: parquet, duckdb, vcf, bed, Annovar and snpEff (Annovar and snpEff databases are automatically downloaded, see howard databases tool).

Usage examples:

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.vcf.gz --annotations='tests/databases/an

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --assembly=hg19 --annotations='annovar:refGene,annovar:cosmic70,snpeff,tests/databases/annotations/current/hg19/clinvar_20210123.parquet'

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --assembly=hg19 --annotation_parquet='tests/databases/annotations/current/hg19/avsnp150.parquet,tests/databases/annotations/current/hg19/

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --assembly=hg19 --annotation_bcftools='tests/databases/annotations/current/hg19/nci60.vcf.gz,tests/databases/annotations/current/hg19/dbnsf

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --assembly=hg19 --annotation_snpsift='tests/databases/annotations/current/hg19/nci60.vcf.gz,tests/databases/annotations/current/hg19/dbnsfp

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --assembly=hg19 --annotation_annovar='nci60:cosmic70'

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --assembly=hg19 --annotation_snpeff='-hgvs'

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --assembly=hg19 --annotation_exomiser='preset=exome:transcript_source=refseq'

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --assembly=hg19 --annotation_splice='split_mode=one:spliceai_distance=500:spliceai_mask=1'

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --assembly=hg19 --annotations='ALL:parquet'

howard annotation --input=tests/data/example.vcf.gz --output=/tmp/example.howard.tsv --param=config/param.json

6.1 Main options

```
    --input=<input> | required

Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).
```

```
    --output=<output> | required

Output file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).
```

```
    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.

    --annotations=<annotations>

Annotation with databases files, or with tools,
as a list of files in Parquet, VCF, BED, or keywords
 (e.g. 'file.parquet,bcftools:file2.vcf.gz,annovar:refGene,snpeff').
- For a Parquet/VCF/BED, use file paths
 (e.g. 'file1.parquet,file2.vcf.gz').
- For BCFTools annotation, use keyword 'bcftools' with file paths
 (e.g. 'bcftools:file.vcf.gz:file.bed.gz').
- For Parquet annotation, use keyword 'parquet' with file paths
 (e.g. 'parquet:file.parquet').
- For Annovar annotation, use keyword 'annovar' with annovar code
 (e.g. 'annovar:refGene', 'annovar:refGene:cosmic70').
- For snpeff annotation, use keyword 'snpeff' with options
 (e.g. 'snpeff', 'snpeff:-hgvs -noShiftHgvs -spliceSiteSize 3').
- For snpSift annotation, use keyword 'snpsift' with file paths
 (e.g. 'snpsift:file.vcf.gz:file.bed.gz').
- For Exomiser annotation, use keyword 'exomiser' with options as key=value
 (e.g. 'exomiser:preset=exome:transcript_source=refseq').
- For add all availalbe databases files, use 'ALL' keyword,
 with filters on format (e.g. 'parquet', 'vcf') and release (e.g. 'current', 'devel')
 (e.g. 'ALL', ALL:format=parquet', 'ALL:format=parquet:release=current', 'ALL:format=parquet+vcf:release

    --annotation_parquet=<annotation parquet>

Annotation with Parquet method, as a list of files in Parquet, VCF or BED
 (e.g. 'file1.parquet,file2.vcf.gz').
For add all availalbe databases files, use 'ALL' keyword,
 with filters on type and release
 (e.g. 'ALL', 'ALL:parquet:current', 'ALL:parquet,vcf:current,devel').

    --annotation_bcftools=<annotation BCFTools>

Annotation with BCFTools, as a list of files VCF or BED
 (e.g. 'file.vcf.gz,file.bed.gz').

    --annotation_annovar=<annotation Annovar>

Annotation with Annovar, as a list of database keywords
 (e.g. 'refGene', 'refGene:cosmic70').

    --annotation_snpeff=<annotation snpEff>

Annotation with snpEff, with options
 (e.g. '', '-hgvs -noShiftHgvs -spliceSiteSize 3').

    --annotation_snpsift=<annotation snpSift>

Annotation with snpSift, as a list of files VCF
 (e.g. 'file.vcf.gz,file.bed.gz').

    --annotation_exomiser=<annotation Exomiser>

Annotation with Exomiser, as a list of options
 (e.g. 'preset=exome:transcript_source=refseq').

    --annotation_splice=<annotation Splice>
```

```
Annotation with Splice, as a list of options
 (e.g. 'split_mode=one:spliceai_distance=500:spliceai_mask=1').

    --assembly=<assembly> (default: hg19)

Genome Assembly (e.g. 'hg19', 'hg38').
```

6.2 Annotation

```
    --annotations_update

Update option for annotation (Only for Parquet annotation).
If True, annotation fields will be removed and re-annotated.
These options will be applied to all annotation databases.

    --annotations_append

Append option for annotation (Only for Parquet annotation).
If True, annotation fields will be annotated only if not annotation exists for the variant.
These options will be applied to all annotation databases.
```

7 CALCULATION tool

Calculation processes variants information to generate new information, such as: identify variation type (VarType), harmonizes allele frequency (VAF) and calculate sttistics (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.

Usage examples:

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

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

howard calculation --input=tests/data/example.vcf.gz --output=/tmp/example.calculated.tsv --calculations='TRIO' --trio_pedigree='sample1,sample2,sample4'

howard calculation --input=tests/data/example.vcf.gz --output=/tmp/example.calculated.tsv --calculations='BARCODEFAMII --family_pedigree='sample1,sample2,sample4'

howard calculation --input=tests/data/example.ann.transcripts.vcf.gz --output=/tmp/example.calculation.transcripts.tsv --param=config/param.transcripts.json --calculations='TRANSCRIPTS_ANNOTATIONS,TRANSCRIPTS_PRIORITIZATIO

howard calculation --input=tests/data/example.ann.vcf.gz --output=/tmp/example.ann.tsv --param=config/param.json

howard calculation --show_calculations

7.1 Main options

```
    --input=<input>

Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --output=<output>

Output file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).
```

```
    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.

    --calculations=<operations>

Quick calculations on genetic variants information and genotype information,
as a list of operations (e.g. 'VARTYPE,variant_id').
List of available calculations by default
 (unsensitive case, see doc for more information):
 VARTYPE  snpeff_hgvs  FINDBYPIPELINE  GENOTYPECONCORDANCE  BARCODE  TRIO  VAF  VAF_STATS  DP_STATS
```

7.2 Calculation

```
    --calculation_config=<calculation config>

Calculation configuration JSON file.

    --show_calculations

Show available calculation operations.
```

7.3 NOMEN

```
    --hgvs_field=<HGVS field> (default: hgvs)

HGVS INFO/tag containing a list o HGVS annotations.

    --transcripts=<transcripts>

Transcripts TSV file,
with Transcript in first column, optional Gene in second column.
```

7.4 TRIO

```
    --trio_pedigree=<trio pedigree>

Pedigree Trio for trio inheritance calculation.
Either a JSON file or JSON string or a list of samples
(e.g. 'sample1,sample2,sample3' for father, mother and child,
 '{"father": "sample1", "mother": "sample2", "child": "sample3"}').
```

7.5 BARCODEFAMILY

```
    --family_pedigree=<family pedigree>

Pedigree family for barcode calculation on genotype.
Either a JSON file or JSON string or a list of samples
(e.g. 'sample1,sample2,sample3,sample4',
 '{"father": "sample1", "mother": "sample2", "child1": "sample3", "child2": "sample3"}').
```

8 PRIORITIZATION tool

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). Multiple profiles can be used

simultaneously, which is useful to define multiple validation/prioritization levels (example: 'standard', 'stringent', 'rare variants', 'low allele frequency').

Usage examples:

howard prioritization --input=tests/data/example.vcf.gz --output=/tmp/example.prioritized.vcf.gz --prioritizations='default'

howard prioritization --input=tests/data/example.vcf.gz --output=/tmp/example.prioritized.vcf.gz --prioritizations='default,GERMLINE' --prioritization_config=config/prioritization_profiles.json

howard prioritization --input=tests/data/example.vcf.gz --output=/tmp/example.prioritized.tsv --param=config/param.json

8.1 Main options

```
    --input=<input> | required

Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --output=<output> | required

Output file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.

    --prioritizations=<prioritisations>

List of prioritization profiles to process (based on Prioritization JSON file),
such as 'default', 'rare variants', 'low allele frequency', 'GERMLINE'.
By default, all profiles available will be processed.
```

8.2 Prioritization

```
    --default_profile=<default profile>

Prioritization profile by default (see doc).
Default is the first profile in the list of prioritization profiles.

    --pzfields=<pzfields> (default: PZScore,PZFlag)

Prioritization fields to provide (see doc).
Available: PZScore, PZFlag, PZTags, PZComment, PZInfos

    --prioritization_score_mode=<prioritization score mode> ['HOWARD', 'VaRank'] (default: HOWARD)

Prioritization Score mode (see doc).
Available: HOWARD (increment score), VaRank (max score)

    --prioritization_config=<prioritization config>

Prioritization configuration JSON file (defines profiles, see doc).
```

9 PROCESS tool

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
- translates into various formats
- query genetic variants and annotations
- generates variants statistics

Usage examples:

howard process --input=tests/data/example.vcf.gz --output=/tmp/example.annotated.vcf.gz --param=config/param.json

howard process --input=tests/data/example.vcf.gz --annotations='snpeff' --calculations='snpeff_hgvs' --prioritizations='default' --explode_infos --output=/tmp/example.annotated.tsv --query='SELECT "#CHROM", POS, ALT, REF, snpeff_hgvs FROM variants'

howard process --input=tests/data/example.vcf.gz --hgvs_options='full_format,use_exon' --explode_infos --output=/tmp/example.annotated.tsv --query='SELECT "#CHROM", POS, ALT, REF, hgvs FROM variants'

howard process --input=tests/data/example.vcf.gz --output=/tmp/example.howard.vcf.gz --hgvs='full_format,use_exon' --annotations='tests/databases/annotations/current/hg19/avsnp150.parquet,tests/databases/annotations/current/hg19/dbnsfp4 --calculations='NOMEN' --explode_infos --query='SELECT NOMEN, REVEL_score, SIFT_score, AF AS 'gnomad_AF', ClinPred_score, ClinPred_pred FROM variants'

9.1 Main options

```
    --input=<input> | required

Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --output=<output> | required

Output file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.

    --hgvs_options=<HGVS options>

Quick HGVS annotation options.
This option will skip all other hgvs options.
Examples:
- 'default' (for default options)
- 'full_format' (for full format HGVS annotation)
- 'use_gene=True:add_protein=true:codon_type=FULL'

    --annotations=<annotations>

Annotation with databases files, or with tools,
as a list of files in Parquet, VCF, BED, or keywords
 (e.g. 'file.parquet,bcftools:file2.vcf.gz,annovar:refGene,snpeff').
```

```
- For a Parquet/VCF/BED, use file paths
 (e.g. 'file1.parquet,file2.vcf.gz').
- For BCFTools annotation, use keyword 'bcftools' with file paths
 (e.g. 'bcftools:file.vcf.gz:file.bed.gz').
- For Parquet annotation, use keyword 'parquet' with file paths
 (e.g. 'parquet:file.parquet').
- For Annovar annotation, use keyword 'annovar' with annovar code
 (e.g. 'annovar:refGene', 'annovar:refGene:cosmic70').
- For snpeff annotation, use keyword 'snpeff' with options
 (e.g. 'snpeff', 'snpeff:-hgvs -noShiftHgvs -spliceSiteSize 3').
- For snpSift annotation, use keyword 'snpsift' with file paths
 (e.g. 'snpsift:file.vcf.gz:file.bed.gz').
- For Exomiser annotation, use keyword 'exomiser' with options as key=value
 (e.g. 'exomiser:preset=exome:transcript_source=refseq').
- For add all availalbe databases files, use 'ALL' keyword,
 with filters on format (e.g. 'parquet', 'vcf') and release (e.g. 'current', 'devel')
 (e.g. 'ALL', ALL:format=parquet', 'ALL:format=parquet:release=current', 'ALL:format=parquet+vcf:release
```

```
    --calculations=<operations>

Quick calculations on genetic variants information and genotype information,
as a list of operations (e.g. 'VARTYPE,variant_id').
List of available calculations by default
 (unsensitive case, see doc for more information):
 VARTYPE  snpeff_hgvs  FINDBYPIPELINE  GENOTYPECONCORDANCE  BARCODE  TRIO  VAF  VAF_STATS  DP_STATS
```

```
    --prioritizations=<prioritisations>

List of prioritization profiles to process (based on Prioritization JSON file),
such as 'default', 'rare variants', 'low allele frequency', 'GERMLINE'.
By default, all profiles available will be processed.
```

```
    --assembly=<assembly> (default: hg19)

Genome Assembly (e.g. 'hg19', 'hg38').
```

9.2 HGVS

```
    --use_gene

Use Gene information to generate HGVS annotation
(e.g. 'NM_152232(TAS1R2):c.231T>C')
```

```
    --use_exon

Use Exon information to generate HGVS annotation
(e.g. 'NM_152232(exon2):c.231T>C').
Only if 'use_gene' is not enabled.
```

```
    --use_protein

Use Protein level to generate HGVS annotation
(e.g. 'NP_689418:p.Cys77Arg').
Can be used with 'use_exon' or 'use_gene'.
```

```
    --add_protein

Add Protein level to DNA HGVS annotation (e.g 'NM_152232:c.231T>C,NP_689418:p.Cys77Arg').
```

```
    --full_format

Generates HGVS annotation in a full format
```

```
by using all information to generates an exhaustive annotation
(non-standard, e.g. 'TAS1R2:NM_152232:NP_689418:c.231T>C:p.Cys77Arg').
Use 'use_exon' to add exon information
(e.g 'TAS1R2:NM_152232:NP_689418:exon2:c.231T>C:p.Cys77Arg').
```

```
    --codon_type=<Codon type> ['1', '3', 'FULL'] (default: 3)
```

```
Amino Acide Codon format type to use to generate HGVS annotation.
Available:
- '1': codon in 1 character (e.g. 'C', 'R')
- '3': codon in 3 character (e.g. 'Cys', 'Arg')
-'FULL': codon in full name (e.g. 'Cysteine', 'Arginine')
```

```
    --refgene=<refGene>
```

```
Path to refGene annotation file.
```

```
    --refseqlink=<refSeqLink>
```

```
Path to refSeqLink annotation file.
```

9.3 Annotation

```
    --annotations_update
```

```
Update option for annotation (Only for Parquet annotation).
If True, annotation fields will be removed and re-annotated.
These options will be applied to all annotation databases.
```

```
    --annotations_append
```

```
Append option for annotation (Only for Parquet annotation).
If True, annotation fields will be annotated only if not annotation exists for the variant.
These options will be applied to all annotation databases.
```

9.4 Calculation

```
    --calculation_config=<calculation config>
```

```
Calculation configuration JSON file.
```

9.5 Prioritization

```
    --default_profile=<default profile>
```

```
Prioritization profile by default (see doc).
Default is the first profile in the list of prioritization profiles.
```

```
    --pzfields=<pzfields> (default: PZScore,PZFlag)
```

```
Prioritization fields to provide (see doc).
Available: PZScore, PZFlag, PZTags, PZComment, PZInfos
```

```
    --prioritization_score_mode=<prioritization score mode> ['HOWARD', 'VaRank'] (default: HOWARD)
```

```
Prioritization Score mode (see doc).
Available: HOWARD (increment score), VaRank (max score)
```

```
    --prioritization_config=<prioritization config>
```

```
Prioritization configuration JSON file (defines profiles, see doc).
```

9.6 Query

```
    --query=<query>

Query in SQL format
(e.g. 'SELECT * FROM variants LIMIT 50').

    --query_limit=<query limit> (default: 10)

Limit of number of row for query (only for print result, not output).

    --query_print_mode=<print mode> [None, 'markdown', 'tabulate', 'disabled']

Print mode of query result (only for print result, not output).
Either None (native), 'markdown', 'tabulate' or disabled.
```

9.7 Explode

```
    --explode_infos

Explode VCF INFO/Tag into 'variants' table columns.

    --explode_infos_prefix=<explode infos prefix>

Explode VCF INFO/Tag with a specific prefix.

    --explode_infos_fields=<explode infos list> (default: *)

Explode VCF INFO/Tag specific fields/tags.
Keyword `*` specify all available fields, except those already specified.
Pattern (regex) can be used, such as `.*_score` for fields named with '_score' at the end.
Examples:
- 'HGVS,SIFT,Clinvar' (list of fields)
- 'HGVS,*,Clinvar' (list of fields with all other fields at the end)
- 'HGVS,.*_score,Clinvar' (list of 2 fields with all scores in the middle)
- 'HGVS,.*_score,*' (1 field, scores, all other fields)
- 'HGVS,*,.*_score' (1 field, all other fields, all scores)
```

9.8 Export

```
    --include_header

Include header (in VCF format) in output file.
Only for compatible formats (tab-delimiter format as TSV or BED).

    --order_by=<order by>

List of columns to sort the result-set in ascending or descending order.
Use SQL format, and keywords ASC (ascending) and DESC (descending).
If a column is not available, order will not be considered.
Order is enable only for compatible format (e.g. TSV, CSV, JSON).
Examples: 'ACMG_score DESC', 'PZFlag DESC, PZScore DESC'.

    --parquet_partitions=<parquet partitions>

Parquet partitioning using hive (available for any format).
This option is faster parallel writing, but memory consuming.
Use 'None' (string) for NO partition but split parquet files into a folder.
Examples: '#CHROM', '#CHROM,REF', 'None'.
```

10 DATABASES tool

Download databases and needed files for howard and associated tools

Usage examples:

howard databases --assembly=hg19 --download-genomes=~/howard/databases/genomes/current --download-genomes-provider=UCSC --download-genomes-contig-regex='chr[0-9XYM]+$'

howard databases --assembly=hg19 --download-annovar=~/howard/databases/annovar/current --download-annovar-files='refGene,cosmic70,nci60'

howard databases --assembly=hg19 --download-snpeff=~/howard/databases/snpeff/current

howard databases --assembly=hg19 --download-refseq=~/howard/databases/refseq/current --download-refseq-format-file='ncbiRefSeq.txt'

howard databases --assembly=hg19 --download-dbnsfp=~/howard/databases/dbnsfp/current --download-dbnsfp-release='4.4a' --download-dbnsfp-subdatabases

howard databases --assembly=hg19 --download-alphamissense=~/howard/databases/alphamissense/current

howard databases --assembly=hg19 --download-exomiser=~/howard/databases/exomiser/current

howard databases --assembly=hg19 --download-dbsnp=~/howard/databases/dbsnp/current --download-dbsnp-vcf

cd ~/howard/databases && howard databases --assembly=hg19 --download-genomes=genomes/current --download-genomes-provider=UCSC --download-genomes-contig-regex='chr[0-9XYM]+$' --download-annovar=annovar/current --download-annovar-files='refGene,cosmic70,nci60' --download-snpeff=snpeff/current --download-refseq=refseq/current --download-refseq-format-file='ncbiRefSeq.txt' --download-dbnsfp=dbnsfp/current --download-dbnsfp-release='4.4a' --download-dbnsfp-subdatabases --download-alphamissense=alphamissense/current --download-exomiser=exomiser/current --download-dbsnp=dbsnp/current --download-dbsnp-vcf --threads=8

howard databases --generate-param=/tmp/param.json --generate-param-description=/tmp/test.description.json --generate-param-formats=parquet

howard databases --input_annovar=tests/databases/others/hg19_nci60.txt --output_annovar=/tmp/nci60.from_annovar.vcf.gz --annovar_to_parquet=/tmp/nci60.from_annovar.parquet --annovar_code=nci60 --genome=~/howard/databases/genomes/cur

Notes:

- Downloading databases can take a while, depending on network, threads and memory
- Proxy: Beware of network and proxy configuration
- dbNSFP download: More threads, more memory usage (8 threads ~ 16Gb, 24 threads ~ 32Gb)

10.1 Main options

```
    --assembly=<assembly> (default: hg19)

Genome Assembly (e.g. 'hg19', 'hg38').

    --genomes-folder=<genomes> (default: ~/howard/databases/genomes/current)

Folder containing genomes.
(e.g. '~/howard/databases/genomes/current'

    --genome=<genome> (default: ~/howard/databases/genomes/current/hg19/hg19.fa)

Genome file in fasta format (e.g. 'hg19.fa', 'hg38.fa').

    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.
```

10.2 Genomes

```
    --download-genomes=<genomes>

Path to genomes folder
with Fasta files, indexes,
and all files generated by pygenome module.
(e.g. '~/howard/databases/genomes/current').

    --download-genomes-provider=<genomes provider> ['GENCODE', 'Ensembl', 'UCSC', 'NCBI'] (default: UCSC

Download Genome from an external provider.
Available: GENCODE, Ensembl, UCSC, NCBI.

    --download-genomes-contig-regex=<genomes contig regex>

Regular expression to select specific chromosome
(e.g 'chr[0-9XYM]+$').
```

10.3 snpEff

```
    --download-snpeff=<snpEff>

Download snpEff databases within snpEff folder
```

10.4 Annovar

```
    --download-annovar=<Annovar>

Path to Annovar databases
(e.g. '~/howard/databases/annovar/current').

    --download-annovar-files=<Annovar code>

Download Annovar databases for a list of Annovar file code (see Annovar Doc).
Use None to donwload all available files,
or Annovar keyword (e.g. 'refGene', 'cosmic70', 'clinvar_202*').
Note that refGene will at least be downloaded,
and only files that not already exist or changed will be downloaded.

    --download-annovar-url=<Annovar url> (default: http://www.openbioinformatics.org/annovar/download)

Annovar databases URL (see Annovar Doc).
```

10.5 refSeq

```
    --download-refseq=<refSeq>

Path to refSeq databases
(e.g. '~/howard/databases/refseq/current').

    --download-refseq-url=<refSeq url> (default: http://hgdownload.soe.ucsc.edu/goldenPath)

refSeq databases URL (see refSeq WebSite)
(e.g. 'http://hgdownload.soe.ucsc.edu/goldenPath')•/n

    --download-refseq-prefix=<refSeq prefix> (default: ncbiRefSeq)

Check existing refSeq files in refSeq folder.

    --download-refseq-files=<refSeq files> (default: ncbiRefSeq.txt,ncbiRefSeqLink.txt)

List of refSeq files to download.
```

```
    --download-refseq-format-file=<refSeq format file>

Name of refSeq file to convert in BED format
(e.g. 'ncbiRefSeq.txt').
Process only if not None.

    --download-refseq-include-utr5

Formating BED refSeq file including 5'UTR.

    --download-refseq-include-utr3

Formating BED refSeq file including 3'UTR.

    --download-refseq-include-chrM

Formating BED refSeq file including Mitochondiral chromosome 'chrM' or 'chrMT'.

    --download-refseq-include-non-canonical-chr

Formating BED refSeq file including non canonical chromosomes.

    --download-refseq-include-non-coding-transcripts

Formating BED refSeq file including non coding transcripts.

    --download-refseq-include-transcript-version

Formating BED refSeq file including transcript version.
```

10.6 dbNSFP

```
    --download-dbnsfp=<dbNSFP>

Download dbNSFP databases within dbNSFP folder(e.g. '~/howard/databases').

    --download-dbnsfp-url=<dbNSFP url> (default: https://dbnsfp.s3.amazonaws.com)

Download dbNSFP databases URL (see dbNSFP website)
(e.g. https://dbnsfp.s3.amazonaws.com').

    --download-dbnsfp-release=<dnNSFP release> (default: 4.4a)

Release of dbNSFP to download (see dbNSFP website)
(e.g. '4.4a').

    --download-dbnsfp-parquet-size=<dbNSFP parquet size> (default: 100)

Maximum size (Mb) of data files in Parquet folder.
Parquet folder are partitioned (hive) by chromosome (sub-folder),
which contain N data files.

    --download-dbnsfp-subdatabases

Generate dbNSFP sub-databases.
dbNSFP provides multiple databases which are split onto multiple columns.
This option create a Parquet folder for each sub-database (based on columns names).

    --download-dbnsfp-parquet

Generate a Parquet file for each Parquet folder.

    --download-dbnsfp-vcf

Generate a VCF file for each Parquet folder.
```

```
Need genome FASTA file (see --download-genome).

    --download-dbnsfp-no-files-all

Not generate database Parquet/VCF file for the entire database ('ALL').
Only sub-databases files will be generated.
(see '--download-dbnsfp-subdatabases').

    --download-dbnsfp-add-info

Add INFO column (VCF format) in Parquet folder and file.
Useful for speed up full annotation (all available columns).
Increase memory and space during generation of files.

    --download-dbnsfp-only-info

Add only INFO column (VCF format) in Parquet folder and file.
Useful for speed up full annotation (all available columns).
Decrease memory and space during generation of files.
Increase time for partial annotation (some available columns).

    --download-dbnsfp-uniquify

Uniquify values within column
(e.g. "D,D" to "D", "D,.,T" to "D,T").
Remove transcripts information details.
Usefull to reduce size of the database.
Increase memory and space during generation of files.

    --download-dbnsfp-row-group-size=<dnNSFP row grooup size> (default: 100000)

Minimum number of rows in a parquet row group (see duckDB doc).
Lower can reduce memory usage and slightly increase space during generation,
speed up highly selective queries, slow down whole file queries (e.g. aggregations).
```

10.7 AlphaMissense

```
    --download-alphamissense=<AlphaMissense>

Path to AlphaMissense databases

    --download-alphamissense-url=<AlphaMissense url> (default: https://storage.googleapis.com/dm_alphami

Download AlphaMissense databases URL (see AlphaMissense website)
(e.g. 'https://storage.googleapis.com/dm_alphamissense').
```

10.8 Exomiser

```
    --download-exomiser=<Exomiser>

Path to Exomiser databases
(e.g. ~/howard/databases/exomiser/current).

    --download-exomiser-application-properties=<Exomiser application properties>

Exomiser Application Properties configuration file (see Exomiser website).
This file contains configuration settings for the Exomiser tool.
If this parameter is not provided, the function will attempt to locate
the application properties file automatically based on the Exomiser.
Configuration information will be used to download expected releases (if no other parameters).
CADD and REMM will be downloaded only if 'path' are provided.

    --download-exomiser-url=<Exomiser url> (default: http://data.monarchinitiative.org/exomiser)
```

```
URL where Exomiser database files can be downloaded from
(e.g. 'http://data.monarchinitiative.org/exomiser').

    --download-exomiser-release=<Exomiser release>

Release of Exomiser data to download.
If "default", "auto", or "config", retrieve from Application Properties file.
If not provided (None), from Application Properties file (Exomiser data-version)
or default '2109'.

    --download-exomiser-phenotype-release=<Exomiser phenoptye release>

Release of Exomiser phenotype to download.
If not provided (None), from Application Properties file (Exomiser Phenotype data-version)
or Exomiser release.

    --download-exomiser-remm-release=<Exomiser remm release>

Release of ReMM (Regulatory Mendelian Mutation) database to download.
If "default", "auto", or "config", retrieve from Application Properties file.

    --download-exomiser-remm-url=<Exomiser remm url> (default: https://kircherlab.bihealth.org/download/

URL where ReMM (Regulatory Mendelian Mutation) database files can be downloaded from
(e.g. 'https://kircherlab.bihealth.org/download/ReMM').

    --download-exomiser-cadd-release=<Exomiser cadd release>

Release of CADD (Combined Annotation Dependent Depletion) database to download.
If "default", "auto", or "config", retrieve from Application Properties file.

    --download-exomiser-cadd-url=<Exomiser cadd url> (default: https://kircherlab.bihealth.org/download/

URL where CADD (Combined Annotation Dependent Depletion) database files can be downloaded from
(e.g. 'https://kircherlab.bihealth.org/download/CADD').

    --download-exomiser-cadd-url-snv-file=<Exomiser url snv file> (default: whole_genome_SNVs.tsv.gz)

Name of the file containing the SNV (Single Nucleotide Variant) data
for the CADD (Combined Annotation Dependent Depletion) database.

    --download-exomiser-cadd-url-indel-file=<Exomiser cadd url indel> (default: InDels.tsv.gz)

Name of the file containing the INDEL (Insertion-Deletion) data
for the CADD (Combined Annotation Dependent Depletion) database.
```

10.9 dbSNP

```
    --download-dbsnp=<dnSNP>

Path to dbSNP databases
(e.g. '~/howard/databases/exomiser/dbsnp').

    --download-dbsnp-releases=<dnSNP releases> (default: b156)

Release of dbSNP to download
(e.g. 'b152', 'b152,b156').

    --download-dbsnp-release-default=<dnSNP release default>

Default Release of dbSNP ('default' symlink)
(e.g. 'b156').
If None, first release to download will be assigned as default
```

```
only if it does not exists.

    --download-dbsnp-url=<dbSNP url> (default: https://ftp.ncbi.nih.gov/snp/archive)

URL where dbSNP database files can be downloaded from.
(e.g. 'https://ftp.ncbi.nih.gov/snp/archive').

    --download-dbsnp-url-files=<dbSNP url files>

Dictionary that maps assembly names to specific dbSNP URL files.
It allows you to provide custom dbSNP URL files for specific assemblies
instead of using the default file naming convention.

    --download-dbsnp-url-files-prefix=<dbSNP url files prefix> (default: GCF_000001405)

String that represents the prefix of the dbSNP file name for a specific assembly.
It is used to construct the full URL of the dbSNP file to be downloaded.

    --download-dbsnp-assemblies-map=<dbSNP assemblies map> (default: {'hg19': '25', 'hg38': '40'})

dictionary that maps assembly names to their corresponding dbSNP versions.
It is used to construct the dbSNP file name based on the assembly name.

    --download-dbsnp-vcf

Generate well-formatted VCF from downloaded file:
- Add and filter contigs associated to assembly
- Normalize by splitting multiallelics
- Need genome (see --download-genome)

    --download-dbsnp-parquet

Generate Parquet file from VCF.
```

10.10 HGMD

```
    --convert-hgmd=<HGMD>

Convert HGMD databases.
Folder where the HGMD databases will be stored.
Fields in VCF, Parquet and TSV will be generated.
If the folder does not exist, it will be created.

    --convert-hgmd-file=<HGMD file>

File from HGMD.
Name format 'HGMD_Pro_<release>_<assembly>.vcf.gz'.

    --convert-hgmd-basename=<HGMD basename>

File output basename.
Generated files will be prefixed by basename
(e.g. 'HGMD_Pro_MY_RELEASE')
By default (None), input file name without '.vcf.gz'.
```

10.11 from_Annovar

```
    --input_annovar=<input annovar>

Input Annovar file path.
Format file must be a Annovar TXT file, associated with '.idx'.

    --output_annovar=<output annovar>
```

```
Output Annovar file path.
Format file must be either VCF compressesd file '.vcf.gz'.

    --annovar_code=<Annovar code>

Annovar code, or database name.
Usefull to name databases columns.

    --annovar_to_parquet=<to parquet>

Parquet file conversion.

    --annovar_reduce_memory=<reduce memory> ['auto', 'enable', 'disable'] (default: auto)

Reduce memory option for Annovar convert,
either 'auto' (auto-detection), 'enable' or 'disable'.

    --annovar_multi_variant=<Annovar multi variant> ['auto', 'enable', 'disable'] (default: auto)

Variant with multiple annotation lines on Annovar file.
Either 'auto' (auto-detection), 'enable' or 'disable'.
```

10.12 from_extann

```
    --input_extann=<input extann>

Input Extann file path.
Format file must be a Extann TXT file or TSV file.
File need to have at least the genes column.

    --output_extann=<output extann>

Output Extann file path.
Output extann file, should be BED or BED.gz.

    --refgene=<refGene>

Path to refGene annotation file.

    --transcripts=<transcripts>

Transcripts TSV file,
with Transcript in first column, optional Gene in second column.

    --param_extann=<param extann>

Param extann file path.
Param containing configuration, options to replace chars and
bedlike header description, conf vcf specs.
(e.g. '~/howard/config/param.extann.json')

    --mode_extann=<mode extann> ['all', 'longest', 'chosen'] (default: longest)

Mode extann selection.
How to pick transcript from ncbi, keep all,
keep the longest, or keep the chosen one (transcript_extann).
```

10.13 Parameters

```
    --generate-param=<param>

Parameter file (JSON) with all databases found.
Databases folders scanned are defined in config file.
Structure of databases follow this structure (see doc):
```

```
.../<database>/<release>/<assembly>/*.[parquet|vcf.gz|...]

    --generate-param-description=<param description>

Description file (JSON) with all databases found.
Contains all databases with description of format, assembly, fields...

    --generate-param-releases=<param release> (default: current)

List of database folder releases to check
(e.g. 'current', 'latest').

    --generate-param-formats=<param formats> (default: parquet)

List of database formats to check
(e.g. 'parquet', 'parquet,vcf,bed,tsv').

    --generate-param-bcftools

Generate parameter JSON file with BCFTools annotation for allowed formats
(i.e. 'vcf', 'bed').
```

11 GUI tool

Graphical User Interface tools

Usage examples:

howard gui

12 HELP tool

Help tools

Usage examples:

howard help --help_md=docs/help.md --help_html=docs/html/help.html --help_pdf=docs/pdf/help.pdf

howard help --help_json_input=docs/json/help.configuration.json --help_json_input_title='HOWARD Configuration' --help_md=docs/help.configuration.md --help_html=docs/html/help.configuration.html --help_pdf=docs/pdf/help.configuration.pdf --code_type='json'

howard help --help_json_input=docs/json/help.parameteres.json --help_json_input_title='HOWARD Parameters' --help_md=docs/help.parameteres.md --help_html=docs/html/help.parameteres.html --help_pdf=docs/pdf/help.parameteres.pdf --code_type='json'

howard help --help_json_input=docs/json/help.parameteres.databases.json --help_json_input_title='HOWARD Parameters Databases' --help_md=docs/help.parameteres.databases.md --help_html=docs/html/help.parameteres.databases.ht --help_pdf=docs/pdf/help.parameteres.databases.pdf --code_type='json'

12.1 Main options

```
    --help_md=<help markdown>

Help Output file in MarkDown format.

    --help_html=<help html>

Help Output file in HTML format.

    --help_pdf=<help pdf>

Help Output file in PDF format.
```

```
    --help_md_input=<help MarkDown input>

Help input file in MarkDown format.

    --help_json_input=<help JSON input>

Help input file in JSON format.

    --help_json_input_title=<help JSON input title> (default: Help)

Help JSON input title.

    --code_type=<example code type>

Help example code type for input JSON format
(e.g. 'json', 'bash').
```

13 UPDATE_DATABASE tool

Update HOWARD database

Usage examples:

howard update_database --database clinvar --databases_folder /home1/DB/HOWARD --update_config update_databases.json

13.1 Main options

```
    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.
```

13.2 Update_database

```
    --databases_folder=<databases_folder> (default: ~/howard/databases)

Path of HOWARD database folder.

    --database=<database> ['clinvar'] (default: clinvar)

Which database to update.

    --update_config=<update_config>

Path of json configuration file.

    --current_folder=<current_folder> (default: current)

Path of json configuration file.
```

13.3 Options

```
    --show=<show>

None

    --limit=<limit>

None
```

14 TRANSCRIPTS_CHECK tool

Check if a transcript list is present in a generated transcript table from a input VCF file.

Usage examples:

howard transcripts_check --input=plugins/transcripts_check/tests/data/example.ann.transcripts.vcf.gz --param=plugins/transcripts_check/tests/data/param.transcripts.json --transcripts_expected=plugins/transcripts_check/tests/d --stats=/tmp/transcripts.stats.json --transcripts_missing=/tmp/transcripts.missing.tsv

14.1 Main options

```
    --input=<input> | required

Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --param=<param> (default: {}) | required

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.

    --transcripts_expected=<List of transcripts (file)> | required

File with a list of transcripts in first column.

    --transcripts_missing=<List of missing transcripts (file)>

File with a list of missing transcripts in first column.

    --stats_json=<stats json>

Stats Output file in JSON format.
```

15 GENEBE tool

GeneBe annotation using REST API (see https://genebe.net/).

Usage examples:

howard genebe --input=tests/data/example.vcf.gz --output=/tmp/example.genebe.vcf.gz --genebe_use_refseq

15.1 Main options

```
    --input=<input> | required

Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --output=<output> | required

Output file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).

    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.
```

```
    --assembly=<assembly> (default: hg19)

Genome Assembly (e.g. 'hg19', 'hg38').
```

15.2 GeneBe

```
    --genebe_use_refseq

Use refSeq to annotate (default).

    --genebe_use_ensembl

Use Ensembl to annotate.

    --not_flatten_consequences

Use exploded annotation informations.
```

15.3 Explode

```
    --explode_infos

Explode VCF INFO/Tag into 'variants' table columns.

    --explode_infos_prefix=<explode infos prefix>

Explode VCF INFO/Tag with a specific prefix.

    --explode_infos_fields=<explode infos list> (default: *)

Explode VCF INFO/Tag specific fields/tags.
Keyword `*` specify all available fields, except those already specified.
Pattern (regex) can be used, such as `.*_score` for fields named with '_score' at the end.
Examples:
- 'HGVS,SIFT,Clinvar' (list of fields)
- 'HGVS,*,Clinvar' (list of fields with all other fields at the end)
- 'HGVS,.*_score,Clinvar' (list of 2 fields with all scores in the middle)
- 'HGVS,.*_score,*' (1 field, scores, all other fields)
- 'HGVS,*,.*_score' (1 field, all other fields, all scores)
```

15.4 Export

```
    --include_header

Include header (in VCF format) in output file.
Only for compatible formats (tab-delimiter format as TSV or BED).

    --order_by=<order by>

List of columns to sort the result-set in ascending or descending order.
Use SQL format, and keywords ASC (ascending) and DESC (descending).
If a column is not available, order will not be considered.
Order is enable only for compatible format (e.g. TSV, CSV, JSON).
Examples: 'ACMG_score DESC', 'PZFlag DESC, PZScore DESC'.

    --parquet_partitions=<parquet partitions>

Parquet partitioning using hive (available for any format).
This option is faster parallel writing, but memory consuming.
Use 'None' (string) for NO partition but split parquet files into a folder.
Examples: '#CHROM', '#CHROM,REF', 'None'.
```

16 MINIMALIZE tool

Minimalize a VCF file consists in put missing value ('.') on INFO/Tags, ID, QUAL or FILTER fields. Options can also minimalize samples (keep only GT) or remove all samples. INFO/tags can by exploded before minimalize to keep tags into separated columns (useful for Parquet or TSV format to constitute a database).

Usage examples:

howard minimalize --input=tests/data/example.vcf.gz --output=/tmp/example.minimal.vcf.gz --minimalize_info --minimalize_filter --minimalize_qual --minimalize_id --minimalize_samples

howard minimalize --input=tests/data/example.vcf.gz --output=/tmp/example.minimal.tsv --remove_samples --explode_infos --minimalize_info

16.1 Main options

```
    --input=<input> | required

Input file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).
    --output=<output> | required

Output file path.
Format file must be either VCF, Parquet, TSV, CSV, PSV or duckDB.
Files can be compressesd (e.g. vcf.gz, tsv.gz).
    --param=<param> (default: {})

Parameters JSON file (or string) defines parameters to process
annotations, calculations, prioritizations, convertions and queries.
```

16.2 Minimalize

```
    --minimalize_info

Minimalize INFO field (e.g. '.' value).
    --minimalize_id

Minimalize ID field (e.g. '.' value).
    --minimalize_qual

Minimalize QUAL field (e.g. '.' value).
    --minimalize_filter

Minimalize FILTER field (e.g. '.' value).
    --minimalize_samples

Minimalize samples to keep only genotypes (i.e. 'GT').
    --remove_samples

Remove all samples to keep only variants.
```

16.3 Explode

```
    --explode_infos
```

```
Explode VCF INFO/Tag into 'variants' table columns.

    --explode_infos_prefix=<explode infos prefix>

Explode VCF INFO/Tag with a specific prefix.

    --explode_infos_fields=<explode infos list> (default: *)

Explode VCF INFO/Tag specific fields/tags.
Keyword `*` specify all available fields, except those already specified.
Pattern (regex) can be used, such as `.*_score` for fields named with '_score' at the end.
Examples:
- 'HGVS,SIFT,Clinvar' (list of fields)
- 'HGVS,*,Clinvar' (list of fields with all other fields at the end)
- 'HGVS,.*_score,Clinvar' (list of 2 fields with all scores in the middle)
- 'HGVS,.*_score,*' (1 field, scores, all other fields)
- 'HGVS,*,.*_score' (1 field, all other fields, all scores)
```

16.4 Export

```
    --include_header

Include header (in VCF format) in output file.
Only for compatible formats (tab-delimiter format as TSV or BED).

    --order_by=<order by>

List of columns to sort the result-set in ascending or descending order.
Use SQL format, and keywords ASC (ascending) and DESC (descending).
If a column is not available, order will not be considered.
Order is enable only for compatible format (e.g. TSV, CSV, JSON).
Examples: 'ACMG_score DESC', 'PZFlag DESC, PZScore DESC'.

    --parquet_partitions=<parquet partitions>

Parquet partitioning using hive (available for any format).
This option is faster parallel writing, but memory consuming.
Use 'None' (string) for NO partition but split parquet files into a folder.
Examples: '#CHROM', '#CHROM,REF', 'None'.
```

17 Shared arguments

```
    --config=<config> (default: {})

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.

    --threads=<threads> (default: -1)

Specify the number of threads to use for processing HOWARD.
It determines the level of parallelism,
either on python scripts, duckdb engine and external tools.
It and can help speed up the process/tool.
Use -1 to use all available CPU/cores.
Either non valid value is 1 CPU/core.

    --memory=<memory>
```

```
Specify the memory to use in format FLOAT[kMG]
(e.g. '8G', '12.42G', '1024M').
It determines the amount of memory for duckDB engine and external tools
(especially for JAR programs).
It can help to prevent 'out of memory' failures.
By default (None) is 80%% of RAM (for duckDB).

    --chunk_size=<chunk size> (default: 1000000)

Number of records in batch to export output file.
The lower the chunk size, the less memory consumption.
For Parquet partitioning, files size will depend on the chunk size.

    --tmp=<Temporary folder>

Temporary folder (e.g. '/tmp').
By default, '.tmp' for duckDB (see doc),external tools and python scripts.

    --duckdb_settings=<duckDB settings>

DuckDB settings (see duckDB doc) as JSON (string or file).
These settings have priority (see options 'threads', 'tmp'...).
Examples: '{"TimeZone": "GMT", "temp_directory": "/tmp/duckdb", "threads": 8}'.

    --verbosity=<verbosity> ['CRITICAL', 'ERROR', 'WARNING', 'INFO', 'DEBUG', 'NOTSET'] (default: INFO)

Verbosity level
Available: CRITICAL, ERROR, WARNING, INFO, DEBUG or NOTSET
- DEBUG: Detailed information, typically of interest only when diagnosing problems.
- INFO: Confirmation that things are working as expected.
- WARNING: An indication that something unexpected happened.
- ERROR: Due to a more serious problem.
- CRITICAL: A serious error.
- NOTSET: All messages.

    --log=<log>

Logs file
(e.g. 'my.log').

    --quiet

==SUPPRESS==

    --verbose

==SUPPRESS==

    --debug

==SUPPRESS==
```