HOWARD Parameters

HOWARD Parameters JSON file defines parameters to process annotations, calculations, prioritizations, convertions and queries.

Table of contents

Examples:

> Parameters for annotation, calculation, prioritization, HGVS annotation and export

```
{
   "hgvs": {
      "full_format": true,
      "use_exon": true
   }
   "annotation": {
      "parquet": {
         "annotations": {
            "/path/to/database3.parquet": {
               "field1": null,
               "field2": "field2_renamed"
            },
            "/path/to/database4.vcf.gz": {
               "INFO": null
            },
            "/path/to/database5.bed.gz": {
               "INFO": null
            }
         }
      },
      "bcftools": {
         "annotations": {
            "/path/to/database6.vcf.gz": {
               "field1": null,
               "field2": "field2_renamed"
            },
            "/path/to/database7.bed": {
               "INFO": null
            }
         }
      },
      "annovar": {
         "annotations": {
            "annovar_keyword2": {
               "field1": null,
               "field2": "field2_renamed"
            },
            "annovar_keyword3": {
               "INFO": null
            }
         }
      },
      "snpeff": {
         "options": " -hgvs -noShiftHgvs -spliceSiteSize 3 -lof -oicr "
      },
      "exomiser": {
         "release": "2109",
         "transcript_source": "refseq",
         "hpo": ["HP:0001156", "HP:0001363", "HP:0011304", "HP:0010055"]
      },
      "options": {
         "append": true
      }
   },
   "calculation": {
      "operation1": null,
      "operation2": {
        "options": {
          "option1": "value1",
          "option2": "value2"
        }
      }
   },
   "prioritization": {
      "prioritizations": "config/prioritization_profiles.json",
      "profiles": ["GENOME", "GERMLINE"],
      "default_profile": "GERMLINE",
      "pzfields": ["PZScore", "PZFlag", "PZComment"],
      "prioritization_score_mode": "VaRank"
   },
   "export": {
      "include_header": true
   }
}
```

hgvs

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. Several options are available, to add gene, exon and protein information, to generate a 'full format' detailed annotation, to choose codon format.

Examples:

> HGVS annotation with operations for generate variant_id and variant type, extract HGVS from snpEff annotation, select NOMEN from snpEff HGVS with a list of transcripts of preference

```
"hgvs": {
  "full_format": true,
  "use_exon": true
}
```

hgvs::use_gene

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

Default: `False`

Examples:

> Use Gene in HGVS annotation

```
"use_gene": true
```

hgvs::use_exon

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

Default: `False`

Examples:

> Use Exon in HGVS annotation

```
"use_exon": true
```

hgvs::use_protein

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

Default: `False`

Examples:

> Use Protein in HGVS annotation

```
"use_protein": true
```

hgvs::add_protein

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

Default: `False`

Examples:

> Add Protein level to DNA HGVS annotation

```
"add_protein": true
```

hgvs::full_format

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

Default: `False`

Examples:

> Use full format for HGVS annotation

```
"full_format": true
```

hgvs::codon_type

Amino Acide Codon format type to use to generate HGVS annotation (default '3'):

- '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')

Type: `str`

Choices: `['1', '3', 'FULL']`

Default: `3`

Examples:

> Amino Acide Codon format with 1 character

```
"codon_type": "1"
```

> Amino Acide Codon format with 3 character

```
"codon_type": "3"
```

> Amino Acide Codon format with full name

```
"codon_type": "FULL"
```

hgvs::refgene

Path to refGene annotation file (see HOWARD User Guide).

Type: `Path`

Default: `None`

Examples:

> Path to refSeq file

```
"refgene": "~/howard/databases/refseq/current/hg19/ncbiRefSeq.txt"
```

hgvs::refseqlink

Path to refGeneLink annotation file (see HOWARD User Guide).

Type: `Path`

Default: `None`

Examples:

> Path to refSeq file

```
"refseqlink": "~/howard/databases/refseq/current/hg19/ncbiRefSeqLink.txt"
```

annotation

Annotation process using HOWARD algorithms or external tools.

For HOWARD Parquet algorithm, provide the list of database files available (formats such as Parquet, VCF, TSV, duckDB, JSON) and select fields (rename possible, 'INFO' keyword for all fields), or use 'ALL' keyword to detect available databases.

For external tools, such as Annovar, snpEff and Exomiser, specify parameters such as annotation keywords (Annovar) and options (depending on the tool), and select fields (BCFtools and Annovar, field rename available).

Examples:

> Annotation with multiple tools in multiple formats with multiple options

```
"annotation": {
   "parquet": {
      "annotations": {
         "/path/to/database3.parquet": {
            "field1": null,
            "field2": "field2_renamed"
         },
         "/path/to/database4.vcf.gz": {
            "INFO": null
         }
         "/path/to/database5.bed.gz": {
            "INFO": null
         }
      }
   }
   "bcftools": {
      "annotations": {
         "/path/to/database6.vcf.gz": {
            "field1": null,
            "field2": "field2_renamed"
         },
         "/path/to/database7.bed": {
            "INFO": null
         }
      }
   }
   "annovar": {
      "annotations": {
         "annovar_keyword2": {
            "field1": null,
            "field2": "field2_renamed"
         },
         "annovar_keyword3": {
            "INFO": null
         }
      }
   }
   "snpeff": {
      "options": " -hgvs -noShiftHgvs -spliceSiteSize 3 -lof -oicr "
   }
   "exomiser": {
      "release": "2109",
      "transcript_source": "refseq",
      "hpo": ["HP:0001156", "HP:0001363", "HP:0011304", "HP:0010055"]
   }
   "options": {
      "append": true
   }
}
```

annotation::parquet

Annotation process using HOWARD Parquet algorithm, for the list of databases available (formats such as Parquet, VCF, TSV, duckDB, JSON).

Examples:

> Annotation with multiple databases in multiple formats

```
"parquet": {
   "annotations": {
      "/path/to/database3.parquet": {
         "field1": null,
         "field2": "field2_renamed",
      },
      "/path/to/database4.vcf.gz": {
         "INFO": null
      }
      "/path/to/database5.bed.gz": {
         "INFO": null
      }
   }
}
```

annotation::parquet::annotations

Specify a list of databases files available (formats such as Parquet, VCF, TSV, duckDB, JSON). This parameter enables users to select specific database fields and optionally rename them (e.g. '"field": null' to keep field name, '"field": "new_name"' to rename field). Use 'INFO' keyword to select all fields within the database INFO/Tags header (e.g. '"INFO": null'). Use 'ALL' keyword to select all fields within the database regardless INFO/Tags header (e.g. '"ALL": null').

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').

If a full path is not provided, the system will automatically detect files within database folders (see Configuration doc) and assembly (see Parameter option).

Examples:

> Annotation with dbSNP database with INFO/tags fields, and dbNSFP databases with all fields

```
"annotations": {
   "tests/databases/annotations/current/hg19/avsnp150.parquet": {
      "INFO": null
   }
   "tests/databases/annotations/current/hg19/dbnsfp42a.parquet": {
      "ALL": null
   }
}
```

> Annotation with dbNSFP (only PolyPhen, ClinVar and REVEL score), and rename fields

```
"annotations": {
   "tests/databases/annotations/current/hg19/dbnsfp42a.parquet": {
      "Polyphen2_HDIV_pred": "PolyPhen",
      "ClinPred_pred": "ClinVar",
      "REVEL_score": null
   }
}
```

> Annotation with refSeq as a BED file

```
"annotations": {
   "tests/databases/annotations/current/hg19/refGene.bed": {
      "INFO": null
   }
}
```

> Annotation with dbNSFP REVEL annotation (as a VCF file) within configured annotation databases folders (default: '~/howard/databases/annotations/current') and assembly (default: 'hg19')

```
"annotations": {
   "dbnsfp42a.REVEL.vcf.gz": {
      "REVEL_score": null,
      "REVEL_rankscore": null
   }
}
```

> Annotation with all available databases in Parquet for ''current release

```
"parquet": {
   "annotations": {
      "ALL": {
         "formats": ["parquet"],
         "releases": ["current"]
      }
   }
}
```

annotation::bcftools

Annotation process using BCFTools. Provide a list of database files and annotation fields.

Examples:

> Annotation with multiple databases in multiple formats

```
"parquet": {
   "bcftools": {
      "/path/to/database1.vcf.gz": {
         "field1": null,
         "field2": "field2_renamed"
      },
      "database2.bed.gz": {
         "INFO": null
      }
   }
}
```

annotation::bcftools::annotations

Specify the list of database files in formats VCF or BED. Files need to be compressed and indexed.

This parameter enables users to select specific database fields and optionally rename them (e.g. '"field": null' to keep field name, '"field": "new_name"' to rename field). Use 'INFO' or 'ALL' keyword to select all fields within the database INFO/Tags header (e.g. '"INFO": null', '"ALL": null').

If a full path is not provided, the system will automatically detect files within database folders (see Configuration doc) and assembly (see Parameter option).

Examples:

> Annotation with dbSNP database with all fields

```
"annotations": {
   "tests/databases/annotations/current/hg19/avsnp150.vcf.gz": {
      "INFO": null
   }
}
```

> Annotation with dbNSFP (only PolyPhen, ClinVar and REVEL score), and rename fields

```
"annotations": {
   "tests/databases/annotations/current/hg19/dbnsfp42a.vcf.gz": {
      "Polyphen2_HDIV_pred": "PolyPhen",
      "ClinPred_pred": "ClinVar",
      "REVEL_score": null
   }
}
```

> Annotation with refSeq as a BED file

```
"annotations": {
   "tests/databases/annotations/current/hg19/refGene.bed": {
      "INFO": null
   }
}
```

> Annotation with dbNSFP REVEL annotation (as a VCF file) within configured annotation databases folders (default: '~/howard/databases/annotations/current') and assembly (default: 'hg19')

```
"annotations": {
   "dbnsfp42a.REVEL.vcf.gz": {
      "REVEL_score": null,
      "REVEL_rankscore": null
   }
}
```

annotation::annovar

Annotation process using Annovar tool. Provides a list of keywords to select Annovar databases, and defines Annovar options (see Annovar documentation).

Examples:

> Annotation with multiple Annovar databases, with fields selection, and Annovar options

```
"annovar": {
   "annotations": {
      "annovar_keyword2": {
         "field1": null,
         "field2": "field2_renamed",
      },
      "annovar_keyword3": {
         "INFO": null
      }
```

```
    }
}
```

annotation::annovar::annotations

List of keywords refering to Annovar databases (see Annovar Databases documentation), with a list of selected fields for each of them (rename available)

Examples:

> Annotation with ClinVar (fields CLNSIG and CLNDN renamed) and Cosmic (all fields)

```
"annotations": {
   "clinvar_20221231": {
      "CLNSIG": "ClinVar_class"
      "CLNDN": "ClinVar_desease",
   },
   "cosmic70": {
      "INFO": null
   },
}
```

annotation::annovar::options

List of options available with Annovar tool (see Annovar documentation). As example, these options allows to define splicing threshold or HGVS annotation with refGene database

Examples:

> HGVS Annotation with refGene (add 'refGene' to 'annotations') and a splicing threshold as 3

```
"options": {
   "splicing_threshold": 3,
   "argument": "'-hgvs'"
   }
}
```

annotation::snpeff

Annotation process using snpEff tool and options (see snpEff documentation).

Examples:

> Annotation with snpEff databases, with options for HGVS annotation and additional tags.

```
"snpeff": {
   "options": " -hgvs -noShiftHgvs -spliceSiteSize 3 -lof -oicr "
}
```

annotation::snpeff::options

String (as command line) of options available such as:

- filters on variants (regions filter, specific changes as intronic or downstream)
- annotation (e.g. HGVS, loss of function)
- database (e.g. only protein coding transcripts, splice sites size)

Examples:

> Annotation with snpEff databases, with options to generate HGVS annotation, specify to not shift variants according to HGVS notation, define splice sites size to 3, add loss of function (LOF), Nonsense mediated decay and OICR tags.

```
"options": " -hgvs -noShiftHgvs -spliceSiteSize 3 -lof -oicr "
```

annotation::snpeff::stats

HTML file for snpEff stats. Use keyword 'OUTPUT' to generate file according to output file.

Examples:

> Annotation with snpEff databases, and generate a specific stats in HTML format.

```
"stats": "/path/to/stats.html"
```

> Annotation with snpEff databases, and generate stats in HTML format associated with output file.

```
"stats": "OUTPUT.html"
```

annotation::snpeff::csvStats

CSV file for snpEff stats. Use keyword 'OUTPUT' to generate file according to output file.

Examples:

> Annotation with snpEff databases, and generate a specific stats in CSV format.

```
"csvStats": "/path/to/stats.csv"
```

> Annotation with snpEff databases, and generate stats in CSV format associated with output file.

```
"csvStats": "OUTPUT.csv"
```

annotation::snpsift

Annotation process using snpSift. Provide a list of database files and annotation fields.

Examples:

> Annotation with multiple databases in multiple formats

```
"parquet": {
   "snpsift": {
      "/path/to/database1.vcf.gz": {
         "field1": null,
         "field2": null
      },
      "/path/to/database2.vcf.gz": {
         "field1": null,
         "field2": null
      }
   }
}
```

annotation::snpsift::annotations

Specify the list of database files in formats VCF. Files need to be compressed and indexed.

This parameter enables users to select specific database fields and optionally rename them (e.g. '"field": null' to keep field name, '"field": "new_name"' to rename field). Use 'INFO' or 'ALL' keyword to select all fields within the database INFO/Tags header (e.g. '"INFO": null', '"ALL": null').

If a full path is not provided, the system will automatically detect files within database folders (see Configuration doc) and assembly (see Parameter option).

Examples:

> Annotation with dbSNP database with all fields

```
"annotations": {
   "tests/databases/annotations/current/hg19/avsnp150.vcf.gz": {
      "INFO": null
   }
}
```

> Annotation with dbNSFP (only PolyPhen, ClinVar and REVEL score)

```
"annotations": {
   "tests/databases/annotations/current/hg19/dbnsfp42a.vcf.gz": {
      "Polyphen2_HDIV_pred": "PolyPhen",
      "ClinPred_pred": "ClinVar",
      "REVEL_score": null
   }
}
```

> Annotation with dbNSFP REVEL annotation (as a VCF file) within configured annotation databases folders (default: '~/howard/databases/annotations/current') and assembly (default: 'hg19')

```
"annotations": {
   "dbnsfp42a.REVEL.vcf.gz": {
      "REVEL_score": null,
      "REVEL_rankscore": null
   }
}
```

annotation::exomiser

Annotation process using Exomiser tool and options (see Exomiser website documentation).

Examples:

> Annotation with Exomiser, using database release '2109', transcripts source as UCSC and a list of HPO terms.

```
"exomiser": {
   "release": "2109"
   "transcript_source": "refseq"
   "hpo": ["HP:0001156", "HP:0001363", "HP:0011304", "HP:0010055"]
}
```

annotation::exomiser::release

Release of Exomiser database. This option replace the release variable in 'application.properties' file (see 'exomiser_application_properties' option). The release will be downloaded if it is not available locally.

Examples:

> Annotation with release '2109' of Exomiser database.

```
"release": "2109"
```

annotation::exomiser::transcript_source

Transcription source of Exomiser. This option replace the release variable in 'application.properties' file (see 'exomiser_application_properties' option). The release will be downloaded if it is not available locally.

Examples:

> Annotation with transcription source 'refseq' of Exomiser.

```
"transcript_source": "refseq"
```

annotation::exomiser::hpo

List of HPO for Exomiser. This option replace the release variable in 'application.properties' file (see 'exomiser_application_properties' option). The release will be downloaded if it is not available locally.

Examples:

> Annotation with a list of 4 HPO for Exomiser.

```
"hpo": ["HP:0001156", "HP:0001363", "HP:0011304", "HP:0010055"]
```

annotation::splice

Annotation process using Splice tool and options. This annotation will be proccessed only for variants that are not already annotated (i.e. without annotation like 'SpliceAI_\w+' and 'SPiP_\w+')

Examples:

> Annotation with Splice, using database splice mode ('one'), spliceAI distance (500) and spliceAI mask (1).

```
"splice": {
   "split_mode": "one",
   "spliceai_distance": 500,
   "spliceai_mask": 1
}
```

annotation::splice::split_mode

Split mode of Exomiser database (default 'one'):

- all: report all annotated transcript for one gene.
- one: keep only the transcript with the most pathogenic score (in case of identical score, take the first).
- list: keep transcript provided in transcript file, if no matching transcript in file 'one' mode is activated.
- mixed: 'one' mode, if identical score, list mode is activated.

Examples:

> Split mode to report all annotated transcript for one gene.

```
"split_mode": "all"
```

annotation::splice::spliceai_distance

Maximum distance between the variant and gained/lost splice site (default: 500).

Examples:

> Maximum distance of '500' between variant and splice site.

```
"spliceai_distance": 500
```

annotation::splice::spliceai_mask

Mask scores representing annotated acceptor/donor gain and unannotated acceptor/donor loss (default: 1).

Examples:

> Mask score of '1' for acceptor/donor gain fain and loss.

```
"spliceai_mask": 1
```

annotation::splice::transcript

Path to a list of transcripts of preference (default '').

Examples:

> Path to file of transcripts.

```
"transcript": "tests/data/transcripts.tsv"
```

annotation::splice::rm_snps

Do not consider SNV for the analysis, only Indels and MNV (default 'false').

Examples:

> Analysing only non SNV.

```
"rm_snps": "true"
```

annotation::splice::rm_annot

Remove existing annotation before analysing (default 'true').

Examples:

> Remove annotation before analysing.

```
"rm_annot": "true"
```

annotation::splice::whitespace

Remove spaces in INFO field, 'true' to remove (default 'true').

Examples:

> Remove spaces in INFO field.

```
"whitespace": "true"
```

annotation::options

Options for annotations, such as annotation strategy (skip if exists, update, append)

Examples:

> Annotation with Parquet databases, with update annotation strategy.

```
"options": {
   "update": true
}
```

annotation::options::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.

Default: `False`

Examples:

> Apply update on all annotation fields for all databases.

```
"update": true
```

annotation::options::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.

Default: `False`

Examples:

> Apply append on all annotation fields for all databases.

```
"append": true
```

calculation

Calculation process operations that are defiend in a Calculation Configuration JSON file. List available calculation operations with possible options (see Calculation JSON file help).

Examples:

> Calculation of operations 'operation1' and 'operation2' (with options) defined in 'calculation_config.json' file

```
"calculation": {
  "calculations": {
    "operation1": null,
    "operation2": {
      "options": {
        "option1": "value1",
        "option2": "value2"
      }
    }
  },
  "calculation_config": "calculation_config.json"
}
```

calculation::calculations

List of operations to process with possible options (see Calculation JSON file help).

Examples:

> Calculation with operations for generate variant_id and variant type, extract HGVS from snpEff annotation, select NOMEN from snpEff HGVS with a list of transcripts of preference

```
"calculations": {
  "variant_id": null,
  "vartype": null,
  "snpeff_hgvs": null,
  "NOMEN": {
    "options": {
      "hgvs_field": "snpeff_hgvs",
      "transcripts": "tests/data/transcripts.tsv"
    }
  }
}
```

calculation::calculation_config

Calculation configuration JSON file.

Type: `Path`

Default: `None`

Examples:

> Calculation configuration JSON file as an option

```
"calculation_config": "calculation_config.json"
```

prioritization

Prioritization process use a JSON configuration file that defines all profiles that can be used. By default, all profiles will be calculated from the JSON configuration file, and the first profile will be considered as default. Proritization annotations (INFO/tags) will be generated depending of a input list (default 'PZScore' and 'PZFlag'), for all profiles (e.g. 'PZScore_GERMLINE' for 'GERMLINE' profile) and for default profile (e.g. 'PZScore' for default). Prioritization score mode is 'HOWARD' by default.

Examples:

> Prioritization with 'GENOME' and 'GERMLINE' (default) profiles, from a list of configured profiles, only 3 prioritization fields returned, and score calculated in 'VaRank' mode

```
"prioritization": {
  "prioritizations": "config/prioritization_profiles.json",
  "profiles": ["GENOME", "GERMLINE"],
  "default_profile": "GERMLINE",
  "pzfields": ["PZScore", "PZFlag", "PZComment"],
  "prioritization_score_mode": "VaRank"
}
```

prioritization::prioritizations

Prioritization configuration profiles JSON file defining profiles to calculate. All configured profiles will be calculated by default (see 'profiles' parameter). First profile will be considered as 'default' if none are provided (see 'default_profile' parameter). Default score calculation mode is 'HOWARD'. This option refers to the quick prioritization command line parameter `--prioritizations`.

Type: `str`

Default: `None`

Examples:

> Prioritization configuration profile JSON file

```
"prioritizations": "config/prioritization_profiles.json"
```

prioritization::profiles

Prioritization profiles to consider, from the list of configured profiles. If empty, all configured profiles will be calculated. First profile will be considered as 'default' if none are provided (see 'default_profile' parameter). Prioritization annotations (INFO/tags) will be generated for all these profiles (e.g. 'PZScore_GERMLINE' for 'GERMLINE' profile).

Type: `str`

Default: `None`

Examples:

> Prioritization with 'GERMLINE' profile only

```
"profiles": ["GERMLINE"]
```

> Prioritization with 'GENOME' and 'GERMLINE' profiles

```
"profiles": ["GENOME", "GERMLINE"]
```

prioritization::default_profile

Prioritization default profile from the list of processed profiles. Prioritization annotations (INFO/tags) will be generated for this default profile (e.g. 'PZScore', 'PZFlags').

Type: `str`

Default: `None`

Examples:

> Prioritization default profile 'GERMLINE'

```
"default_profile": "GERMLINE"
```

prioritization::pzfields

Prioritization annotations (INFO/tags) to generate. By default 'PZScore', 'PZFlags'.

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

Type: `str`

Default: `PZScore,PZFlag`

Examples:

> Prioritization annotations (INFO/tags) list

```
"pzfields": ["PZScore", "PZFlag", "PZComment"]
```

prioritization::prioritization_score_mode

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.

Type: `str`

Choices: `['HOWARD', 'VaRank']`

Default: `HOWARD`

Examples:

> Prioritization score calculation mode 'HOWARD'

```
"prioritization_score_mode": "HOWARD"
```

> Prioritization score calculation mode 'VaRank'

```
"prioritization_score_mode": "VaRank"
```

stats

Statistics on loaded variants.

stats::stats_md

Stats Output file in MarkDown format.

Type: `Path`

Default: `None`

Examples:

> Export statistics in Markdown format

```
"stats_md": "/tmp/stats.md"
```

stats::stats_json

Stats Output file in JSON format.

Type: `Path`

Default: `None`

Examples:

> Export statistics in JSON format

```
"stats_json": "/tmp/stats.json"
```

query

Query options tools. Mainly a SQL query, based on 'variants' table corresponding on input file data, or a independant query. Print options for 'query' tool allow limiting number of lines and choose printing mode.

Type: `str`

Default: `None`

query::query

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

Type: `str`

Default: `None`

Examples:

> Simple query to show all variants file

```
SELECT "#CHROM", POS, REF, ALT, INFO
FROM variants
```

query::query_limit

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

Type: `int`

Default: `10`

query::query_print_mode

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

Type: `str`

Choices: `[None, 'markdown', 'tabulate']`

Default: `None`

export

Export options for output files, such as data order, include header in output and hive partitioning.

export::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'.

Type: `str`

Default:

Examples:

> Order by ACMG score in descending order

```
"order_by": "ACMG_score DESC"
```

> Order by PZFlag and PZScore in descending order

```
"order_by": PZFlag DESC, PZScore DESC"
```

export::include_header

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

Default: `False`

export::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'.

Type: `str`

Default: `None`

explode

Explode options for INFO/tags annotations within VCF files.

explode::explode_infos

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

Default: `False`

explode::explode_infos_prefix

Explode VCF INFO/Tag with a specific prefix.

Type: `str`

Default:

explode::explode_infos_fields

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)

Type: `str`

Default: `*`

transcripts

Transcripts information to create transcript view. Useful to add transcripts annotations in INFO field, to calculate transcripts specific scores (see Calculation JSON file help).

Type: `dict`

Default: `None`

Examples:

> Trancripts information from snpEff and dbNSFP annotation

```
"transcripts": {
  "table": "transcripts",
  "transcripts_info_field": "transcripts_json",
  "transcripts_info_json": "transcripts_json",
  "struct": {
      "from_column_format": [
          {
              "transcripts_column": "ANN",
              "transcripts_infos_column": "Feature_ID"
          }
      ],
      "from_columns_map": [
          {
              "transcripts_column": "Ensembl_transcriptid",
              "transcripts_infos_columns": [
                  "genename",
                  "Ensembl_geneid",
                  "LIST_S2_score",
                  "LIST_S2_pred"
              ]
          },
          {
              "transcripts_column": "Ensembl_transcriptid",
              "transcripts_infos_columns": [
                  "genename",
                  "VARITY_R_score",
                  "Aloft_pred"
              ]
          }
      ]
  }
}
```

transcripts::table

Transcripts table name to create.

Examples:

> Name of transcript table:

```
"table": "transcripts"
```

transcripts::transcripts_info_field

Transcripts INFO field name to add in VCF INFO field.

Examples:

> Transcripts INFO field name:

```
"transcripts_info_field": "transcripts_json"
```

transcripts::transcripts_info_json

Transcripts column name to add to transcripts table.

Examples:

> Transcripts column name:

```
"transcripts_info_json": "transcripts_json"
```

transcripts::struct

Structure of transcripts information, corresponding to columns dedicated to transcripts, such as :

- a uniq annotation field with a specific format (from_column_format) like snpEff annotation,
- a list of annotation fields corresponding to transcripts in another specific field (from_columns_map), like dbNSFP annotation.

transcripts::struct::from_column_format

Structure of transcripts information from a uniq annotation field with a specific format (such as snpEff annotation):

- 'transcripts_column' correspond to INFO field with annotations
- 'transcripts_infos_column' correspond to transcription ID annotations field within INFO field

Examples:

> Structure from snpEff annotation:

```
"from_column_format": [
  {
    "transcripts_column": "ANN",
    "transcripts_infos_column": "Feature_ID"
  }
]
```

transcripts::struct::from_columns_map

list of annotation fields corresponding to transcripts in another specific field (such as dbNSFP annotation):

- 'transcripts_column' correspond to INFO field with transcription ID
- 'transcripts_infos_columns' correspond to a list of INFO fields with transcript information

Examples:

> Structure from snpEff annotation:

```
"from_columns_map": [
  {
    "transcripts_column": "Ensembl_transcriptid",
    "transcripts_infos_columns": [
      "genename",
      "Ensembl_geneid",
      "LIST_S2_score",
      "LIST_S2_pred"
    ]
  }
]
```

threads

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.

Type: `int`

Default: `-1`

Examples:

> Automatically detect all available CPU/cores

```
"threads": -1
```

> Define 8 CPU/cores

```
"threads": 8
```

databases

HOWARD Parameters Databases JSON describes configuration JSON file for databases download and convert.