Jannovar Documentation

Peter N Robinson, Marten Jaeger, Manuel Holtgrewe, Max Schuba

Sep 06, 2021

Installation Getting Started

Jannovar is a Java-based program and library for the functional annotation of VCF files. The documentation is split into five parts (accessible through the navigation on the left).

Installation & Getting Started Instructions for the Installation of the program and some examples to get you started.

Jannovar Usage An overview of how Jannovar works and documentation and examples for the Jannovar sub commands. The original and prime feature of Jannovar predicting molecular impact of variants given a transcriptome model. Further features such as the conversion between HGVS and VCF are also described.

Further Annotation This part documents the annotation with different databases and compatible modes of inheritance. Also, it quickly describes how to (hard-)filter your VCF files given annotations using `bcftools`.

Tips & Tricks Further documentation on Jannovar, information on troubleshooting etc.

Project Information More information on the project, including the changelog, list of contributing authors, and contribution instructions.

CHAPTER 1

Quick Example

```
$ java -jar jannovar-0.36.jar \
    download -d hg19/refseq
[...]
$ java -jar jannovar-0.36.jar \
    annotate-vcf -d data/hg19_refseq.ser -i IN.vcf.gz -o OUT.vcf.gz
```

CHAPTER 2

Features

- annotation of VCF files for functional impact, supporting different transcript databases (RefSeq, ENSEMBL, UCSC)
- annotation with information from dbSNP, ExAC, UK10K, ...
- use Sequence Ontology for variant effect annotation
- API Javadoc for the library is here: https://javadoc.io/doc/de.charite.compbio/jannovar-core/0.36/.
- ... and more

CHAPTER 3

Feedback

The best place to leave feedback, ask questions, and report bugs is the Jannovar Issue Tracker.

CHAPTER 4

API Documentation

The friendly people at javadoc.io host our API documentation:

jannovar-core https://javadoc.io/doc/de.charite.compbio/jannovar-core/0.36/

jannovar-hgvs https://javadoc.io/doc/de.charite.compbio/jannovar-hgvs/0.36/

jannovar-htsjdk https://javadoc.io/doc/de.charite.compbio/jannovar-htsjdk/0.36/

jannovar-vardbs https://javadoc.io/doc/de.charite.compbio/jannovar-vardbs/0.36/

4.1 Quickstart

This short How-To guides you from downloading the Jannovar program to annotating a VCF file in 4 steps.

1. Download the current stable release from our GitHub project by clicking 0.360.36.
2. Extract the ZIP archive.
   - you should find file called 0.36 in the ZIP
   - you should also find a file `small.vcf` file in the folder `examples`
3. Download the RefSeq transcript database for the release *hg19/GRCh37*.

   Note: If you are behind a proxy then you have to pass its path to the `--proxy` option, e.g., `--proxy http://proxy.example.com:8080`. See the section *Proxy Settings* for more information.

   ```
   $ java -jar jannovar-cli-0.36.jar download -d hg19/refseq
   ```

   This will create the file `data/hg19_refseq.ser` which is a self-contained transcript database and can be used for functional annotation.
4. Annotate the file `small.vcf` from the `examples` directory.

```
$ java -jar jannovar-cli-0.36.jar annotate -d data/hg19_refseq.ser -i␣
↪examples/small.vcf
```

Jannovar will now load the transcript database from `data/hg19_refseq.ser` and then read `examples/small.vcf` file. Each contained variant in this file will be annotated with an `EFFECT` and an `HGVS` field in the `VCF` info column. The `EFFECT` field contains an effect, e.g., `SYNONYMOUS` and the `HGVS` field contains a HGVS representation of the variant. The result will be written out to `small.jv.vcf`.

Note: The variant effect codes in the output and their mapping to sequence ontology is described in the Jannovar API documentation.

The following excerpt shows the first three variants of the `small.vcf` file with their effect and HGVS annotation.

```
1 866511  rs60722469      C       CCCCT   258.62  PASS    EFFECT=INTRONIC;
↪HGVS=SAMD11:NM_152486.2:c.305+42_305+43insCCCT  GT:AD:DP:GQ:PL  1/1:6,5:11:14.
↪79:300,15,0
1 879317  rs7523549       C       T       150.77  PASS    EFFECT=MISSENSE;
↪HGVS=SAMD11:XM_005244727.1:exon9:c.799C>T:p.Arg267Cys   GT:AD:DP:GQ:PL  0/1:14,
↪7:21:99:181,0,367
1 879482  .       G       C       484.52  PASS    EFFECT=MISSENSE;HGVS=SAMD11:XM_
↪005244727.1:exon9:c.964G>C:p.Asp322His   GT:AD:DP:GQ:PL  0/1:28,20:48:99:515,0,
↪794
```

4.1.1 Next Steps

Of course, you can follow the other manual chapters and get more extensive information on Jannovar. In addition, here are some external links that can help you in your understanding:

Current VCF Specification can be found in the **hts-specs** project on GitHub.

HGVS Mutation Nomenclature. is mainainted by the Human Genome Variation Society and the nomenclature can be found in the Sequence Variant Nomenclature.

4.2 Installation

4.2.1 Pre-built Binaries

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

Pre-built binaries are available from Maven Central. Download jannovar-cli-0.36 .jar from here.

Or the jannovar command line tool is available via bioconda. You can install it with the command

```
$ conda install jannovar-cli
```

4.2.2 Install from Source

Note: You only need to install from source if you want to develop Jannovar in Java yourself.

There are two options of installing Jannovar. The recommended way for most users is to download a prebuilt binary and is well-described in the *Quickstart* section. This section describes how to build Jannovar from scratch.

Prequisites

For building Jannovar, you will need

1. Java JDK 8 or higher for compiling Jannovar,
2. Maven 3 for building Jannovar, and
3. Git for getting the sources.

Git Checkout

In this tutorial, we will download the Jannovar sources and build them in `~/Development/jannovar`.

```
~ # mkdir -p ~/Development
~ # cd ~/Development
Development # git clone https://github.com/charite/jannovar.git jannovar
Development # cd jannovar
```

Maven Proxy Settings

If you are behind a proxy, you will get problems with Maven downloading dependencies. If you run into problems, make sure to also delete `~/.m2/repository`. Then, execute the following commands to fill `~/.m2/settings.xml`.

```
jannovar # mkdir -p ~/.m2
jannovar # test -f ~/.m2/settings.xml || cat >~/.m2/settings.xml <<END
<settings>
  <proxies>
   <proxy>
      <active>true</active>
      <protocol>http</protocol>
      <host>proxy.example.com</host>
      <port>8080</port>
      <nonProxyHosts>*.example.com</nonProxyHosts>
    </proxy>
  </proxies>
</settings>
END
```

Building

You can build Jannovar using `mvn package`. This will automatically download all dependencies, build Jannovar, and run all tests.

```
jannovar # mvn package
```

In case that you have non-compiling test, you can use the *-DskipTests=true* parameter for skipping them.

```
jannovar # mvn install -DskipTests=true
```

Creating Eclipse Projects

Maven can be used to generate Eclipse projects that can be imported by the Eclipse IDE. This can be done calling `mvn eclipse:eclipse` command after calling `mvn install`:

```
jannovar # mvn install
jannovar # mvn eclipse:eclipse
```

4.3 Downloading Transcript Databases

The first step after installing Jannovar is to obtain a **transcript database**. This database stores information about the transcripts, such as the location of a transcript and its exons, its CDS start and end position, and the transcript sequence. There are three major sources of annotation databases for the main model organisms: (1) the UCSC genome browser, (2) the Ensembl project, and (3) the RefSeq database at NCBI. Each database is linked to a certain release of a reference genome.

4.3.1 Displaying Available Database

Note: You can use your own datasources by editing the ini file. See **datasource_** for more information.

Jannovar has built-in support for the human and mouse genomes in releases `hg18`, `hg19`, `hg38`, `mm9`, and `mm10`. For each release, the database can originate from the sources `ucsc`, `ensembl`, and `refseq`. Further, the database can be limited to the curated transcripts only when using RefSeq: `refseq_curated`.

The genome release names and the source names are joint into database descriptors such as `hg19/ucsc` and `hg38/refseq`. You can view the built-in database names using the `db-list` Jannovar command:

```
$ java -jar jannovar-cli-0.36.jar db-list
[...]
    hg18/refseq_curated
    hg19/ucsc
[...]
```

4.3.2 Database Download

A database can be downloaded using the `download` command. You can pass a list of database source names to this command. For each, Jannovar will download the database files over the network to the directory `data/${source}` This directory is created if necessary. When a to be downloaded file already exists, Jannovar will not attempt to overwrite this file.

Note: If you have problems with downloading files (e.g., because of proxy settings) and later on building the database fails then you should delete the directory `data/${source}` and retry downloading the file.

Finally, Jannovar will build a file with the extension `.ser` in the directory `data`, e.g. `data/hg19_ucsc.ser`.

Note: If you are behind a proxy then you have to pass the appropriate argument to Jannovar download. For most users, adding `--proxy http://proxy.example.com:8080/` should suffice. Advanced proxy settings and details are explained in the section *Proxy Settings*

Let us now download the RefSeq and UCSC annotations for human release *hg19*:

```
$ java -jar jannovar-cli-0.36.jar download -d hg19/refseq -d hg19/ucsc
```

4.4 Annotating VCF Files

The main purpose of Jannovar is the annotation of all variants in a VCF file. That is, for each annotation, predict the results for all transcripts that can be afflicted by the change. Depending on the configuration, the one effect that is most pathogenic, or all, are written out.

This is done using the `annotate-vcf` command. You pass the path to an annotation database and one VCF file that should be annotated. The resulting annotated file is written to the file specified by `-o` or `--output-vcf`.

For example, for annotating the `small.vcf` file (see small.vcf) in the `examples` directory:

```
# java -jar jannovar-cli-0.36.jar annotate-vcf \
-d data/hg19_refseq.ser -i examples/small.vcf -o examples/small.jv.vcf
[...]
# ls examples/small.jv.vcf
small.jv.vcf
```

The first three variant lines of `examples/small.jv.vcf` will look as follows.

```
1       866511  rs60722469      C       CCCCT   258.62  .       ANN=CCCCT|coding_
↪transcript_intron_variant|LOW|SAMD11|148398|transcript|NM_152486.2|Coding|4/13|c.
↪305+42_305+43insCCCT|p.(%3D)|386/18841|306/2046|102/682||     GT:AD:DP:GQ:PL  1/1:6,
↪5:11:14.79:300,15,0
1       879317  rs7523549       C       T       150.77  .       ANN=T|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.799C>T|p.
↪(Arg267Cys)|1155/19962|799/1188|267/396|| GT:AD:DP:GQ:PL  0/1:14,7:21:99:181,0,367
1       879482  .       G       C       484.52  .       ANN=C|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.964G>C|p.
↪(Asp322His)|1320/19962|964/1188|322/396|| GT:AD:DP:GQ:PL  0/1:28,20:48:99:515,0,794
```

4.4.1 Disabling 3' Shifting

The HGVS Nomenclature for the description fo sequence variants requires that variants are to be shifted towards the 3' end of transcripts in case of ambiguities. This is in partial conflict with the VCF standard which requires all variant calls to be shifted towards the 3' end of the genome. In the case that Jannovar shifted the variants towards the 3' end of the transcript, it will generate a `INFO_REALIGN_3_PRIME` information in the message field of the annotation (`ANN` field).

To comply with the VCF annotation standard, Jannovar also implements the `--no-3-prime-shifting` option. Using this switch suppresses this shifting and the variant will be kept as given in the VCF file. Here is an example of using this command line option:

```
# java -jar jannovar-cli-0.36.jar annotate --no-3-prime-shifting \
-d data/hg19_refseq.ser -i examples/small.vcf -o examples/small.jv.vcf
```

4.4.2 The Show-All Option

By default, Jannovar will only write out one most pathogenic variant as predicted. You can use the `--show-all` option to write out all functional annotations:

```
# java -jar jannovar-cli-0.36.jar annotate-vcf --show-all \
-d data/hg19_refseq.ser -i examples/small.vcf -o examples/small.jv.vcf
```

For example, the first line of `small.jv.vcf` will look as follows and contain multiple effects and HGVS annotations.

```
1       866511  rs60722469      C       CCCCT   258.62  .       ANN=CCCCT|coding_
↪transcript_intron_variant|LOW|SAMD11|148398|transcript|NM_152486.2|Coding|4/13|c.
↪305+42_305+43insCCCT|p.(%3D)|386/18841|306/2046|102/682||,CCCCT|coding_transcript_
↪intron_variant|LOW|SAMD11|148398|transcript|XM_005244723.1|Coding|4/12|c.305+42_
↪305+43insCCCT|p.(%3D)|662/19962|306/2145|102/715||,CCCCT|coding_transcript_intron_
↪variant|LOW|SAMD11|148398|transcript|XM_005244724.1|Coding|4/13|c.305+42_
↪305+43insCCCT|p.(%3D)|662/19962|306/2001|102/667||,CCCCT|coding_transcript_intron_
↪variant|LOW|SAMD11|148398|transcript|XM_005244725.1|Coding|4/13|c.305+42_
↪305+43insCCCT|p.(%3D)|662/19962|306/1998|102/666||,CCCCT|coding_transcript_intron_
↪variant|LOW|SAMD11|148398|transcript|XM_005244726.1|Coding|4/11|c.305+42_
↪305+43insCCCT|p.(%3D)|662/19962|306/1719|102/573||,CCCCT|coding_transcript_intron_
↪variant|LOW|SAMD11|148398|transcript|XM_005244727.1|Coding|4/8|c.305+42_
↪305+43insCCCT|p.(%3D)|662/19962|306/1188|102/396||,CCCCT|non_coding_transcript_
↪intron_variant|LOW|SAMD11|148398|transcript|XR_241028.1|Noncoding|4/12|n.661+42_
↪661+43insCCCT||662/19541||||,CCCCT|non_coding_transcript_intron_
↪variant|LOW|SAMD11|148398|transcript|XR_241029.1|Noncoding|4/12|n.661+42_
↪661+43insCCCT||662/19541||||     GT:AD:DP:GQ:PL  1/1:6,5:11:14.79:300,15,0
```

4.5 Annotating Positions

Sometimes, it is useful to annotate a single position only, for example for quick checks or for debugging purposes. You can do this using the `annotate-pos` command of Jannovar.

You have to pass a path to a annotation database file and one or more chromosomal change specifiers. Jannovar will then return the effect and the HGVS annotation for each chromosomal change.

```
# java -jar jannovar-cli-0.36.jar annotate-pos \
-d data/hg19_refseq.ser -c 'chr1:12345C>A' -c 'chr1:12346C>A'
[...]
#change     effect  hgvs_annotation
chr1:12345C>A   NON_CODING_TRANSCRIPT_INTRON_VARIANT    DDX11L1:NR_046018.2:n.
↪354+118C>A:
chr1:12346C>A   NON_CODING_TRANSCRIPT_INTRON_VARIANT    DDX11L1:NR_046018.2:n.
↪354+119C>A:
```

The format for the chromsomal change is as follows:

```
{CHROMOSOME}:{POSITION}{REF}>{ALT}
```

CHROMOSOME name of the chromosome or contig

POSITION position of the first change base on the chromosome; in the case of insertions the first base after the insertion; the first base on the chromosome has position `1`

REF the reference bases

ALT the alternative bases

4.6 Annotating CSV Files

Sometimes, it is useful to annotate a whole CSV file where you have the chromosome, position, reference allele, and alternative allele in different columns. You can do this using the `annotate-csv` command of Jannovar.

You have to pass a path to a annotation database file and one or more chromosomal change specifiers. Jannovar will then return the effect, the HGVS annotation at the end of the line in the given CSV format and prints it out to the standard output.

Just imagine we have tab separated file with a header named input.tsv

```
contig  position        reference       alt
chr1    12345   C       A
chr1    12346   C       A
```

Now we run jannovar with this command an will get this output:

```
# java -jar jannovar-cli-0.36.jar annotate-csv -d data/hg19_refseq.ser --
↪input input.tsv -c 1 -p 2 -r 3 -a 4 --header --type TDF
[...]
contig  position    reference       alt     HGVS    FunctionalClass
chr1        12345   C       A       DDX11L1:NR_046018.2:n.354+118C>A:
↪NON_CODING_TRANSCRIPT_INTRON_VARIANT
chr1        12346   C       A       DDX11L1:NR_046018.2:n.354+119C>A:
↪NON_CODING_TRANSCRIPT_INTRON_VARIANT
```

The format for the chromsomal change is as follows:

```
{CHROMOSOME}        {POSITION}      {REF}   {ALT}
```

CHROMOSOME name of the chromosome or contig

POSITION position of the first change base on the chromosome; in the case of insertions the first base after the insertion; the first base on the chromosome has position `1`

REF the reference bases

ALT the alternative bases

Right now it is only possible to use the column number and not the header column. This might be extended in the future. Possible CSV file types are:

Default Standard comma separated format, as for RFC4180 but allowing empty lines.

TDF Tab-delimited format.

RFC4180 Comma separated format as defined by RFC4180.

Excel Excel file format (using a comma as the value delimiter). Note that the actual value delimiter used by Excel is locale dependent, it might be necessary to customize this format to accommodate to your regional settings.

MySQL Default MySQL format. This is a tab-delimited format with a LF character as the line separator. Values are not quoted and special characters are escaped with `\`. The default *NULL* string is `\\N`.

4.7 Convert HGVS to VCF

Because Jannovar uses HGVS to describe the variant change in a transcript it is also possible to use HGVS strings and get the genomic position. In Jannovar there is a command line interface to decode a list of HGVS notations into a VCF file.

This is done using the `hgvs-to-vcf` command. You pass the path to an annotation database that you use in you HGVS code and a file where each line is exactly one HGVS notation. In addition the indexed reference fasta file with a dictionary is needed. The resulting file is a fully supported VCF file.

For example, for converting the `small_hgvs.lst` file (see small_hgvs.lst) in the `examples` directory:

```
# java -jar jannovar-cli-0.36.jar hgvs-to-vcf -d data/hg19_refseq.ser -i␣
↪examples/small_hgvs.lst -o examples/small_hgvs.vcf -r hg19.fa
[...]
# ls examples/small_hgvs.vcf
small_hgvs.vcf
```

The file `examples/small_hgvs.vcf` will look as follows.

```
##fileformat=VCFv4.2
##ALT=<ID=ERROR,Description="Error in conversion">
##FILTER=<ID=PARSE_ERROR,Description="Problem in parsing original HGVS variant string,
↪ written out as variant at 1:g.1N>N">
##INFO=<ID=ERROR_MESSAGE,Number=1,Type=String,Description="Error message">
##INFO=<ID=ORIG_VAR,Number=1,Type=String,Description="Original HGVS variant string␣
↪from input file to hgvs-to-vcf">
##contig=<ID=1,length=249250621>
##contig=<ID=2,length=243199373>
##contig=<ID=3,length=198022430>
##contig=<ID=4,length=191154276>
##contig=<ID=5,length=180915260>
##contig=<ID=6,length=171115067>
##contig=<ID=7,length=159138663>
##contig=<ID=8,length=146364022>
##contig=<ID=9,length=141213431>
##contig=<ID=10,length=135534747>
##contig=<ID=11,length=135006516>
##contig=<ID=12,length=133851895>
##contig=<ID=13,length=115169878>
##contig=<ID=14,length=107349540>
##contig=<ID=15,length=102531392>
##contig=<ID=16,length=90354753>
##contig=<ID=17,length=81195210>
##contig=<ID=18,length=78077248>
##contig=<ID=19,length=59128983>
##contig=<ID=20,length=63025520>
##contig=<ID=21,length=48129895>
##contig=<ID=22,length=51304566>
##contig=<ID=X,length=155270560>
##contig=<ID=Y,length=59373566>
##contig=<ID=MT,length=16569>
##contig=<ID=GL000207.1,length=4262>
##contig=<ID=GL000226.1,length=15008>
##contig=<ID=GL000229.1,length=19913>
##contig=<ID=GL000231.1,length=27386>
##contig=<ID=GL000210.1,length=27682>
##contig=<ID=GL000239.1,length=33824>
##contig=<ID=GL000235.1,length=34474>
##contig=<ID=GL000201.1,length=36148>
##contig=<ID=GL000247.1,length=36422>
##contig=<ID=GL000245.1,length=36651>
##contig=<ID=GL000197.1,length=37175>
##contig=<ID=GL000203.1,length=37498>
##contig=<ID=GL000246.1,length=38154>
```

(continues on next page)

(continued from previous page)

```
##contig=<ID=GL000249.1,length=38502>
##contig=<ID=GL000196.1,length=38914>
##contig=<ID=GL000248.1,length=39786>
##contig=<ID=GL000244.1,length=39929>
##contig=<ID=GL000238.1,length=39939>
##contig=<ID=GL000202.1,length=40103>
##contig=<ID=GL000234.1,length=40531>
##contig=<ID=GL000232.1,length=40652>
##contig=<ID=GL000206.1,length=41001>
##contig=<ID=GL000240.1,length=41933>
##contig=<ID=GL000236.1,length=41934>
##contig=<ID=GL000241.1,length=42152>
##contig=<ID=GL000243.1,length=43341>
##contig=<ID=GL000242.1,length=43523>
##contig=<ID=GL000230.1,length=43691>
##contig=<ID=GL000237.1,length=45867>
##contig=<ID=GL000233.1,length=45941>
##contig=<ID=GL000204.1,length=81310>
##contig=<ID=GL000198.1,length=90085>
##contig=<ID=GL000208.1,length=92689>
##contig=<ID=GL000191.1,length=106433>
##contig=<ID=GL000227.1,length=128374>
##contig=<ID=GL000228.1,length=129120>
##contig=<ID=GL000214.1,length=137718>
##contig=<ID=GL000221.1,length=155397>
##contig=<ID=GL000209.1,length=159169>
##contig=<ID=GL000218.1,length=161147>
##contig=<ID=GL000220.1,length=161802>
##contig=<ID=GL000213.1,length=164239>
##contig=<ID=GL000211.1,length=166566>
##contig=<ID=GL000199.1,length=169874>
##contig=<ID=GL000217.1,length=172149>
##contig=<ID=GL000216.1,length=172294>
##contig=<ID=GL000215.1,length=172545>
##contig=<ID=GL000205.1,length=174588>
##contig=<ID=GL000219.1,length=179198>
##contig=<ID=GL000224.1,length=179693>
##contig=<ID=GL000223.1,length=180455>
##contig=<ID=GL000195.1,length=182896>
##contig=<ID=GL000212.1,length=186858>
##contig=<ID=GL000222.1,length=186861>
##contig=<ID=GL000200.1,length=187035>
##contig=<ID=GL000193.1,length=189789>
##contig=<ID=GL000194.1,length=191469>
##contig=<ID=GL000225.1,length=211173>
##contig=<ID=GL000192.1,length=547496>
##contig=<ID=NC_007605,length=171823>
##contig=<ID=hs37d5,length=35477943>
#CHROM      POS     ID      REF     ALT     QUAL    FILTER  INFO
1   197112812       .       GCTC    G       .       .       .
1   866511  .       C       CCCCT   .       .       .
1   879317  .       C       T       .       .       .
1   879482  .       G       C       .       .       .
```

4.8 Mode Of Inheritance Filters

Jannovar includes functionality to filter variants for being compatible with a given pedigree and a mode of inheritance. These filters work well for single individuals and the common case of two parents and a number of children. However, there are limitations when using them for larger pedigrees. For such larger families, the filters lose *specificity* but not *sensitivity*. That is, they can fail to filter out less than theoretically possible, but they should not lose any data.

This section describes in detail the checks performed on the variant and pedigrees to give the user a clear understanding on the algorithms and limitations.

The filters are passed a pedigree and a list of VariantContext calls from the HTSJDK lib (which include Genotypes). The mode of inheritance is selected by the filter choice. Multiple modes are possible. The whole list of VariantContext calls (usually the variants falling onto one gene or transcripts) is then checked for compatibility with the given mode of inheritance(s) and pedigree.

The program rewrites the VariantContext Genotypes to own genotypes. These own genotypes can either be `NOCALL` (no genotype was determined for the person), `REF` (homozygous wild-type), `HET` (heterozygous alternative), or `HOM` (homozygous alternative). In general a caller calls a hemizygous mutations as homozygous. Therefore we use `HOM` and for sensitivity `HET` on known males as hemizygous. The persons can either be affected, unaffected, or their affection state is unknown.

4.8.1 Autosomal Dominant Filter

This filter can be used to filter for *de novo* mutations as well.

- If the pedigree only contains one person then the variant call list must contain one `HET` call.
- If there is more than one person in the pedigree then there must be at least one compatible call, meaning:
  - at least one affected person has a `HET` call for this variant,
  - no affected person has a `REF` or `HOM` call, and
  - no unaffected person has a `HET` or `HOM` call.

4.8.2 Autosomal Recessive Filter

The filter first checks for compatibility with autosomal recessive (AR) homozygous and then AR compound heterozygous mode of inheritance.

For AR homozygous, the following checks are performed.

- If the pedigree only contains one person then the variant call list must contain one `HOM` call.
- If there is more than one person in the pedigree then there must be at least one compatible variant call in the list. For this, the following must be true for one variant in the list:
  - at least one affected person has a `HOM` call for this variant and
  - no affected person has a `REF` or `HET` call.
  - The unaffected parents of affected persons must not be `REF` or `HOM`.
  - There is no unaffected person that has a `HOM` call.

For AR compound heterozygous, the following checks are performed.

- If the pedigree only contains one person then there must be at least two `HET` entries in the variant list.
- If there is more than one person in the pedigree then the algorithm first enumerates *candidate pairs* of variants. The pairs are enumerated for all affected persons that have a father, a mother, or both in the pedigree.

  - The first entry in the pair is compatible with inheritance from the maternal side and the second entry in the pair is compatible from the paternal side.
  - A variant is compatible regarding the paternal side if:
    * the person has calls `HET` or `NOCALL`,
    * the person has no father or the father has calls `HET` or `NOCALL`,
    * the person has no mother or the mother has calls `REF` or `NOCALL`.
  - A variant is compatible regarding the maternal side if:
    * the person has calls `HET` or `NOCALL`,
    * the person has no mother or the mother has calls `HET` or `NOCALL`,
    * the person has no father or the father has calls `REF` or `NOCALL`.
  - Further, no candidate pair may contain the same call for both the maternal and the paternal side, and
  - there must be at least one call for the person, mother, or father that is not `NOCALL`.
- Each candidate pair is then check for compatibility with affected persons. The following is performed as described below and also with a role swap of the paternal and maternal variant call list.
  - For each affected person, the maternal and paternal variant call list is performed for compatibility. For this, each of the following must be checked:
    * If the maternal list is not empty then the genotype of the person in the paternal list must not be `REF` or `HOM`.
    * If the paternal list is not empty then the genotype of the person in the maternal list must not be `REF` or `HOM`.
    * If the paternal list is not empty and the person has a father then the father's genotype in the paternal list must not be `REF` or `HOM`.
    * If the maternal list is not empty and the person has a mother then the mother's genotype in the maternal list must not be `REF` or `HOM`.
    * None of the affected person's unaffected siblings must be both `HET` in the paternal or maternal list.
    * Every affected siblings of an afffected person must have `HET` in the paternal or maternal list.
- Finally, we check every unaffected person in the pedigree.
  - For each unaffected person in the pedigree, neither the maternal nor the paternal call list from the candidate can contain a `HOM` call for the unaffected person.
  - If the call for the unaffected persons is `HET` in both the paternal and the maternal call list. Then, the father's and mother's genotype are checked in the maternal call list of the candidate their genotypes in the paternal call list are considered.
    * Let the first two genotypes be `pp` and `mp` and the second two genotypes be `pm` and `mm`.
    * In the case of `pp == HET and mp == REF and pm == REF and mm == HET` and the case of `pp == REF and mp == HET and pm == HET and mm == REF`, the candidate pairs incompatible and compatible otherwise.

4.8.3 X-Dominant Filter

- First of all variants must be X-Chromosomal.

- If the pedigree only contains one person then we decide if * the person is female then the variant call list must contain one HET call. * else the variant call list must contain a HET or a HOM call.
- If there is more than one person in the pedigree then there must be at least one compatible call, meaning: * at least one affected male has a HET or HOM call or a affected female a HET call for this variant, * no affected person has a REF call, * no a affected female has a HOM call, and * no unaffected person has a HET or HOM call.

4.8.4 X-Recessive Filter

The filter first checks for compatibility with X-chromosomal recessive (XR) homozygous and then XR compound heterozygous mode of inheritance. XR is different to the AR filter, because affected males are always hemizygous (homozygous for the callers). So males do not have compund heterozygous variants.

For XR homozygous, the following checks are performed.

- First of all variants must be X-Chromosomal.
- **If the pedigree only contains one person then we decide if**
  - the person is female then the variant call list must contain one HOM call,
  - else the variant call list must contain a HET or a HOM call.
- If there is more than one person in the pedigree then there must be at least one compatible variant call in the list. For this, the following must be true for one variant in the list:
  - at least one affected male has a HET or HOM call or a affected female a HOM call for this variant,
  - no affected person has a REF or no affected female person has a HET call.
  - **For the parents of affected femals**
    - the father must be affected and
    - the mother cannot have it REF or HOM
  - For the parents of affected males * the unaffected father cannot have the variant HET or HOM * the mother cannot be HOM
  - There is no unaffected person that has a HOM call.
  - There is no unaffected male person that has a HET call.

For XR compound heterozygous, the following checks are performed.

- First of all variants must be X-Chromosomal.
- **If the pedigree only contains one person then we decide if**
  - the person male we do not allow any call. Please use the XR filter.
  - else we use the AR compound heterozygous filter.
- If there is more than one person in the pedigree then the algorithm first enumerates *candidate pairs* of variants. The pairs are enumerated for all affected persons that have a father, a mother, or both in the pedigree.
  - The first entry in the pair is compatible with inheritance from the maternal side and the second entry in the pair is compatible from the paternal side.
  - A variant is compatible regarding the paternal side if:
    - the person has calls HET, NOCALL, or if not female HOM,
    - the person has no father or the father has calls HET, HOM, or NOCALL,
    - the person has no mother or the mother has calls REF or NOCALL.

  - A variant is compatible regarding the maternal side if:
    * the person has calls `HET`, `NOCALL`, or if not female `HOM`,
    * the person has no mother or the mother has calls `HET` or `NOCALL`, and
    * no restrcition to the father because he must be affected. See ckecks later.
  - Further, no candidate pair may contain the same call for both the maternal and the paternal side, and
  - there must be at least one call for the person, mother, or father that is not `NOCALL`.
- Each candidate pair is then check for compatibility with affected persons. The following is performed as described below and also with a role swap of the paternal and maternal variant call list.
  - For each affected person, the maternal and paternal variant call list is performed for compatibility. For this, each of the following must be checked:
    * If the maternal list is not empty then the genotype of a female person in the paternal list must not be `REF` or `HOM`.
    * If the paternal list is not empty then the genotype of the person in the paternal list must not be `REF` or in case of a female `HOM`.
    * If the paternal list is not empty and the person has a father then the father's genotype in the paternal list must not be `REF`.
    * If the maternal list is not empty and the person has a mother then the mother's genotype in the maternal list must not be `REF` or `HOM`.
    * None of the affected person's unaffected siblings must be both `HET` in the paternal or maternal list.
    * Every affected siblings of an afffected person must have `HET` in the paternal or maternal list.
- Finally, we check every unaffected person in the pedigree.
  - For each unaffected person in the pedigree, neither the maternal nor the paternal call list from the candidate can contain a `HOM` or for males also a `HET` call for the unaffected person.
  - If the call for the unaffected persons is `HET` in both the paternal and the maternal call list. Then, the father's and mother's genotype are checked in the maternal call list of the candidate their genotypes in the paternal call list are considered.

Mitochondrial Filter

Jannovar's filter therefore uses the following rules.

- Males and females can be affected by mitochrondial mutations, and so if there is any call from a variant on the mitochondrion, a singleton sample is compatible with mitochondrial inheritance.
- For analyses of mulitple family members, Jannovar calls a variant as compatible with mitochondrial inheritance if all affecteds have the mutation and if the mutation was transmitted by the mother. If an affected person is homozygous reference, then mitochondrial inheritance is ruled out. At least one affected must be called heterozygous or homozygous ALT. Jannovar has no requirement for a certain genotype because current variant callers do not reliably report levels of heteroplasmy.

4.9 Jannovar as a Library

You can obtain the Jannovar library JAR files from Maven Central. Pre-built binaries are available from Maven Central. This query yields the Jannovar libraries.

Note: Note that you should only use JAR files from version 0.36 and not older versions. Some modules have been removed (and merged with other modules) in previous versions.

4.9.1 API Documentation

You can find the Jannovar Javadoc API documentation here:

- https://javadoc.io/doc/de.charite.compbio/jannovar-core/0.36/

4.9.2 Jannovar in your pom.xml

If you plan to process HTSJDK *VariantContext* objects then you will probably only need to depend on `jannovar-htsjdk`.

```
<dependency>
        <groupId>de.charite.compbio</groupId>
        <artifactId>jannovar-htsjdk</artifactId>
        <version>0.36</version>
</dependency>
```

Otherwise, there are the following JAR files:

jannovar-cli Command line interface for Jannovar, not a library.

jannovar-core Core Jannovar functionality with molecular impact annotation and inheritance filtering.

jannovar-hgvs Support for parsing HGVS Variant Nomenclature and representing HGVS variants as Java objects.

jannovar-htsjdk Bridge between core Jannovar functionality and HTSJDK

jannovar-inheritance-checker Older version of inheritance filtering.

jannovar-vardbs Support for annotating variants with VCF databases from various sources, e.g. dbSNP

4.10 Annotation Of Other Databases

In addition to the transcript variant effect annotation Jannovar can annotate some additional databases like ExAC or dbSNP. For these extra annotation the `annotate-vcf` command is used with some additional options.

4.10.1 Exact vs position wise annotation

Different programs that annotate allele frequency etc. use different strategies to annotate variants. Mostly only position wise is annotated and the exact genotype is discarded. If annotation is used for filtering this might be problematic because a rare genotype is filtered out because of a common genotype at the same position. Therefor Jannovar has enables both: position and genotype wise annotation.

If the annotation matches the position of the variant an additional string `OVL_` is added to the token in the INFO column. If the genotype matches this identifier is missing.

4.10.2 ExAC

The command `--exac-vcf <EXAC_VCF>` will add allele frequencies from the ExAC database to the variants. Please download the current veriosn of the ExAC database as VCF (tested with 0.3.1) from their website. In addition you will need the actual reference file because we do a local normalization of the variants; command `--ref-fasta <REF_FASTA>` (a FASTA index file should be present in the same directory; if needed, one can be created with samtools using hte `faidx` command). With `--exac-prefix <EXAC_PREFIX>` you can define the info token used for the annotation. The default is `EXAC_`.

Note: TODO: Add descriptions how about finding the max frequency and finishing the example

Example:

```
$ java -jar jannovar-cli-0.36.jar annotate-vcf \
-d data/hg19_refseq.ser -i examples/small.vcf -o examples/small.jv.vcf \
--exac-vcf ExAC.r0.3.1.sites.vep.vcf.gz --ref-fasta hg19.fa
```

For example, the second line (first coding variant) of `small.jv.vcf` will look as follows and contain the ExAC annotations.

```
1       879317  rs7523549       C       T       150.77  .       ANN=T|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.799C>T|p.
↪(Arg267Cys)|1155/19962|799/1188|267/396||;EXAC_AC_AFR=1922;EXAC_AC_ALL=5591;EXAC_AC_
↪AMR=684;EXAC_AC_EAS=556;EXAC_AC_FIN=215;EXAC_AC_NFE=1933;EXAC_AC_OTH=39;EXAC_AC_
↪POPMAX=1922;EXAC_AC_SAS=242;EXAC_AF_AFR=0.209;EXAC_AF_ALL=0.048;EXAC_AF_AMR=0.060;
↪EXAC_AF_EAS=0.065;EXAC_AF_FIN=0.033;EXAC_AF_NFE=0.031;EXAC_AF_OTH=0.045;EXAC_AF_
↪POPMAX=0.209;EXAC_AF_SAS=0.015;EXAC_AN_AFR=9190;EXAC_AN_ALL=116406;EXAC_AN_
↪AMR=11472;EXAC_AN_EAS=8498;EXAC_AN_FIN=6536;EXAC_AN_NFE=63374;EXAC_AN_OTH=870;EXAC_
↪AN_SAS=16466;EXAC_OVL_AC_AFR=1922;EXAC_OVL_AC_ALL=5591;EXAC_OVL_AC_AMR=684;EXAC_OVL_
↪AC_EAS=556;EXAC_OVL_AC_FIN=215;EXAC_OVL_AC_NFE=1933;EXAC_OVL_AC_OTH=39;EXAC_OVL_AC_
↪POPMAX=1922;EXAC_OVL_AC_SAS=242;EXAC_OVL_AF_AFR=0.209;EXAC_OVL_AF_ALL=0.048;EXAC_
↪OVL_AF_AMR=0.060;EXAC_OVL_AF_EAS=0.065;EXAC_OVL_AF_FIN=0.033;EXAC_OVL_AF_NFE=0.031;
↪EXAC_OVL_AF_OTH=0.045;EXAC_OVL_AF_POPMAX=0.209;EXAC_OVL_AF_SAS=0.015;EXAC_OVL_AN_
↪AFR=9190;EXAC_OVL_AN_ALL=116406;EXAC_OVL_AN_AMR=11472;EXAC_OVL_AN_EAS=8498;EXAC_OVL_
↪AN_FIN=6536;EXAC_OVL_AN_NFE=63374;EXAC_OVL_AN_OTH=870;EXAC_OVL_AN_SAS=16466
↪GT:AD:DP:GQ:PL  0/1:14,7:21:99:181,0,367
```

4.10.3 dnSBP

The command `--dbsnp-vcf <DBSNP_VCF>` will add rsIDs from dbSNP to the variants in the info column and, if the genotype is identical, the ID column will be annotated with te rsID. Please download the latest version of the dbSNP database in VCF format for your genome release from the NCBI website. In addition you will need the actual reference file because we do a local normalization of the variants; command `--ref-fasta <REF_FASTA>`. With `--dbsnp-prefix <DBSNP_PREFIX>` you can define the info token used for the annotation of the rsID. The default is `DBSNP_`.

Example:

```
$ java -jar jannovar-cli-0.36.jar annotate-vcf \
-d data/hg19_refseq.ser -i examples/small.vcf -o examples/small.jv.vcf \
--dbsnp-vcf 00-All.vcf.gz --ref-fasta hg19.fa
```

For example, the first line of `small.jv.vcf` will look as follows and contain the dnSBNP annotation *rs375757231* as exact match.

```
1       866511  rs60722469;rs375757231  C       CCCCT   258.62  .       ␣
↪ANN=CCCCT|coding_transcript_intron_variant|LOW|SAMD11|148398|transcript|NM_152486.
↪2|Coding|4/13|c.305+42_305+43insCCCT|p.(%3D)|386/18841|306/2046|102/682||;DBSNP_
↪CAF=0.149;DBSNP_COMMON=1;DBSNP_G5=1;DBSNP_IDS=rs375757231;DBSNP_OVL_CAF=0.149;DBSNP_
↪OVL_COMMON=1;DBSNP_OVL_G5=1;DBSNP_OVL_IDS=rs375757231     GT:AD:DP:GQ:PL  1/1:6,
↪5:11:14.79:300,15,0
```

4.10.4 UK10K

The command `--uk10k-vcf <UK10K_VCF>` will add allele frequencies from the UK10K project to the variants. Please download the actual UK10K samples from their website (you have to register). In addition you will need the actual reference file because we do a local normalization of the variants; command `--ref-fasta <REF_FASTA>`. With `--uk10k-prefix <UK10K_PREFIX>` you can define the info token used for the annotation. The default is `UK10K_`.

Note: TODO: Add descriptions how about finding the max frequency and finishing the example

Example:

```
$ java -jar jannovar-cli-0.36.jar annotate-vcf \
-d data/hg19_refseq.ser -i examples/small.vcf -o examples/small.jv.vcf \
--uk10k-vcf UK10K_COHORT.20160215.sites.vcf.gz --ref-fasta hg19.fa
```

For example, the first line of `small.jv.vcf` will look as follows and contain the UK10K annotations.

```
1       866511  rs60722469      C       CCCCT   258.62  .       ANN=CCCCT|coding_
↪transcript_intron_variant|LOW|SAMD11|148398|transcript|NM_152486.2|Coding|4/13|c.
↪305+42_305+43insCCCT|p.(%3D)|386/18841|306/2046|102/682||;UK10K_AC=5708;UK10K_AF=0.
↪755;UK10K_AN=7562;UK10K_OVL_AC=5708;UK10K_OVL_AF=0.755;UK10K_OVL_AN=7562  ␣
↪GT:AD:DP:GQ:PL  1/1:6,5:11:14.79:300,15,0
```

4.11 Inheritance Annotation

The command `jannovar annotate-vcf` can also be used for annotating variants with compatible modes of inheritance. For this, you have to specify a pedigree file.

In short, these are TSV files with 6 columns. Each line describes one individual.

1. Pedigree/family name (only individuals of the first occuring family name are interpreted).
2. Name of the individual.
3. Name of the father, `0` for "not in pedigree" (for founders)
4. Name of the mother, `0` for "not in pedigree" (for founders)
5. Sex of the individual: `1` for male, `2` for female, `0` for other/unknown
6. Disease status: `1` for unaffected, `2` for affected, 0 for unknown

4.11.1 Used Pedigree Files

The file `ar.ped` contains the following pedigree which matches autosomal recessive inheritance. Note that the index could also have a de novo mutation (which is flagged as autosomal dominant by Jannovar).

```
FAM index   father  mother  1       2
FAM father  0       0       1       1
FAM mother  0       0       2       1
```

The file `ad.ped` contains the following pedigree which matches autosomal dominant inheritance.

```
FAM index   father  mother  1       2
FAM father  0       0       1       2
FAM mother  0       0       2       1
```

4.11.2 Used VCF File

The flie `small.vcf` contains the following variant file.

```
##fileformat=VCFv4.1
##contig=<ID=1,length=249250621>
##INFO=<ID=INHERITANCE,Number=.,Type=String,Description="Mode of Inheritance">
#CHROM      POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  index
↪father  mother
1   866511  .       C       CCCCT   .       .       .       GT      1/1     0/1     0/
↪1
1   879317  .       C       T       .       .       .       GT      0/1     0/1     0/
↪0
1   879318  .       G       T       .       .       .       GT      0/1     0/0     0/
↪0
1   879482  .       G       C       .       .       .       GT      0/1     0/1     0/
↪0
```

Note that all variants lie within the same gene, as shown in the following, annotated version of small.vcf.

```
##fileformat=VCFv4.2
##INFO=<ID=ANN,Number=.,Type=String,Description="Functional annotations:
↪'Allele|Annotation|Annotation_Impact|Gene_Name|Gene_ID|Feature_Type|Feature_
↪ID|Transcript_BioType|Rank|HGVS.c|HGVS.p|cDNA.pos / cDNA.length|CDS.pos / CDS.
↪length|AA.pos / AA.length|Distance|ERRORS / WARNINGS / INFO'">
##INFO=<ID=INHERITANCE,Number=.,Type=String,Description="Mode of Inheritance">
##contig=<ID=1,length=249250621>
##jannovarCommand=annotate-vcf -d data/hg19_refseq.ser -i small.vcf -o small.jv.vcf
##jannovarVersion=0.17
#CHROM      POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  index
↪father  mother
1   866511  .       C       CCCCT   .       .       ANN=CCCCT|coding_transcript_
↪intron_variant|LOW|SAMD11|148398|transcript|NM_152486.2|Coding|4/13|c.305+42_
↪305+43insCCCT|p.(%3D)|386/18841|306/2046|102/682||     GT      1/1     0/10/1
1   879317  .       C       T       .       .       ANN=T|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.799C>T|p.
↪(Arg267Cys)|1155/19962|799/1188|267/396|| GT      0/1     0/1     0/0
1   879318  .       G       T       .       .       ANN=T|missense_
↪variant|MODERATE|SAMD11|148398|transcript|NM_152486.2|Coding|14/14|c.1831G>T|p.
↪(Val611Leu)|1911/18841|1831/2046|611/682||         GT      0/1     0/0     0/0
1   879482  .       G       C       .       .       ANN=C|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.964G>C|p.
↪(Asp322His)|1320/19962|964/1188|322/396|| GT      0/1     0/0     0/1
```

Also note that for annotating for compatibility with inheritance, **all** variants assigned to a gene will be used. This includes deep intronic as well as upstream/downstream (up to 5kbp) variants. Thus, it is a good idea to first filter out

low-quality and non-coding variants before annotating compatible modes of inheritance. It remains for future work to include a quality/variant type filter for the pedigree-based filtration.

4.11.3 Annotating AR Variants

The following shows the annotation result with the AR pedigree. The molecular impact annotation `ANN` is suppressed for brevity.

```
$ java -jar jannovar-cli.jar annotate-vcf \
    -d data/hg19_refseq.ser -i small.vcf \
    -o small.ar.vcf --pedigree-file ar.ped
$ cat small.ar.vcf
##fileformat=VCFv4.2
##INFO=<ID=ANN,Number=.,Type=String,Description="Functional annotations:
↪'Allele|Annotation|Annotation_Impact|Gene_Name|Gene_ID|Feature_Type|Feature_
↪ID|Transcript_BioType|Rank|HGVS.c|HGVS.p|cDNA.pos / cDNA.length|CDS.pos / CDS.
↪length|AA.pos / AA.length|Distance|ERRORS / WARNINGS / INFO'">
##INFO=<ID=INHERITANCE,Number=.,Type=String,Description="Mode of Inheritance">
##contig=<ID=1,length=249250621>
##jannovarCommand=annotate-vcf -d data/hg19_refseq.ser -i small.vcf -o small.ar.vcf --
↪pedigree-file ar.ped
##jannovarVersion=0.17
#CHROM      POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  index  ␣
↪father  mother
1   866511  .       C       CCCCT   .       .       ANN=[...];INHERITANCE=AR        ␣
↪GT      1/1     0/1     0/1
1   879317  .       C       T       .       .       ANN=[...];INHERITANCE=AR        ␣
↪GT      0/1     0/1     0/0
1   879318  .       G       T       .       .       ANN=[...];INHERITANCE=AD        ␣
↪GT      0/1     0/0     0/0
1   879482  .       G       C       .       .       ANN=[...];INHERITANCE=AR        ␣
↪GT      0/1     0/0     0/1
```

The variant at

- `chr1:866511` is annotated as compatible with autosomal recessive as it is a "classic" autosomal recessive variant.
- `chr1:879317` is annotated as compatible with autosomal recessive as together with the variant at `chr1:879482`, it matches the composite autosomal recessive mode of inheritance.
- `chr1:879318` is annotated as compatible with autosomal dominant as it could be a de novo variant that is autosomal dominant.

4.11.4 Annotating AD Variants

```
$ $ java -jar jannovar-cli.jar annotate-vcf \
    -d data/hg19_refseq.ser -i small.vcf \
    -o small.ad.vcf --pedigree-file ad.ped
$ cat small.ad.vcf
##fileformat=VCFv4.2
##INFO=<ID=ANN,Number=.,Type=String,Description="Functional annotations:
↪'Allele|Annotation|Annotation_Impact|Gene_Name|Gene_ID|Feature_Type|Feature_
↪ID|Transcript_BioType|Rank|HGVS.c|HGVS.p|cDNA.pos / cDNA.length|CDS.pos / CDS.
↪length|AA.pos / AA.length|Distance|ERRORS / WARNINGS / INFO'">
##INFO=<ID=INHERITANCE,Number=.,Type=String,Description="Mode of Inheritance">
```

(continues on next page)

(continued from previous page)

```
##contig=<ID=1,length=249250621>
##jannovarCommand=annotate-vcf -d data/hg19_refseq.ser -i small.vcf -o small.ad.vcf --
↪pedigree-file ad.ped
##jannovarVersion=0.17
#CHROM      POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  index␣
↪father  mother
1   866511  .       C       CCCCT   .       .       ANN=[...]       GT      1/1     0/
↪1     0/1
1   879317  .       C       T       .       .       ANN=[...];INHERITANCE=AD        ␣
↪GT      0/1     0/1     0/0
1   879318  .       G       T       .       .       ANN=[...]       GT      0/1     0/
↪0     0/0
1   879482  .       G       C       .       .       ANN=[...];INHERITANCE=AD        ␣
↪GT      0/1     0/1     0/0
```

The variants at `chr1:879317` and `chr1:879482` match the autosomal dominant mode of inheritance from the father. The remaining variants do not match this mode of inheritance.

4.11.5 No-calls and Mixed genotypes

We implemented the filter that we might loose specificity but not some sensitibvity.Therfore a genotype call of `./1` or `1/.` can be `HET` or `HOM_ALT`. `0/.` or `./0` are `HET` or `HOM_REF`. A no-call of `./.` is `NO_CALL` and will be used only as a wildcard in multi-vcfs but at least one called correct genotype must be observed. For more information see ped_filters.

4.12 Filtering Annotations

Jannovar is foremost a program for **annotating** variants. Its features include the annotation based on predicted molecular impact, but also for compatibility with different inheritance models.

These annotations can in turn be used for **filtering** variants, i.e., including or excluding variants based on some criteria. The functionality for filtration is not included in Jannovar itself but can easily be performed with bcftools (or even `grep` if you are brave).

4.12.1 Variant Filtration with BCFtools

Given an annotated VCF file, we can easily use the `bcftools view` command for filtering the variant to

- contain only variants with a given annotation, or
- contain no variant with a given annotation.

We will use the following annotated VCF file `small.vcf` that we will filter.:

```
##INFO=<ID=INHERITANCE,Number=.,Type=String,Description="Mode of Inheritance">
##contig=<ID=1,length=249250621>
##jannovarCommand=annotate-v -d data/hg19_refseq.ser -i small.vcf -o small.jv.vcf
##jannovarVersion=0.17
#CHROM      POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  individual
1   866511  rs60722469      C       CCCCT   258.62  .       ANN=CCCCT|coding_
↪transcript_intron_variant|LOW|SAMD11|148398|transcript|NM_152486.2|Coding|4/13|c.
↪305+42_305+43insCCCT|p.(%3D)|386/18841|306/2046|102/682||;INHERITANCE=AR      ␣
↪GT:AD:DP:GQ:PL  1/1:6,5:11:14.79:300,15,0
```

(continues on next page)

(continued from previous page)

```
1   879317  rs7523549       C       T       150.77  .       ANN=T|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.799C>T|p.
↪(Arg267Cys)|1155/19962|799/1188|267/396|| GT:AD:DP:GQ:PL  0/1:14,7:21:99:181,0,367
1   879482  .       G       C       484.52  .       ANN=C|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.964G>C|p.
↪(Asp322His)|1320/19962|964/1188|322/396|| GT:AD:DP:GQ:PL  0/1:28,20:48:99:515,0,794
```

For example, we can limit the variants to those compatible with autosomal recessive mode of inheritance.

```
$ bcftools view -i 'INHERITANCE[*] = "AD"' small.vcf
##fileformat=VCFv4.1
##FILTER=<ID=PASS,Description="All filters passed">
##contig=<ID=1,length=249250621>
##INFO=<ID=INHERITANCE,Number=.,Type=String,Description="Mode of Inheritance">
##bcftools_viewVersion=1.2+htslib-1.2.1
##bcftools_viewCommand=view -i 'INHERITANCE[*] = "AD"' small.vcf
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  individual
1   866511  rs60722469      C       CCCCT   258.62  .       ANN=CCCCT|coding_
↪transcript_intron_variant|LOW|SAMD11|148398|transcript|NM_152486.2|Coding|4/13|c.
↪305+42_305+43insCCCT|p.(%3D)|386/18841|306/2046|102/682||;INHERITANCE=AR
↪GT:AD:DP:GQ:PL  1/1:6,5:11:14.79:300,15,0
```

Similarly, we can remove all files compatible with this mode of inheritance.

```
$ bcftools view -e 'INHERITANCE[*] = "AD"' small.vcf
##fileformat=VCFv4.1
##FILTER=<ID=PASS,Description="All filters passed">
##contig=<ID=1,length=249250621>
##INFO=<ID=INHERITANCE,Number=.,Type=String,Description="Mode of Inheritance">
##bcftools_viewVersion=1.2+htslib-1.2.1
##bcftools_viewCommand=view -i 'INHERITANCE[*] = "AD"' small.vcf
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  individual
1   879317  rs7523549       C       T       150.77  .       ANN=T|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.799C>T|p.
↪(Arg267Cys)|1155/19962|799/1188|267/396|| GT:AD:DP:GQ:PL  0/1:14,7:21:99:181,0,367
1   879482  .       G       C       484.52  .       ANN=C|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.964G>C|p.
↪(Asp322His)|1320/19962|964/1188|322/396|| GT:AD:DP:GQ:PL  0/1:28,20:48:99:515,0,794
```

The following shows how to limit the variants to those having a missense functional impact.

```
$ bcftools view -i 'ANN ~ "missense"' small.vcf
##fileformat=VCFv4.2
##FILTER=<ID=PASS,Description="All filters passed">
##INFO=<ID=ANN,Number=.,Type=String,Description="Functional annotations:
↪'Allele|Annotation|Annotation_Impact|Gene_Name|Gene_ID|Feature_Type|Feature_
↪ID|Transcript_BioType|Rank|HGVS.c|HGVS.p|cDNA.pos / cDNA.length|CDS.pos / CDS.
↪length|AA.pos / AA.length|Distance|ERRORS / WARNINGS / INFO'">
##INFO=<ID=INHERITANCE,Number=.,Type=String,Description="Mode of Inheritance">
##contig=<ID=1,length=249250621>
##jannovarCommand=annotate-v -d data/hg19_refseq.ser -i small.vcf -o small.jv.vcf
##jannovarVersion=0.17
##bcftools_viewVersion=1.2+htslib-1.2.1
##bcftools_viewCommand=view -i 'ANN ~ "missense"' small.jv.vcf
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  individual
1   879317  rs7523549       C       T       150.77  .       ANN=T|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.799C>T|p.
↪(Arg267Cys)|1155/19962|799/1188|267/396|| GT:AD:DP:GQ:PL  0/1:14,7:21:99:181,0,367
```

(continues on next page)

(continued from previous page)

```
1   879482  .       G       C       484.52  .       ANN=C|missense_
↪variant|MODERATE|SAMD11|148398|transcript|XM_005244727.1|Coding|9/9|c.964G>C|p.
↪(Asp322His)|1320/19962|964/1188|322/396|| GT:AD:DP:GQ:PL  0/1:28,20:48:99:515,0,794
```

Similarly, we could use `-e` instead of `-i` for inverting the selection.

4.13 Java Memory Settings

Jannovar is a Java program that runs using the Java Virtual Machine (JVM). The program does not allocate memory directly but through the JVM which uses a fixed maximal memory limit. If Jannovar terminates by throwing an exception `java.lang.OutOfMemoryError` then you have to increase the memory limit of the JVM.

One way of doing this by setting the environment variable `JAVA_TOOL_OPTIONS`. For example, the following line increases the available memory to 2 GB of RAM.

```
export JAVA_TOOL_OPTIONS="-Xms2G -Xmx2G"
```

If you prefer, then you can also pass these options to the invokation of JVM. The following Jannovar invocation allows to use up to 2 GB of RAM:

```
java -Xms2G -Xmx2G -jar jannovar-cli-0.36.jar [...]
```

4.14 Proxy Settings

If you have to use a proxy for connecting to the internet then you can do so either using command line parameters to Jannovar `dowload` or using environment variables. If you do not have to use a proxy then you can ignore this section.

4.14.1 Proxy Command Line Arguments

You can specify one proxy URL for all protocols or give a different proxy for each protocol. Below is a list of the proxy-related command line arguments with an example value. The value of `--proxy` can be overridden by the protocol-specific options.

`--proxy http://proxy.example.com:8080/` Fallback proxy URL **most users only have to specify this**.

`--http-proxy http://proxy.example.com:8080/` Proxy URL for the HTTP protocol.

`--ftp-proxy http://proxy.example.com:8080/` Proxy URL for the FTP protocol.

`--https-proxy http://proxy.example.com:8080/` Proxy URL for the HTTPS protocol.

For most users, it is sufficient to use `--proxy` only:

```
# java -jar jannovar-cli-0.36.jar download --proxy http://proxy.example.com:
8080/ -d hg19/ucsc
```

4.14.2 Proxy Environment Variables

It might be more convenient to use environment variables that can be configured globally. Jannovar interprets the following environment variables that are commonly used on Unix systems (and also interpreted by tools such as `curl`). For each protocol, Jannovar accepts both the upper and the lower case version and it is sufficient to specify one for each protocol.

http_proxy, HTTP_PROXY Proxy URL for the HTTP protocol.

https_proxy, HTTPS_PROXY Proxy URL for the HTTPS protocol.

ftp_proxy, FTP_PROXY Proxy URL for the FTP protocol.

If you are on Linux and have not already done so, you can add the following lines to your startup script (e.g., `~/.profile`):

```
export http_proxy=http://proxy.example.com:8080/
export https_proxy=http://proxy.example.com:8080/
export ftp_proxy=http://proxy.example.com:8080/
export no_proxy="localhost,127.0.0.1,localaddress,.localdomain.com,*.example.com"
export HTTP_PROXY=http://proxy.example.com:8080/
export HTTPS_PROXY=http://proxy.example.com:8080/
export FTP_PROXY=http://proxy.example.com:8080/
```

If you have write access to `/etc/environment`, you can add the following lines there:

```
http_proxy=http://proxy.example.com:8080/
https_proxy=http://proxy.example.com:8080/
ftp_proxy=http://proxy.example.com:8080/
no_proxy="localhost,127.0.0.1,localaddress,.localdomain.com,*.example.com"
HTTP_PROXY=http://proxy.example.com:8080/
HTTPS_PROXY=http://proxy.example.com:8080/
FTP_PROXY=http://proxy.example.com:8080/
```

4.15 Custom Datasources

Jannovar ships with a number of predefined data sources (e.g., UCSC, Ensembl, and RefSeq for human releases hg18 to hg38, and mouse mm9 and mm10). However, it is quite easy to define your own data source by writing a datasource INI file. This section describes how to define your own data source.

Note: If you think that your new data source would be useful for others, please send them to us either using our issue tracker or by sending an email to Peter N Robinson <peter.robinson@jax.org>.

4.15.1 Datasource INI Files

The data sources are defined in INI files. For example, consider the following definition of human release `hg19` from UCSC:

```
[hg19/ucsc]
type=ucsc
alias=MT,M,chrM
chromInfo=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/chromInfo.txt.gz
chrToAccessions=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_
↪RELEASE.105/Assembled_chromosomes/chr_accessions_GRCh37.p13
chrToAccessions.format=chr_accessions
knownCanonical=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownCanonical.
↪txt.gz
knownGene=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownGene.txt.gz
knownGeneMrna=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownGeneMrna.
↪txt.gz
```

(continues on next page)

(continued from previous page)

```
kgXref=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/kgXref.txt.gz
knownToLocusLink=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/
↪knownToLocusLink.txt.gz
```

The section name `hg19/ucsc` defines the data source name. When saving the above file contents as `my_ucsc.ini`, you can pass it to the Jannovar `download` command `--data-source-list/-s`.

```
java -Xms2G -Xmx2G -jar jannovar-cli-0.36.jar download -s my_ucsc.ini -d hg19/
↪ucsc
```

Your INI file can either add new definitions or override the built-in ones. In fact, the definition from above is part of the INI file that is contained in the Jannovar JAR file and used by default.

The `type` setting of the data source section defines the type of the data source. Currently, Jannovar supports the types `ensembl`, `refseq`, and `ucsc`. The sections below explain the general settings and the data source types further.

4.15.2 Chromosome Aliasing

The `alias` setting defines an aliasing of the contigs and chromosomes. It can be used regardless of the used data source type.

The names of the contigs from the different data sources usually differ between UCSC and RefSeq (and Ensembl which uses the same names as RefSeq). Usually, the UCSC names can be derived from the RefSeq names by prepending `"chr"`. However, this is not true for the important case of the mitochondrial chromosome.

The `alias` line from above defines an alias between the chromosome names *MT*, *M*, and *chrM*. The first entry (*MT*) is implicitely added if it is not in the *chromInfo* file (see *Name Mapping and Lengths*). This is the case for older RefSeq releases.

4.15.3 Name Mapping and Lengths

The `chromInfo` setting defines the URL to the `chromInfo.txt.gz` file from UCSC. Usually, this URL is `http://hgdownload.soe.ucsc.edu/goldenPath/${RELEASE}/database/chromInfo.txt.gz`. This file contains the contig lengths for each chromosome with the UCSC name of the chromosome/contig (e.g., `chr19`).

The `chrToAccessions` setting defines the URL to the RefSeq file that contains the mapping from the RefSeq names to the RefSeq and GenBank contig sequence accessions. It is assumed that the UCSC contig names are derived from the RefSeq contig names by prepending `"chr"`, also see *Chromosome Aliasing*. This information is required as it is equally common to use the RefSeq names, UCSC names, or Genbank or RefSeq contig sequence accessions.

The two settings `chromInfo` and `chrToAccessions` have to be provided for all data source types.

The `chroToAccessions` file can have different formats, specified as `chrToAccessions.format`. The "modern" one is `chr_accessions` where the file is a TSV file with five columns, e.g.:

```
#Chromosome RefSeq Accession.version        RefSeq gi       GenBank Accession.version␣
↪      GenBank gi
1   NC_000001.10    224589800       CM000663.1      224384768
2   NC_000002.11    224589811       CM000664.1      224384767
3   NC_000003.11    224589815       CM000665.1      224384766
[...]
```

The first column gives the RefSeq name, the second the RefSeq sequence accession number, and the fourth one the GenBank accession number.

The `chr_NC_gi` file format has four columns and contains the mapping for the HuRef but also alternative assemblies, e.g.:

```
#Chr        Accession.ver   gi      Assembly
1   AC_000044.1     89161184        Celera
2   AC_000045.1     89161198        Celera
[...]
1   AC_000133.1     157704448       HuRef
2   AC_000134.1     157724517       HuRef
```

In this case, you have to specify a value that the last column should match to. The hg18 release uses the `chr_NC_gi` format, for example. Here, we filter the lines to those having `"HuRef"` in the last column:

```
[hg18/refseq]
type=refseq
alias=MT,M,chrM
chromInfo=http://hgdownload.soe.ucsc.edu/goldenPath/hg18/database/chromInfo.txt.gz
chrToAccessions=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/BUILD.36.3/
↪Assembled_chromosomes/chr_NC_gi
chrToAccessions.format=chr_NC_gi
chrToAccessions.matchLast=HuRef
gff=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/BUILD.36.3/GFF/ref_NCBI36_
↪top_level.gff3.gz
rna=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/BUILD.36.3/RNA/rna.fa.gz
```

4.15.4 Ensembl Data Sources

When selecting the `ensembl` data source type then you have to pass the transcript definition GTF URL to `gtf` and the cDNA FASTA file to `cdna`. Below is an example for the Ensemble data source for human release hg19.

```
[hg19/ensembl]
type=ensembl
alias=MT,M,chrM
chromInfo=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/chromInfo.txt.gz
chrToAccessions=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_
↪RELEASE.105/Assembled_chromosomes/chr_accessions_GRCh37.p13
chrToAccessions.format=chr_accessions
gtf=ftp://ftp.ensembl.org/pub/release-74/gtf/homo_sapiens/Homo_sapiens.GRCh37.74.gtf.
↪gz
cdna=ftp://ftp.ensembl.org/pub/release-74/fasta/homo_sapiens/cdna/Homo_sapiens.GRCh37.
↪74.cdna.all.fa.gz
```

4.15.5 RefSeq Data Sources

When selecting the `refseq` data source type then you have to pass the transcript definition GFF URL to `gff` and the RNA FASTA file to `rna`. Below is an example for the RefSeq data source for human release hg19.

```
[hg19/refseq]
type=refseq
alias=MT,M,chrM
chromInfo=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/chromInfo.txt.gz
chrToAccessions=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_
↪RELEASE.105/Assembled_chromosomes/chr_accessions_GRCh37.p13
chrToAccessions.format=chr_accessions
```

(continues on next page)

(continued from previous page)

```
gff=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_RELEASE.105/GFF/
↪ref_GRCh37.p13_top_level.gff3.gz
rna=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_RELEASE.105/RNA/
↪rna.fa.gz
```

For RefSeq, you can also limit building the database to those transcripts that are curated (e.g., that do not have a name starting with `"XM_"` or `"XR_"`. You can do this by setting `onlyCurated` to `true`:

```
[hg19/refseq_curated]
type=refseq
alias=MT,M,chrM
onlyCurated=true
chromInfo=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/chromInfo.txt.gz
chrToAccessions=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_
↪RELEASE.105/Assembled_chromosomes/chr_accessions_GRCh37.p13
chrToAccessions.format=chr_accessions
gff=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_RELEASE.105/GFF/
↪ref_GRCh37.p13_top_level.gff3.gz
rna=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_RELEASE.105/RNA/
↪rna.fa.gz
```

Additionally, `hg19/refseq_interim` defines the URLS for the *GRCh37.p13 interim release of the RefSeq data <https://www.ncbi.nlm.nih.gov/books/NBK430989/#_news_02-14-2017-interim-annotation-update-human_>*:

```
[hg19/refseq_interim]
type=refseq
alias=MT,M,chrM
chromInfo=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/chromInfo.txt.gz
chrToAccessions=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_
↪RELEASE.105/Assembled_chromosomes/chr_accessions_GRCh37.p13
chrToAccessions.format=chr_accessions
gff=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/GRCh37.p13_interim_annotation/
↪interim_GRCh37.p13_top_level_2017-01-13.gff3.gz
rna=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/GRCh37.p13_interim_annotation/
↪interim_GRCh37.p13_rna.fa.gz
```

RefSeq transcripts for genes in the pseudo-autosomal regions on chromosome X and Y may have more than one location. Per default, the last entry in the downloaded GFF file (typically, transcripts on chromosome Y) will be preferred over those on chromosome X. To change this behavior (e.g. because the underlying data were aligned against a reference genome with a hard-masked PAR on chromosome Y), use `preferPARTranscriptsOnChrX=true` (default is `false`).

4.15.6 UCSC Data Sources

For UCSC data sources, you have specify the settings `knownCanonical`, `knownGene`, `knownGeneMrna`, `kgXref`, and `knownToLocusLink`. These can usually be derived from the example below by exchanging `hg19` by the release id (e.g., `mm10` for mouse release 10).

```
[hg19/ucsc]
type=ucsc
alias=MT,M,chrM
chromInfo=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/chromInfo.txt.gz
chrToAccessions=ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/ARCHIVE/ANNOTATION_
↪RELEASE.105/Assembled_chromosomes/chr_accessions_GRCh37.p13
chrToAccessions.format=chr_accessions
```

(continues on next page)

(continued from previous page)

```
knownCanonical=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownCanonical.
↪txt.gz
knownGene=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownGene.txt.gz
knownGeneMrna=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownGeneMrna.
↪txt.gz
kgXref=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/kgXref.txt.gz
knownToLocusLink=http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/
↪knownToLocusLink.txt.gz
```

4.16 Frequently Asked Questions

4.16.1 Where are the questions?

Right now, there are no frequently asked questions.

4.17 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.17.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/charite/jannovar/issues

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the Github issues for bugs. If you want to start working on a bug then please write short message on the issue tracker to prevent duplicate work.

Implement Features

Look through the Github issues for features. If you want to start working on an issue then please write short message on the issue tracker to prevent duplicate work.

Write Documentation

Jannovar could always use more documentation, whether as part of the official vcfpy docs, in docstrings, or even on the web in blog posts, articles, and such.

Jannovar uses Sphinx for the user manual (that you are currently reading). See *doc_guidelines* on how the documentation reStructuredText is used. See *doc_setup* on creating a local setup for building the documentation.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/charite/jannovar/issues

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.17.2 Documentation Guidelines

For the documentation, please adhere to the following guidelines:

- Put each sentence on its own line, this makes tracking changes through Git SCM easier.
- Provide hyperlink targets, at least for the first two section levels.
- Use the section structure from below.

```
.. heading_1:

=========
Heading 1
=========


.. heading_2:

---------
Heading 2
---------


.. heading_3:

Heading 3
=========


.. heading_4:

Heading 4
---------


.. heading_5:
```

(continues on next page)

(continued from previous page)

```
Heading 5
~~~~~~~~~


.. heading_6:

Heading 6
:::::::::
```

4.17.3 Documentation Setup

For building the documentation, you have to install the Python program Sphinx. This is best done in a virtual environment. The following assumes you have a working Python 3 setup.

Use the following steps for installing Sphinx and the dependencies for building the Jannovar documentation:

```
$ cd jannovar/manual
$ virtualenv -p python3 .venv
$ source .venv/bin/activate
$ pip install --upgrade -r requirements.txt
```

Use the following for building the documentation. The first two lines is only required for loading the virtualenv. Afterwards, you can always use `make html` for building.

```
$ cd jannovar/manual
$ source .venv/bin/activate
$ make html  # rebuild for changed files only
$ make clean && make html  # force rebuild
```

4.17.4 Get Started!

Ready to contribute? First, create your Java/Documentation development setup as described in *install_from_source/doc_setup*.

1. Fork the *Jannovar* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/jannovar.git
   ```

3. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making your changes, make sure that the build runs through. For Java:

   $ mvn package

   For documentation:

   $ make clean && make html

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.17.5 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated.
3. Describe your changes in the `CHANGELOG.md` file.

4.18 Authors

in alphabetical order

- Manuel Holtgrewe
- Marten Jaeger
- Max Schubach
- Peter N. Robinson

4.18.1 Contributors

- Roland Ewald

4.19 History

```
# Jannovar Changelog

## v0.36

**Highlight: Pre-Built Databases**

We now provide pre-built databases via Zenodo.
This addresses various issues that were caused by changing upstream data URLs.
See `README.md` for details.

### overall

* Switching to Github Workflows for continuous integration.
* Bumping a couple of dependencies.
* Switching to JUNIT 5.
* Pre-built databases are now available via Zenodo (see `README.md` for instructions).
  We provide shell scripts instead of using Java programs for downloading the files.

### jannovar-core
```

(continues on next page)

(continued from previous page)

```
* Apply fix for (#498, PR #499 by @roland-ewald of @limbus-medtec).
  This fixes a problem with right-shifting deletions on amino acid sequences.
  See the tickes and merge request for details.
* Fixing problem with GRCh38 RefSeq annotation / projection (#524).

### jannovar-cli

* Adding command `vardb-import` for importing annotations from VCF files into H2␣
↪database files.
* Adding command `vardb-list` for listing annotation meta data from `vardb-import`.
* Adding command `vardb-annotate` for annotating VCF files from Jannovar H2 database␣
↪files.

### jannovar-vardbs

* Deprecating previous content of the package.
  The deprecated classes are due to be removed in v0.36.
* Adding modules for importing VCF files into H2 database files, listing the meta␣
↪data conents, and annotating VCF files.

### jannovar-hgvs

* Fixing parsing of unchanged (`=`) for nucleic acide sequences (#493).
* Fixing issue with HGVS `delins` with `del` bases but no `ins` bases (#491).

## v0.35

### jannovar-core

- Update Genotype internals to pre-calculate values.

### jannovar-cli

- Fixing `default_sources.ini` file.

## v0.34

### jannovar-cli

* disabeling ensemble for mouse (does not work with an hgnc file)
* updating broken links in download source file
* Adding faMT genomes for all refseq annotations
* bumping dependency on lombok

### jannovar-core

* Remove hard-coded chrMT renaming. Filenames for download that have a ? in their URL␣
↪will be splitted and only the first part before the ? is used a file name in the␣
↪download path.
* Making faMT annotation for refSeq optional
* Update RefSeq parser that does not run into null-pointer exceptions on mouse and␣
↪rat genome (e.g. when no exon defines the gene name)
* Fixing issue with block substitutions (#475).
* Fixing RefSeq build to properly assign transcript model (use best match with␣
↪alignment) in the case of duplicates.
* Fixing issue in projection in the case of leading gaps (has no effect on CDS␣
↪position prediction).
```

(continues on next page)

(continued from previous page)

```
* Adding `TranscriptModel.getTrimmedSequence()` that removes leading and trailing␣
↪(unaligned in RefSeq) sequence.

### jannovar-htsjdk

* Bumping HTSJDK dependency to v2.20.3 because 2.20.0 has a bug in the␣
↪VariantContextBuilder


## v0.33

### overall

* Fixing `ANN` annotation field to have `.`/unbound cardinality.
* Bumping several dependencies, including HTSJDK.

### manual

* Adding conda support in installation documentation
* Fixing broken link in quickstart

### jannovar-core

* Fixing annotation of SVs that look like sequence variants (#456).
  Interpretation is to use the sequence variant annotation code now.
  This fixes a bug with annotating latest ClinVar for GRCh38.
* Prevent `Annotation.{getPutativeImpact,getPutativeImpact}()` from returning `null` (
↪#458).
* Correctly parsing RefSeq mitochondrial transcripts.
  Bumping the required versin to `0.33-SNAPSHOT` to higlight this (#463).

## v0.32

### overall

* Changing log4j version to 2.11.2
* Changing slf4j version to 1.7.25

### jannovar-cli

* Adding a simple REST server for annotating single variants.
    * Launch with `jannovar-cli rest-server -d data/hg19_refseq.ser -d data/hg19_
↪ensembl.ser`
    * Then, query with `/annotate-var/refseq/hg19/chr7/140453136/A/T` or `/annotate-
↪var/ensembl/hg19/chr7/140453136/A/T`

## v0.31

### jannovar-core

* Introducing classes for representation of gapped sequences, alignments, and␣
↪position projection.
* Fixing bug in ENSEMBL transcript database generation (tx version was appended twice)
* Adding flags for "has substitutions" and "has indels" to `TranscriptModel` that get␣
↪filled for RefSeq transcripts.
* Correctly parsing of RefSeq transcripts with indels.
```

(continues on next page)

(continued from previous page)

```
### jannovar-cli

* Using HTTP protocol instead of FTP everywhere as it's possible (#451).
* Fixing HGVS conversion for indels (#452).

## v0.30

Bugfix release.

### jannovar-core

* Fixing interpretation of `INFO/SVTYPE`, urldecode and only use the first component␣
↪after splitting at ":".
* Interpreting SV type annotation for SV2 more correctly.

## v0.29

### jannovar-core

* Putative impact of splice_region_variant has changed from MODERATE to LOW (see␣
↪issue #439)
* Decreasing log verbosity in one location when building database.
* Fixing CDS region import in `RefSeqParser`
* Putative impact of `splice_region_variant has` changed from MODERATE to LOW (see␣
↪issue #439)
* Fixing SV annotation using hg38/ucsc for transcripts without gene ID (see #444).
* Adding support for rn6 RefSeq transcripts.
  Adding `allowNonCodingNm` directive for data source INI file to disable check that␣
↪RefSeq NM transcript has CDS.
* Adding versions to ENST accessions for ENSEMBL.

### jannovar-vardbs

* Bugfix: TSVAnnotator did not use end given column.

## v0.28

### jannovar-cli

* Bumping ENSEMBL versions for GRCh37 and GRCh38.
* Fixing sources information for updated ENSEMBL downloads.
* For ENSEMBL, use ENSEMBL-provided mapping from ENSG to HGNC ID for Entrez ID␣
↪assignment.
  This is necessary as Ensembl gene IDs turn out to be not so stable between hg37 and␣
↪hg38 after all.
  Case in point: `ENSG00000276141` vs. `ENSG00000187667`.
* Adding `--gene-ids` argument to downloader for creating smaller databases (mostly␣
↪for test purposes).
* Adding SV support to jannovar-cli, includes tests.
* Using ENSEMBL-provided mapping from ENSG to Entrez ID in the case HGNC mapping does␣
↪not work.

### jannovar-htsjdk

* Adding SV support to jannovar-htsjdk
* Bumping HTSJDK dependency to v2.18.2
```

(continues on next page)

(continued from previous page)

```
### jannovar-core

* Changing upstream/downstream size to 5kbp.
* Support for prioritizing RefSeq transcript on the PAR of chrX over those of chrY
* Refactorizations to improve performace using `EnumSet`.
* Extended `VariantEffect` for the effects of structural variants.
  Removing documentation that the effect is not used in Jannovar for some now␣
↪interpreted ones.
  Also variant effect for non-coding variants is added using the [current VEP␣
↪predictions](https://www.ensembl.org/info/genome/variation/prediction/predicted_
↪data.html) as a template.
* Prohibiting creating `GenomeVariant` with symbolic alleles.
  Throwing new checked exception `InvalidGenomeVariant` case of error.
* Fixing SO term ID for `VariantEffect.DISRUPTIVE_INFRAME_DELETION`
* Correctly parsing transcript version for ENSEMBL when available (not available for␣
↪b75/GRCh37).
* Making transcript model building (for `download`) more memory efficient.

## v0.27

### jannovar-cli

* Integrating support for thousand genomes VCF
* Integrating thousand genomes/ExAc count limits into inheritance filter

### jannovar-vardbs

* Adding support for thousand genomes VCF

### jannovar-htsjdk

* Adding support for limiting genomes/ExAc counts into inheritance filter

## v0.26

### jannovar-cli

* Making `OneParentGtFiltered` filter optional. The default setting to `false`␣
↪(specify `--one-parent-gt-filtered-filters-affected` to enable).

### jannovar-core

* **Moving variants in non-coding transcripts after UTR variants.**

### jannovar-hgvs

* Fixing parser issue for nucleotide indels (#408).

### jannovar-htsjdk

* Obey the `options.escapeAnnField` parameter for escaping the variant effect in the␣
↪`ANN` field.

## v0.25

### overall
```

(continues on next page)

(continued from previous page)

```
* Changing HTSJDK version to 2.14.3
* Using the one letter amino acid code in HGVS representation as default (changes in␣
↪core, hgvs, htsjdk and cli). Now the cli option `--3-letter-amino-acids` works as␣
↪expected.

### jannovar-cli

* Support for [RefSeq GRCh37.p13 interim release](https://www.ncbi.nlm.nih.gov/books/
↪NBK430989/#_news_02-14-2017-interim-annotation-update-human_)
* Support of new RefSeq headers
* Using RefSeq GRCh38.p12 annotation instead of GRCh38.p7

### jannovar-vardbs

* Replacing whitespace with string when annotating from TSV file.

### jannovar-htsjdk

* Fixing bug in GenomeRegionSequenceExtraction. Error reports always sequences from␣
↪the first contig in the referebnce file and not the requested contig. Affects only␣
↪the cli command `hgvs-to-vcf`.

## v0.24

### jannovar-cli

* Fixing annotation with Polyphen prediction (data type)

### overall

* Changing HTSJDK version to 2.14.0
* Codestyle improvements

### jannovar-core

* Fixing mendelian "bug" #393 (has no affect because check was not necessary)
* New inheritance mode: mitochondrial
* Bugfix ProgressBar (doPrint was always true)

### jannovar-vardbs

* Fixed problem with interpretation of Clinvar annotation origin.
* Clinvar `BEST_AC` and `BEST_AF` are now named `AC_POPMAX` and `AF_POPMAX` to be␣
↪consitent with gnomAD

## v0.23

### overall

* Changing Guava version to 0.22
* Changing slf4j version to 1.7.24
* Changing log4j version to 2.8.2

### jannovar-cli

* Adding experimental support for annotating with VCF files.
* Adding experimental support for annotating with tabix-indexed TSV files and dbNSFP.
```

(continues on next page)

(continued from previous page)

```
* Integrating the advanced pedigree-based filters (useful for filtration to de novo␣
↪variants).
* Making it possible to override database INI settings using user-specified INI files.

### jannovar-core

* Fixing stop loss annotation (#351).
* Finishing renaming of TranscriptInfo to TranscriptModel (#348).
* Upstream and downstream variant were considered "not off exome". They now are.
* Adding mitochondrial filtering function (#362).

### jannovar-filter

* Adding code for performing more advanced filtration/annotation filtering to de novo␣
↪variants.
* Improving documentation of `MaxFreqAr` and `MaxFreqAd` in header.


### jannovar-vardbs

* Adding experimental support for annotating with VCF files
* Adding experimental support for annotating with tabix-indexed TSV files and dbNSFP

### jannovar-filter

* Fixing bug that ignored variant filters for recessive annotation

## v0.22

### jannovar-htsjdk

* Fixin NPE problem with inheritance annotation

### jannovar-statistics

* Also counting number of variants on contigs
* Fixing counting bug that made UTR3 variants be counted as UTR5
* Fixing NPE in case of null variant annotations (e.g., unknown contig)

### jannovar-vardbs

* Fixing a problem with normalization on variant annotation
* Fixing problem with default value of `CLNSIG` (`"25"` -> `"255"`)

### jannovar-filter

* Incorporating gnomAD annotation into exclusion by frequency for inheritance filter (
↪#343)
* Fixing header description for `MinAafHomAlt` and `MaxAafHomRef` (#342)

### jannovar-cli

* Checking that reference is given also for gnomAD VCF annotation

## v0.21

### all
```

(continues on next page)

(continued from previous page)

```
* Fixing language in mvn surfire plugin. Now mvn tests work on locale de_DE etc..

### jannovar-cli

* Adding `--interval` argument for only processing a part of the file
* Adding `statistics` command for computing statistics on variants in VCF file
* Fixing bug in HGVS to VCF
* Better handling missing `.dict` file for HGVS to VCF translation
* Adding `--annotate-as-singleton-pedigree` parameter for annotation of singleton␣
↪pedigrees without pedigree file (single individual is assumed to be affected)
* More friendly user message in case of unsorted files on inheritance mode annotation
* Interpretation of filters in compatible inheritance mode annotation
* Integrating new jannovar-filter into Jannovar CLI.
  Filtered genotypes will be passed into the inheritance filter as no-call.
* Adding annotation with ClinVar
* Printing warnings next to the annotations in `annotate-pos`
* AR inheritance annotation of two siblings bugfix (no parents avaiable in comp.het␣
↪mode) #314

### jannovar-filter

* Adding functionality to add filters based on frequencies found in dbSNP and ExAC
* Adding back as module for threshold-based filtration.
  This module allows to create genotype-wise soft-filters for low coverage.
  Also, variants can be soft-filtered based on whether the genotype calls of all␣
↪affected individuals are filtered out.

### jannovar-core

* Extending API to expose mendelian checks for comp het./ad alt (via␣
↪`SubModuleOfInheritance` and `MendelianInheritanceChecker`
* Jannovar version is now written out to database file which allows better error␣
↪checks and compatibility messages
* Un-deprecating `BestAnnotationListTextGenerator` and␣
↪`AllAnnotationListTextGenerator` classes, useful for text-based output formats
* Changing behaviour of `VariantEffect.isOffExome()` and adding a variant that allows␣
↪to decide between UTR on/off exome and non-consensus splice region on/off exome
* Making the behaviour of overriding transcripts configurable at least in the code,␣
↪using default to not do this any more
* Adding `WARNING_REF_DOES_NOT_MATCH_TRANSCRIPT` to `AnnotationMessage`
* Properly pushing through warnings from the annotators into the returned␣
↪`VariantAnnotation` object
* Pedigree files are now more compatible to the PLINK format
        * whitespace separated instead of tab separated (read only, written as TSV)
        * interpreting any value not in {1, 2} to be "unknown" sex instead (coded as␣
↪0) of throwing

### jannovar-htsjdk

* Fixing bug in transcript-to-genome translation, in HGVS the stop codon is not part␣
↪of the CDS but in `TranscriptModel` it is
* Optional interpretation of certain filters in GeneWiseMendelianAnnotationProcessor.
* Extending interface of `VariantContextAnnotator` for automatic error annotation␣
↪generation, previously in jannovar-cli
* Adding `VariantEffectHeaderExtender` class to `jannovar-htsjdk`
* Fixing bug with problems of unmodifiable Attributes (error annotation).
```

(continues on next page)

(continued from previous page)

```
### jannovar-vardbs

* Also writing out variant allele origin for dbSNP
* Adding annotation with COSMIC
* Fixing header description for exac database
* Fixing output of `DBSNP_CAF` to also contain reference allele AF
* Adding annotation with ClinVar, can annotate all clinvar variants

### jannovar-inheritance-checker

* Removing this outdated module.
  Use the classes in `de.charite.compbio.jannovar.mendel` instead

### jannovar-stats

* all-new module for gathering statistics on VCF files

## v0.20

### all

* Change email/organisations in master pom

### jannovar-core

* `GenotypeCalls.getGenotypeForSample()` returns a "no-call" genotype now instead of␣
↪`null`

### jannovar-htsjdk

* fix to annotation with compatible mode of inheritance (#289)
* update to htsjdk 2.8.1

### jannovar-cli

* removing requirement for proper contig `contig` lines in gene-wise gene annotation
* fixing NPE in the case of no `contig` lines
* improving error message on samples in VCF file that are not in pedigree
* fix to annotation with compatible mode of inheritance (#289)
* better overview on CLI help message
* if ref-fasta is not set properly a nicer error message will be shown.

### jannovar-vardbs

* Fixing bug with problems of unmodifieable Attributes.
* Including Hom/Het/Hemi counds of ExAC (#295)
* update to htsjdk 2.8.1

## v0.19

This is a bugfix release.

### manual

* Manual loads version from central POM file now
* Adjusting manual links to point to `javadoc.io`
```

(continues on next page)

(continued from previous page)

```
### jannovar-core

* Fixing integration of HGNC into the downloaded databases
    * For UCSC, HGNC records are searched based on the Entrez ID.
      If HGNC does not know the Entrez then only the Entrez ID from UCSC is written␣
↪as additional ID.
    * For RefSeq, linking is done through Entrez ID.
      If HGNC does not know the Entrez then only the Entrez ID from RefSeq is written␣
↪as additional ID.
    * For ENSEMBL linking is done through the ENSEMBL gene id.
      If this is not known to HGNC then no additional IDs are annotated.
* Fixing problem with `UnsupportedOperationException` in `jannovar-htsjdk`

## v0.18

### all

* replace charite email of p. robinson with the new one of jax

### jannovar-cli

* Renaming `tx-to-chrom` to `hgvs-to-vcf`, also in Java module names.
* CLI changes such that one VCF input and one VCF output path can be used only
* Replacing apache commons-cli with argparse4j for a more modern and usable CLI
* Consistently writing out HUGO symbols for gene names, using the `hgnc_complete_set.
↪txt` information downloaded when building the annotation DB
* Upgrading from ENSEMBL-74 to ENSEMBL-75 for annotation database files
* Removing support for old Jannovar-style annotations (#241)
* Adding new command for annotating csv files (annotate-csv)

### jannovar-htsjdk

* Properly annotating Mendelian inheritance for intergenic variants

### jannovar-core

* downloading `hgnc_complete_set.txt` together with data sets, `TranscriptModel`␣
↪objects now consistently contain additional IDs
* making ENSEMBL parsing more robust (falling back to transcript name if no␣
↪transcript ID)
* fixing bug #248 for ENSEMBL that used `gene_id` for `gene_name`
* bugfix of NullPointerException in RefSeqParser while parsing refSeq curated
* bugfix space in SeqOID of SYNONYMOUS_VARIANT
* Update link to HGVS Nomenclature
* Now BestAnnotationListTextGenerator shows really the best and not all annotations!

### Manual

* Documenting cli changes
* Adding additional sites contributing, FAQ and how to filter
* Better description of installations and quickstart


## v0.17

### jped-cli
```

(continues on next page)

(continued from previous page)

```
* this is gone, the functionality is now available as part of jannovar-cli

### jannovar-filter

* this module is done, everything here is merged into jannovar-htsjdk

### jannovar-vardbs

* The first version ships with support for dbSNP b147, ExAC 0.3, and the UK10K COHORT␣
↪data base
* Initial version of this module, the aim is precise annotation from variant databases

### jannovar-cli

* Updated `default_sources.ini` for latest patches of mouse and human genomes
* Using one-letter amino acid code by default
* Removed slf4j2 warning at program startup
* Checking pedigree for compatibility with VCF file if given

### jannovar-core

* Adjusting API for annotating amino acid code by default
* Checking pedigree for compatibility with genotypes on Mendelian inheritance checking
* Refurbishing `Genotype`, `GenotypeList`, and `GenotypeListBuilder` in `de.charite.
↪compbio.jannovar.mendel`.
* Moving `ModeOfInheritance` to `de.charite.compbio.jannovar.mendel`.
* Creating new package `de.charite.compbio.jannovar.mendel` with code for filtering␣
↪for mendelian inheritance modes.
* Renaming of `ModeOfInheritance.UNINITIALIZED` to `ModeOfInheritance.ANY`.
* Fixing handling of invalid transcripts (e.g., incomplete 3' end)
* Adding `altGeneIDs` mapping to `TranscriptModel`, makes data bases backwards␣
↪incompatible.
* Rewrite of GFF parsers for RefSeq and ENSEMBL.
* Bumping HTSJDK to 2.5.0, requiring Java 8 from now on.
* Removal of `AnnotationCollector`, priotization of variant effects is done after␣
↪collecting all effect predictions now.
* Fix for intronic variants between 5' or 3' UTRs. These variants were misclassified␣
↪as `FIVE_PRIME_UTR_VARIANT` or `THREE_PRIME_UTR_VARIANT`. SequenceOntology␣
↪implements new terms so that we can decide between the two UTR exon and intron␣
↪variants. Now we have `FIVE_PRIME_UTR_EXON_VARIANT` or `FIVE_PRIME_UTR_EXON_INTRON_
↪VARIANT` (the same for `THREE_PRIME_UTR_EXON_VARIANT` or `THREE_PRIME_UTR_EXON_
↪INTRON_VARIANT`).

### jannovar-cli

* Adding better progress display with estimate of pending time.
* Adding support for annotating values from dbSNP VCF file (currently, only b147 is␣
↪supported).
* Adding simple progress reporting (from verbosity level 2).
* Using Java 8 stream interface for `VariantContext` processing.
* Removing support for Jannovar output format, VCF offer all features and more.

## v0.16

### jannovar-cli
```

(continues on next page)

(continued from previous page)

```
* Updating htsjdk to 1.142
* using simple logger of slf4j
* fixing version output in command line help
* changing command line interface to use more named arguments
* removing deprecated usage of commons-cli command line parser
* renaming of some internal classes and functions, fixing Javadocs

### jannovar-core

* fixing bug in `TranscriptSequenceChangeHelper` for reverse transcript (did not␣
↪reverse complement alternate allele)
* fixing bug in parsing GFF3 with some transcripts (e.g. GNAT1)
* less intrusive escaping in `ANN` field
* renaming of some internal classes and functions, fixing Javadocs

### jannovar-htsjdk

* Updating htsjdk to 1.142
* renaming `InvalidGenomeChange` to `InvalidGenomeVariant`
* renaming `VariantContextAnnotator.buildGenomeChange` to `.buildGenomeVariant`
* renaming of some internal classes and functions, fixing Javadocs

### jannovar-hgvs

* extending API of ProteinChange hierarchy for HGVS generation
* renaming of some internal classes and functions, fixing Javadocs

### jped-cli

* Updating htsjdk to 1.142
* changing command line interface to use more named arguments

### jannovar-inheritance-checker

* adding two new functions to InheritanceCompatibilityChecker
* resolve boolean if passes inheritance into set where passed inheritances are stored
* Updating htsjdk to 1.142

### manual

* updating manual for 0.16 and using parameters for commands!
* updating readme for parameters

## v0.15

### jannovar-core

* making `CompatibilityCheckerAutosomalRecessiveHomozygous` public
* using jannovar-hgvs for representing the changes
* more precise HGVS annotation in some cases
* predictions are wrapped in parentheses
* Mark everything that is related to the compatibility checkers as depricated (see␣
↪new jannovar-inheritance-checker)

### jannover-hgvs

* adding module for parsing and representing HGVS-compatible nucleic and protein␣
↪changes
```

(continues on next page)

(continued from previous page)

```
### jannover-htsjdk

* Updating htsjdk to 1.138
* Replacing depricatded method `VariantContext.getChr()` with `VariantContext.
↪getContig()`

### jannovar-cli

* Updating htsjdk to 1.138
* Replacing depricatded method `VariantContext.getChr()` with `VariantContext.
↪getContig()`
* Updating commons-cli to 1.3.1

### jannover-inheritance-checker

* Bugfix detecting autosomal chromosomes
* Bugfix with handling variant files with a leading "chr" in the contig.
* Adding this new module.
* Replaces the compatibility checker oh jannobvar-core.
* Now runs with VariantContext (htsjdk) instead of Jannovar Genotypes
* Use `InheritanceCompatibilityChecker.Builder` to build␣
↪`InheritanceCompatibilityChecker`.
* Use the method `getCompatibleWith` of the `InheritanceCompatibilityChecker` with a␣
↪List of `VariantContext`.
* The method will return all `VariantContext` that matches the inheritance. If no␣
↪variant matches the List is empty.

### jannover-filter

* Refactoring `VariantWiseInheritanceFilter` to handle the new␣
↪`InheritanceCompatibilityChecker`.
* Rewrite `GeneWiseInheritanceFilter` to handle the new␣
↪`InheritanceCompatibilityChecker`.
* Updating htsjdk to 1.138
* Replacing depricatded method `VariantContext.getChr()` with `VariantContext.
↪getContig()`

### jped-cli

* Adapting program to the `GeneWiseInheritanceFilter` and␣
↪`VariantWiseInheritanceFilter` (see jannovar-filter)
* Updating commons-cli to 1.3.1
* Changing cli option inheritance-mode to multiple args (Now you can check multiple␣
↪inheritances at once)

## v0.14

### jannovar-cli

* Improving output file generation, jannovar-cli now uses the same extension
  as in the input and the infix is configurable instead of being fixed to
  ".jv".
* Default extension is ".vcf.gz" instead of ".vcf" now.

### jannovar-core
```

(continues on next page)

(continued from previous page)

```
* Fixing label for `FRAMESHIFT_VARIANT` in `VariantEffect`.
* Moving CompatibilityCheckerException to package
  `...jannovar.pedigree.compatibilitychecker`
* Fixing bug in transcript coordinate projection.
* Renaming `TranscriptSequenceChangeHelper.getCDSWithChange` to
  `.getCDSWithGenomeVariant`.
* Renaming `*.getChange()` to `*.getGenomeVariant()`
* Renaming `VariantAnnotator.buildAnnotationList` to `.buildAnnotations`,
  `VariantContextAnnotator.buildAnnotationList` to `.buildAnnotations`,
  and `VariantContextAnnotator.buildErrorAnnotationList` to
  `VariantContextAnnotator.buildErrorAnnotations`
* VariantAnnotations does not implement `List<Annotation>` any more
* Adding `VariantAnnotations.getAnnotations`
* Renaming `AnnotationList` to `VariantAnnotations`
* changing treatment of insertions at exon/intron junctions; they are
  considered as intronic insertions now that affect splicing
* converting `GenomeVariant` of `AnnotationList` to always be on the forward
  strand after construction of  `AnnotationList`
* deprecating the `{,All,Best}AnnotationTextGenerator` classes

## v.0.13

### jannovar-cli

* Moving `JannovarOptions` into jannovar-cli.
* Displaying online help on unknown Jannovar command.
* Fixing `NullPointerException` bug for local paths.
* Switching to official HTSJDK release and version 0.128.
* Writing out annotation about Jannovar call and version into the VCF file.
* Added option `--no-3-prime-shifting` to disable shifting towards the
  3' end of the transcripts.
* Added option `--no-escape-ann-field` to disable escaping of the `ANN`
  `INFO` field.
* Variants in `ANN` field are now annotated with proper Sequence Ontology
  terms.

### jannovar-htsjdk

* Modified `VariantContextWriterConstructionHelper` to allow explicit
  disabling of index creation.
* Modified `VariantContextAnnotator` for adjustment to the new Exomiser.
* Switching to official HTSJDK release and version 0.128.
* Changing `VariantContextWriterConstructionHelper` to allow writing out
  of additional header lines.
* Added option to `VariantContextAnnotator#Options` for disabling
  3' shifting.
* Modified `VariantContextAnnotator` allowing to disable escaping of the
  `ANN` `INFO` field.

### jannovar-core

* Moving `JannovarOptions` into jannovar-cli.
* Renaming `ACompatibilityChecker` and `ICompatibilityChecker`.
* Adding `GenomePosition.differenceTo(GenomeInterval)`.
* Renaming package `de.charite.compbio.jannovar.io` to `de.charite.compbio.jannovar.
↪data`
```

(continues on next page)

(continued from previous page)

```
* Renaming `AnnotationLocation.toHGVSString` to `.toHGVSChunk`.
* Adding `Pedigree.subsetOfMembers`
* Renaming `GenomeChange` to `GenomeVariant`, same with types having the same
  prefix.
* Introducing `DatasourceOptions` for configuring data download.
* Removing support for using `"-"` as REF or ALT value.
* Making previous `public final` members `private final` (or
 `protected final`) and adding getters for read-only access to them.
* Removing position type member of `CDSInterval`.
* Using type `Strand` instead of `'+'` and `'-'`, requires database rebuild.
* Adding enum `Strand` with `PLUS` and `MINUS` values.
* Adding `VariantEffect.isOffExome` and updating
  `VariantEffect.isOffTranscript`.
* Removing `genomeRegion` member from `GenotypeList`. Also, adjusting the
  pedigree compatibility checkers for this, the check for being on the X
  chromosome has to be performed outside the checker now.
* `VariantList.getHighestImpactEffect` now returns
  `VariantEffect#SEQUENCE_VARIANT` if no annotation can be found.
* `VariantList` implements the `List<Annotation>` interface now and the
  `entries` member has become private.
* Adding `VariantEffect#SEQUENCE_VARIANT` for variants with unknown
  effects.
* `GenomeChange.toString()` now always converts to forward strand.
* Fixing bug in `Annotation` and enforcing forward strand `GenomeChange`
  instances.
* Updates to the manual.
* `JannovarData` now also stores a mapping from transcript accession to
  `TranscriptModel` and from gene symbol to `TranscriptModel`.
* Adding functionality for conversion from CDS to transcript and genome
  position and tests.
* Adding `AnnotationBuilderOption` object that allows disabling of 3'
  shifting towards the transcript.
* Adding `JannovarOptions#escapeAnnField`.
* Renaming `VariantType` to `VariantEffect`
* Changing `VariantType` to use proper Sequence Ontology terms. Legacy
  names can be obtained through `VariantType#getLegacyName`.
* Spliting `CompatibilityCheckerXRecessive` into␣
↪`CompatibilityCheckerXRecessiveCompoundHet` and␣
↪`CompatibilityCheckerXRecessiveHomozygous`. Now all inheritance checkers ar ready␣
↪to use (AR,XR,AD,XD)
* move all pedigree compatibility checkers from `de.charite.compbio.jannovar.
↪pedigree` to `de.charite.compbio.jannovar.pedigree.compatibilitychecker` and divide␣
↪it into ar,xr,ad,xd.
* generate interface `ICompatibilityChecker` for pedigree compatibility checkers.
* Combine compatibility fields and methods in an abstract␣
↪class`ACompatibilityChecker` to unify methods, builders, and fields.

### jannovar-filter

* Splitting into `jped-cli` and `jannovar-filter`
* Changing public final members to accessors.
* `jannovar-filter` now has the Jannovar DB as the mandatory first argument.

### jannovar-htsjdk

* Changing public final members to accessors.
```

(continues on next page)

(continued from previous page)

```
## v0.12

### jannovar-htsjdk

* Started bridge module between Jannovar and HTSJDK.

### jannovar-filter

* Started tool for mode of inheritance--based filters.

### jannovar-cli

* Splitting out bridge module between jannovar-core and HTSJDK to
  jannovar-htsjdk.
* Adding implementation of variant annotation standard 1.0.
* Adding unit tests for jannovar-cli.
* Fixing problem with empty `INFO` fields in output.
* Adding back `--output-dir` to jannovar-cli.
* Writing output parallel to input file by default.
* Adding `-v` and `-vv` command line options.
* Fixing problems with block substitution (delins) case (#87).

### jannovar-core

* Adding initial support for the transcript support level feature of the new VCF
  annotation standard (only in very recent ENSEMBL releases, apparently).
* `TranscriptModel#geneID` is now a `String`
* Update in various classes, e.g. Annotation.
* Fixing bug in PED parsing (empty lines are properly skipped now).
* More tests and fixes for the inheritance compatibility checkers.
* Updating `Annotation` for the variant annotation standard.
* `TranscriptPosition` and `TranscriptInterval` use zero-based positions now.
* Reordering values of `VariantType`.
* Somewhat renaming `VariantType` method names.
* Removing the `VariantType#size` function in favor of a `static public`
  `final` member.
* Using log4j/slf4j for I/O in jannovar-core.
* Adding `PrintStream` as parameter to `JannovarOptions#print`.
* Compressing serialized file.
* Changing namespace to `de.charite.compbio.jannovar`.
* Making `VariantType#priorityLevel` a non-static member.
* Renaming `TranscriptInfo` to `TranscriptModel`.
* Moving `HG19RefDictbuilder` from tests to main.
* Using `ImmutableMap` in `Translator` for small performance improvements.
* Using `StringBuilder`-based concatenation of strings for generation of HGVS
  strings etc. since this is much faster than using `String#format`.
* `GenomePosition` and `GenomeInterval` use zero-based coordinates internally
  now.

## v0.11
```

4.20 Jannovar License

Jannovar is licensed under the 3-Clause BSD License: