VCF and Variant Annotation

Under construction

The variant annotation tool is integrated within the CellBase code and can be accessed in a number of different ways.

Annotate VCF files using the CLI

Please, check the github Readme (https://github.com/opencb/cellbase) to learn how to compile the CellBase code.

Once compiled, you can use the *variant-annotation* command provided within the *cellbase.sh* CLI:

```
cellbase$ build/bin/cellbase.sh variant-annotation -h

Usage:   cellbase.sh variant-annotation [options]

Options:
      -a, --assembly                  STRING     Name of the assembly, if empty the first assembly in
configuration.json will be read
          --batch-size                INT        Number of variants per batch [200]
          --chromosomes               STRING     Comma separated list (no empty spaces in between) of
chromosomes to annotate. One may use this
                                                 parameter only when the --input-variation-collection flag is
activated. Variants from all
                                                 chromosomes will be annotated by default. E.g.: 1,22,X,Y
      -C, --config                    STRING     CellBase configuration.json file. Have a look at
                                                 cellbase/cellbase-core/src/main/resources/configuration.json
for an example
          --custom-file               STRING     String with a comma separated list (no spaces in between) of
files with custom annotation to be
                                                 included during the annotation process. File format must be
VCF. For example: file1.vcf,file2.vcf
          --custom-file-fields        STRING     String containing a colon separated list (no spaces in
between) of field lists which indicate the
                                                 info fields to be taken from each VCF file. For example:
                                                 field1File1,field2File1:field1File2,field3File2
          --custom-file-id            STRING     String with a comma separated list (no spaces in between) of
short identifiers for each custom file.
                                                 For example: fileId1,fileId2
          --exclude                   STRING     Comma separated list of annotators to be excluded
          --gzip                                 Whether the output file is gzipped [false]
      -h, --help                                 Display this help and exit [false]
          --include                   STRING     Comma separated list of annotators to be included
      -i, --input-file                STRING     Input file with the data file to be annotated
          --input-variation-collection           Input will be a local installation of theCellBase variation
collection. Connection details must be
                                                 properly specified at a configuration.json file [false]
      -l, --local                                Database credentials for local annotation are read from
configuration.json file [false]
      -L, --log-level                 STRING     Set the logging level, accepted values are: debug, info,
warn, error and fatal [info]
      -t, --num-threads               INT        Number of threads to be used for loading [4]
    * -o, --output                    STRING     Output file/directory where annotations will be saved. Set
here a directory if flag
                                                 "--input-variation-collection" is activated (see below). Set
a file name otherwise.
          --output-format             STRING     Variant annotation output format. Values: JSON, PB, VEP [JSON]
          --remote-url                STRING     The URL of CellBase REST web services, this has no effect if
--local is present
                                                 [bioinfo.hpc.cam.ac.uk:80/cellbase/webservices/rest]
          --resume                               Whether we resume annotation or overwrite the annotation in
the output file [false]
    * -s, --species                   STRING     Name of the species to be downloaded, valid format include
'Homo sapiens' or 'hsapiens' [Homo
                                                 sapiens]
          --variant                   STRING     A comma separated variant list in the format chr:pos:ref:alt,
ie. 1:451941:A:T,19:45411941:T:C
      -v, --verbose                   BOOLEAN    [Deprecated] Set the level of the logging [false]
```

A typical run of the variant annotator would use default values for most of parameters. For example:

```
cellbase$ build/bin/cellbase.sh variant-annotation -i /tmp/test.vcf.gz -o /tmp/test.json.gz --species hsapiens
--assembly GRCh37
```

By default, the variant annotator will call remote web services deployed at the University of Cambridge (http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/). If a local installation of the CellBase database is available, the *--local* flag can be enabled to improve annotation performance; please, check the github readme (https://github.com/opencb/cellbase) to learn how to build the code and point it to a particular local database.

Variant annotation including custom files

In many occassions, variant annotation users want to complement core CellBase annotation with custom annotations either from own generated files or other resources not currently integrated in CellBase database. Thus, the variant annotation command line allows the user can specify a set of files with custom annotation data. Currently, only VCF files are allowed as custom input. Any attribute value in the INFO field can be used during the annotation process. Three parameters in the command line control custom annotation behaviour:

```
          --custom-file               STRING     String with a comma separated list (no spaces in between) of
files with custom annotation to be
                                                 included during the annotation process. File format must be
VCF. For example: file1.vcf,file2.vcf
          --custom-file-fields        STRING     String containing a colon separated list (no spaces in
between) of field lists which indicate the
                                                 info fields to be taken from each VCF file. For example:
                                                 field1File1,field2File1:field1File2,field3File2
          --custom-file-id            STRING     String with a comma separated list (no spaces in between) of
short identifiers for each custom file.
                                                 For example: fileId1,fileId2
```

1. First, you will need to specify the set of files that should be used as a custom annotation input. You can do this by using the `--custom-file` parameter and providing a comma separated list (no spaces in between) of VCF files. For example, /tmp/file1.vcf,/tmp/file2.vcf,/tmp/file3.vcf.
2. Next, you need to specify which INFO attributes must be taken from each of the files. You can do this by using the `--custom-file-fields` parameter, and providing a colon separated list (no spaces in between) of info tag lists. For example, field1File1,field2File1:field1File2,field3File2
3. Last, you need to provide one label or tag per file. This label will be used in the output file to represent this file in the variant annotation ouput. You can do this by using the `custom-file-id` parameter, and provinding a comma separated list (no spaces in between) of short labels/tags/identifiers.

A variant-annotation command including three custom files would look like:

```
$ cellbase$ build/bin/cellbase.sh variant-annotation -i /tmp/test.vcf.gz \
  -o /tmp/test.json.gz \
  --species hsapiens \
  --assembly GRCh37 \
  --custom-file /tmp/file1.vcf,/tmp/file2.vcf,/tmp/file3.vcf \
  --custom-file-fields AF,DP:AF,DP:AF,DP
  --custom-file-id file1,file2,file3
```

In this example, three files (file1.vcf, file2.vcf, file3.vcf) will be used as a source of custom annotation. Values of the AF and DP attributes will be the annotation data taken from the three files. Finally, labels `file1`, `file2`, `file3` will identify the file from which each AF and DP value were read from.

If annotating with custom files, the first step run by the `variant-annotation` command will consist of creating a RocksDB database per custom file. RocksDB (rocksdb.org) is a high performance key-value store which has shown to be extremely useful for this kind of use-case. Thus, each database will contain one entry for each variant in the VCF; the key being a variant id and the value the corresponding AF value. The RocksDB index path will share the prefix with the custom file and will end in `.idx`. In our example, three RocksDB indexes would be created:

```
/tmp/file1.vcf.idx
/tmp/file2.vcf.idx
/tmp/file3.vcf.idx
```

It is important to note that these indexes need to be created just once. Posterior runs of the variant-annotation command using any of these custom files wont need to re-index them and will directly read from the index. Please, also note, that if the INFO attributes to be used vary, these indexes must be manually removed from the filesystem to force the variant-annotation tool to re-index the three files with the new attributes.

During the annotation process, the annotator will merge core annotation coming from the Mongo queries to the main CellBase database, with these custom annotations coming from queries to the RocksDB indexes. Both annotation results will be integrated into VariantAnnotation objects (https://github.com/opencb/biodata/blob/develop/biodata-models/src/main/avro/variantAnnotation.avdl). Custom annotations will be stored within the additionalAttributes field, looking like this:

```
{
  "names": [],
  "chromosome": "1",
  "start": 10428,
  "reference": "",
  "alternate": "C",
  "studies": [ ... ],
  "type": "INDEL",
  "annotation": {
    "chromosome": "1",
    "start": 10428,
    "reference": "",
    "alternate": "C",
    "displayConsequenceType": "2KB_upstream_variant",
    "consequenceTypes": [ ... ],
    "additionalAttributes": {
      "file1": {
        "attribute": {
          "AF": "0.82",
          "DP": "4"
        }
      },
      "file2": {
        "attribute": {
          "AF": "0.32",
          "DP": "42"
        }
      },
      "file3": {
        "attribute": {
          "AF": "0.2",
          "DP": "14"
        }
      }
    }
  },
  "end": 10427,
  "strand": "+",
  "hgvs": {},
  "length": 1
}
```


Using the Python client (PyCellBase)

A wrapper method is implemented within the VariantClient object for performing variant annotation. Thus, variant annotation can be carried out by simply calling this "get_annotation" method:

```
>>> from pycellbase.cbconfig import ConfigClient
>>> from pycellbase.cbclient import CellBaseClient
>>> cbc = CellBaseClient(cc)
>>> result = variant_client.get_annotation("19:45411941:T:C,1:69585:TGAGGTCGATAGTTTTTA:-,14:38679764:-:
GATCTGAGAAGNGGAANANAAGGG,19:33167329:AC:TT,22:16328960-16344095:<CN5>,22:16328960-16344095:<CN1>,22:16328960-
16344095:<CNV>,22:16328960-16344095:<DEL>,22:16328960-16344095:<DUP>,22:16328960-16344095:<INV>")
>>> len(result)
  10
>>> result[8]["result"][0]
  {u'alternate': u'<DUP>',
 u'chromosome': u'22',
 u'consequenceTypes': [{u'biotype': u'unprocessed_pseudogene',
   u'ensemblGeneId': u'ENSG00000234381',
   u'ensemblTranscriptId': u'ENST00000435410',
   u'geneName': u'MED15P7',
   u'sequenceOntologyTerms': [{u'accession': u'SO:0001889',
     u'name': u'transcript_amplification'}],
   u'strand': u'+',
   u'transcriptAnnotationFlags': [u'basic']},
  {u'biotype': u'unprocessed_pseudogene',
   u'ensemblGeneId': u'ENSG00000224435',
   u'ensemblTranscriptId': u'ENST00000426025',
   u'geneName': u'NF1P6',
   u'sequenceOntologyTerms': [{u'accession': u'SO:0001636',
     u'name': u'2KB_upstream_variant'}],
   u'strand': u'+',
   u'transcriptAnnotationFlags': [u'basic']},
  {u'sequenceOntologyTerms': [{u'accession': u'SO:0001566',
     u'name': u'regulatory_region_variant'}]}],
...
...
...
>>> result[8]["result"][0]["consequenceTypes"]
  [{u'biotype': u'unprocessed_pseudogene',
  u'ensemblGeneId': u'ENSG00000234381',
  u'ensemblTranscriptId': u'ENST00000435410',
  u'geneName': u'MED15P7',
  u'sequenceOntologyTerms': [{u'accession': u'SO:0001889',
    u'name': u'transcript_amplification'}],
  u'strand': u'+',
  u'transcriptAnnotationFlags': [u'basic']},
 {u'biotype': u'unprocessed_pseudogene',
  u'ensemblGeneId': u'ENSG00000224435',
  u'ensemblTranscriptId': u'ENST00000426025',
  u'geneName': u'NF1P6',
  u'sequenceOntologyTerms': [{u'accession': u'SO:0001636',
    u'name': u'2KB_upstream_variant'}],
  u'strand': u'+',
  u'transcriptAnnotationFlags': [u'basic']},
 {u'sequenceOntologyTerms': [{u'accession': u'SO:0001566',
    u'name': u'regulatory_region_variant'}]}]
```

As with any other method offered by the client, additional query parameters can be passed to refine the annotation, e.g. *include*, *exclude, etc.* Available parameters can be found in the Swagger documentation: http://bioinfo.hpc.cam.ac.uk/cellbase/webservices

Note that a *normalize* parameter can be activated enabling, for example, to provide reference and alternate strings in a VCF-like format.

Using the R client (cellbaseR)

Using the RESTful web services

The best way to use of the RESTful Web Services is through the client libraries implemented for different programming languages. Nevertheless, under certain circumstances it may be required to directly access the RESTful API.

Both GET and POST annotation web services are available. These web services can be accessed by either building and accessing the appropriate URL or by using provided Java/Python/R/JavaScript clients.

The URL for accessing the GET web service can be built as: `http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v4/hsapiens/genomic/variant/<VARIANTLIST>/annotation`, `<VARIANTLIST>` containing a comma-separated list of the variants to query. For example, the URL for getting the annotation of variants

```
chr: 19 pos: 45411941 ref: T alt: C
chr: 14 pos: 38679764 ref: - alt: GATCTGAGAAGGGAAAAAGGG
```

querying current stable CellBase at the University of Cambridge would be: http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v4/hsapiens/genomic/variant/19:45411941:T:C,14:38679764:-:GATCTGAGAAGGGAAAAAGGG/annotation

POST queries can also be issued by using almost the same URL: `http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v3/hsapiens/genomic/variant/full_annotation` and providing the list of comma separated variants within the data entity.

- Using the command line interface (CLI):