Variant Annotation

Table of Contents:

Overview

As part of the enrichment step, some extra information can be added to the variants database as Annotations. This VariantAnnotation can be fetch from Cellbase or read from local file provided by the user. The model of the variant annotation is defined in the project Biodata, in variantAnnotation.avdl

Annotation steps

Unless otherwise specified, the annotation process is composed by two main steps, executed sequentially.

Annotation Creation

Reads variants from the database, and generates the annotation using the variant annotator. By default, this process will only annotate variants with no annotation, i.e. new variants. The number of new variants decreases as new files are being loaded to the database, so this step is expected to be shorter every time.

The creation generates an intermediate file with the annotation which will be loaded in the next step.

Annotation Load

The annotation load reads the generated file and writes the annotation into the database. After this step, the users can use the annotation filters when querying variants.

Annotators

Variant Storage Engine can make use of different annotators to produce the annotation for the variants.

The annotator can be modified at the annotating step, and the default value is defined in the *storage-configuration.yml* file.

- **annotator**
  Specify the annotator to use. Available annotators are: "*cellbase_rest*", "*cellbase_db_adaptor"* and "*other*".
  Previous to version v1.3.0, this parameter was named "*annotationSource*" instead of "annotator". See [#747].
  Default: cellbase_rest

CellBase Annotator

CellBase takes advantage of the data integrated to implement a rich and high-performance variant annotator. The variant annotation tool is integrated within the CellBase code and can be accessed in two different ways, via REST API, or connecting directly to the CellBase database. Mode information can be found at CellBase Variant Annotation.

CellBase REST Annotator

This is the default annotator for OpenCGA. This Annotator connects to a CellBase installation using the REST API.

This is an example of cellbase annotation using a REST call:

> http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v4/hsapiens/genomic/variant/19:44934489:G:A/annotation?exclude=expression *
>
> * Exclude the gene expression to reduce significantly the size of the annotation

CellBase Direct Annotator

The CellBase direct annotator creates a connection directly with the CellBase database. This requires an accessible installation of CellBase, which takes some resources, but it speeds up the annotation step. Specify *"annotator : cellbase_db_adaptor*" to use this annotator.

CellBase annotator configuration

The configuration of the CellBase annotator should be placed at the *storage-configuration.yml* file.

- **annotator.cellbase.exclude**
  List of fields to be excluded the variant annotation. Excluding some fields may reduce the size and speed up the annotation process.
  Default: expression
- **annotator.cellbase.use_cache**
  Use the CellBase internal cache.
  Default: true
- **annotator.cellbase.imprecise_variants**
  Specify if imprecise variants are supported by CellBase. This option only applies to CellBase REST annotator.
  Default: false

User defined annotator

Some installations of OpenCGA may require a special variant annotator, either to use a different annotator, or to aggregate data from multiple sources of data.

To use a user defined annotator, extend in java the class *VariantAnnotator* with the custom annotation rules, add the compiled jar to the classpath, and add the new annotator class name to the configuration

- **variant.annotator.classname**
  Class name of the user defined annotator. The annotator and its dependencies must be in the classpath of the application.

Store multiple versions of Variant Annotation

When analyzing data it is really important to be able to review the results.

No more than 3 or 4 times a year, there are new versions of the data contained in CellBase. New Ensembl version, new population frequencies, etc, change the generated annotation, which may change the result of the analysis. After determine which set of variants may be interesting for the study of a certain disease, we may want to know exactly the annotation that was used to select those variants.

Multiple versions of the variant annotation can be stored and queried in OpenCGA.

Implementation details at [#830]

Save current annotation

Create a named read-only copy of the current variant annotation from all the variants in the given project. This action should be executed, if required, every time that the annotator changes.

- **CLI command**
  ```
  variant annotation-save --project myProject --annotation-id v1
  ```

Delete annotation

Delete a certain annotation copy.

- **CLI command**
  ```
  variant annotation-delete --project myProject --annotation-id v1
  ```

Query specific annotation version

Get the variant annotation from a certain version, given a set of variants or a region.

- **REST endpoint**
  ```
  analysis/variant/annotation/query?annotationId=v1&id=1:1234:A:C
  analysis/variant/annotation/query?annotationId=CURRENT&region=1:1000-2000
  ```
- **CLI command**
  ```
  variant annotation-query --annotation-id v1 --id 1:1234:A:C
  variant annotation-query --annotation-id CURRENT --region 1:1000-2000
  ```

Variant Annotation Metadata

It is very important to know the exact version of the variant annotation. Any change may lead into differences in the downstream analysis.

In order to keep track of the existing variant annotation copies in the database, and their source data version, OpenCGA stores internally the version of the annotator and the version of the source data that the annotator uses.

Variant annotation metadata

```
{
  "species" : "hsapiens",
  "assembly" : "GRCh38",
  "release" : 4,
  "annotation" : {
    "current" : {
      "id" : 3,
      "name" : "CURRENT",
      "creationDate" : null,
      "annotator" : {  // https://bioinfo.hpc.cam.ac.uk/cellbase
/webservices/rest/v4/meta/about
        "name: ": "CellBase (OpenCB)",
        "gitCommit: ": "ab4e4dd83ff5b337392f33ae6bc7ba33be385807",
        "version: ": "4.5.3"
      } ,
      "sourceVersion" : [ { ... } ] // https://bioinfo.hpc.cam.ac.uk
/cellbase/webservices/rest/v4/meta/hsapiens/versions?assembly=grch38
    } ,
    "saved" : [
    {
      "id" : 1,
      "name" : "v1",
      "creationDate" : "2017-11-24T11:34:16.461+0000",
      "annotator" : {
        "name: ": "CellBase (OpenCB)",
        "gitCommit: ": "b118caadb557a77f66bbeb4e81261288b11174f7",
        "version: ": "4.5.2"
      } ,
      "sourceVersion" : [ { ... } ]
    } , {
      "id" : 2,
      "name" : "v2",
      "creationDate" : "2018-04-02T16:10:44.194+0000",
      "annotator" : {
        "name: ": "CellBase (OpenCB)",
        "gitCommit: ": "ab4e4dd83ff5b337392f33ae6bc7ba33be385807",
        "version: ": "4.5.3"
      } ,
      "sourceVersion" : [ { ... } ]
    } ]
  }
}
```

Read variant annotation metadata

Annotation metadata can be read via REST API and command line. By default, read all the annotations.

- **REST endpoint**
  `analysis/variant/annotation/metadata`
  `analysis/variant/annotation/metadata?annotationId=v1`
- **CLI command**
  `variant annotation-metadata`
  `variant annotation-metadata --annotation-id v1`

Custom annotation

The VariantAnnotation model includes a field for adding extra annotation attributes. This field is intended to contain custom annotation provided by the end user

Data model

Additional attributes can be grouped by source. Each source will contain a set of key-value attributes creating this structure:

Result

```
VariantAnnotation = {
  // ...
  "additionalAttributes" : {
    "<source1>" : {
      "attribute" : {
        "<key1>":"<value>",
        "<key2>":"<value>",
        "<key3>":"<value>"
      }
    },
    "<source2>" : {
      "attribute" : {
        "<key1>":"<value>",
        "<key2>":"<value>",
        "<key3>":"<value>"
      }
    }
  }
```

Example with multiple sources: In case of having custom annotations from more than one source, more than one source will appear in the additionalAttributes field:

Result

```
VariantAnnotation = {
  // ...
  "additionalAttributes" : {
    "myVcf" : {
      "attribute" : {
        "FEATURE":"specific",
        "SCORE":"300",
        "STRAND":"+"
      }
    },
    "myBed" : {
      "attribute" : {
        "name":"Pos1",
        "score":"353",
        "strand":"+"
      }
    }
  }
}
```

Input formats

OpenCGA Storage is able to load this custom annotation reading from three different formats: **GFF**, **BED** and **VCF**. When loading the new annotation data, the user has to provide a name for the new custom annotation. Due to the differences of these file formats, the information loaded won't be the same.

GFF and BED files describe features within a region, providing a chromosome, start and end. All the variants between the start and end will be annotated with the information.

GFF

From this file format, only the third column, containing the feature is extracted and loaded with the key "feature"

This line of GFF will generate the next additionalAttributes:

GFF

```
chr22 TeleGene enhancer 16053659 16063659 500 + . touch1
```

Result

```
VariantAnnotation = {
  // ...
  "additionalAttributes" : {
    "myGff" : {
      "attribute" : {
        "feature" : "enhancer"
      }
    }
  }
}
```

BED

From the bed format, columns name (4th), score (5th) and strand (6th) will be loaded.

This line of BED will generate the next additionalAttributes:

BED

```
chr22 16053659 16063659 Pos1 353 + 127471196 127472363 255,0,0 0 A A
```

Result

```
VariantAnnotation = {
  // ...
  "additionalAttributes" : {
    "myBed" : {
      "attribute" : {
        "name":"Pos1",
        "score":"353",
        "strand":"+"
      }
    }
  }
}
```

VCF

This format is not region based, so each line will modify a single variant. All the INFO column will be loaded as additional attributes.

The next VCF will generate the next additionalAttributes:

VCF

```
##fileformat=VCFv4.2
##FILTER=<ID=PASS,Description="All filters passed">
##INFO=<ID=FEATURE,Number=1,Type=String,Description="Feature type">
##INFO=<ID=SCORE,Number=1,Type=Integer,Description="Score value">
##INFO=<ID=STRAND,Number=1,Type=Integer,Description="Strand">
#CHROM POS    ID REF    ALT    QUAL   FILTER INFO
chr22 16050075 A G . 100 PASS FEATURE=specific;SCORE=300;STRAND=+
```

Result

```
VariantAnnotation = {
  // ...
  "additionalAttributes" : {
    "myVcf" : {
      "attribute" : {
        "FEATURE":"specific",
        "SCORE":"300",
        "STRAND":"+"
      }
    }
  }
}
```