pyCGA Documentation

Release 0.1.0

Antonio Rueda, Mathew Parker, Razvan Sultana

September 04, 2015

CONTENTS

1 Getting Started **3**
- 1.1 Installation . . . 3
- 1.2 Quick-start . . . 3
- 1.3 Id Considerations in OpenCGA . . . 4

2 API Documentation **5**
- 2.1 Web Services API . . . 5
- 2.2 How To Use API from python . . . 5

3 CLI Documentation **7**
- 3.1 PyCGA . . . 7
- 3.2 pyCGAVariantFetcher . . . 7

4 Developer Documentation **11**
- 4.1 How to implement new web services parsers . . . 11

OpenCGA is an open-source project that aims to provide a Big Data storage engine and analysis framework for genomic scale data analysis of hundreds of terabytes or even petabytes. For users, its main features will include uploading and downloading files to a repository, storing their information in a generic way (non-dependant of the original file-format) and retrieving this information efficiently.

The aim of this python package is provide an interface using the OpenCGA web services from python, and make easy and comprehensive the use of OpenCGA. And also provide a CLI usable from

The code is open source, and available on github.

The main documentation for the project is organized into a couple sections:

- *Getting Started*
- *API Documentation*
- *CLI Documentation*
- *Developer Documentation*

CHAPTER

ONE

GETTING STARTED

1.1 Installation

Here is a step by step guide on how to install pyCGA. It will get you the library ready to be load from a python script and all the scripts we provide installed in your system.

First, obtain Python if you do not already have it. Then you will ned to install pip

Once you have these you will be able to install the requirements

1.1.1 Requirements

Packages	Version
requests	2.7

Although all the packages, The dependencies are specified in the file pyCGA/pip_requirements.txt You can install them using:

```
cd pyCGA
pip install -r pip_requirements.txt
```

1.1.2 Install

Next step to complete the installation is execute the file setup.py, you can use:

```
sudo python setup.py install
```

or:

```
sudo pip install .
```

Note: If you are using pip and you would like to reinstall the packages you should use:

```
sudo pip install . --upgrade
```

1.2 Quick-start

inf

1.3 Id Considerations in OpenCGA

CHAPTER

TWO

API DOCUMENTATION

2.1 Web Services API

inf

2.2 How To Use API from python

inf

CHAPTER

THREE

CLI DOCUMENTATION

3.1 PyCGA

inf

3.2 pyCGAVariantFetcher

3.2.1 CommandLine Options

This program can be used to fetch variant easily from openCGA

```
usage: pyCGAVariantFetcher [-h] --host OpenCGA Host --studyID study Id [--ids ids] [--regio
                           [--type type] [--reference reference] [--alternate alternate] [-
                           [--missingAlleles missingAlleles] [--missingGenotypes missingGen
                           [--genotype genotype] [--consequence_type annot-ct] [--xref xref
                           [--conservation conservation] [--alternate_frequency alternate_f
                           [--limit reference_frequency] [--skip reference_frequency] [--sk
                           [--unknownGenotype unknownGenotype] [--returnedSamples returnedS
                           [--batchSize batchSize] [--outputType output]
                           Session Id
```

Positional arguments:

sid	Required. To get this session ID yoy need to login using pyCGA

Options:

--host	Required. This is the url to web service host. (i,e http://XX.X.XX.XXX:8080/)
--studyID	Required. The id of the study, this can be query usgin pyCGA
--ids	Optional. Select by variant ids (i,e dbSNP ids). Use comma to separate the ids
--region	Optional. Select by region (i,e chr:start-end). Use comma to separate the region
--chromosome	Optional. Select by chromosome. Use comma to separate the region
--gene	Optional. Select by gene name (It can be ENSMBL Ids or HGNCs symbols. Use comma to separate the gene names

--type Optional. Select by type of variant [SNV, MNV, INDEL, SV, CNV]. This filter should not be used alone due to the huge amount of results could be retrieved

--reference Optional. Select by reference base(s)

--alternate Optional. Select by alternate base(s)

--files Optional. Select by files, only the variants found in these file will be selected. Please note these are the id files in the DB, use pyCGA to get the id files. Use comma to separate the file ids

--maf Optional. Select by minor allele frequency. In this filter only the samples in the db are consider, not any external population. Syntax: [<|>|<=|>=]{number}, (e.g, >=0.05)

--mgf Optional. Select by minor genotype frequency. In this filter only the samples in the db are consider, not any external population. Syntax: [<|>|<=|>=]{number}, (e.g, >=0.1)

--missingAlleles Optional. Select by number of missing alleles in the whole sample set. In this filter only the samples in the db are consider, not any external population. Syntax: [<|>|<=|>=]{number}, (e.g, >=0.05)

--missingGenotypes Optional. Select by number of missing genotypes in the whole sample set. In this filter only the samples in the db are consider, not any external population. Syntax: [<|>|<=|>=]{number}, (e.g, >=0.05)

--annotationExists=False Optional. Select only the annotated variants

--annotationDoesNotExist=False Optional. Select only the variants without annotation

--genotype Optional. Select by sample genotype. Samples names must be specified as they are stored the db. Please, find more information in the documentation, Syntax: {samp_1}:{gt_1}(,{gt_n})*(;{samp_n}:{gt_1}(,{gt_n})*)* (e.g. HG0097:0/0;HG0098:0/1,1/1)

--consequence_type Optional. Select by consequence type. Consequence type SO term list. Use comma to separate the SO terms. Please, find the information of SO terms supported in the documentation. (e.g. SO:0000045,SO:0000046

--xref Optional. Select by XRef, this is a field used to map ids from different dbs. Please, find the information of dbs supported in the documentation.

--biotype Optional. Select by Biotype. Consequence type SO term list. Use comma to separate the biotypes. Please, find the information of biotype terms supported in the documentation. (e.g. protein_coding,retained_intron. This filter should not be used alone due to the huge amount of results could be retrieved

--polyphen Optional. Select by polyphen, polyphen score ranges from [0-1]. Syntax: [<|>|<=|>=]{number}, (e.g, >=0.9). This filter is slow if it is not used along others

--sift Optional. Select by sift, sift score ranges from [0-1]. Syntax: [<|>|<=|>=]{number}, (e.g, >=0.9). This filter is slow if it is not used along others

--conservation Optional. Select by conservation sources. Please read the documentation to find the sources available. Use comma to separate the conservation sources. Syntax: sourceName[<|>|<=|>=]number (e.g. phastCons>0.5,phylop<0.1). . This filter is slow if it is not used along others

--alternate_frequency Optional. Select by frequency of the alternate allele in one population. Please read the documentation to find the population available. Use comma to separate the population. Syntax: populationName[<|>|<=|>=]number (e.g. 1000g_CEU>0.5,1000g_AFR<0.1)

--reference_frequency Optional. Select by frequency of the reference allele in one population. Please read the documentation to find the population available. Use comma to separate the population. Syntax: populationName[<|>|<=|>=]number (e.g. 1000g_CEU>0.5,1000g_AFR<0.1)

--limit Optional. limit (number of results)

--skip Optional. skip (number of results)

--sort=False Optional. Sort the output by chromosome coordinates

--group_by Optional. Group the output by

--unknownGenotype=./. Optional. Returned genotype for unknown genotypes.

--returnedSamples Optional. Only the specified samples will be returned. The samples names in the db, if you have doubts about this, please read the documentation. Use comma to separate the sample names

--returnedFiles Optional. Only the information from the specified files will be returned.Please note these are the id files in the DB, use pyCGA to get the id files. Use comma to separate the files id

--batchSize=5000 Optional. This parameter control the size of the batches of variants per query, This number is proportional to the memory used and inversely proportional to the time. This value can not be greater than 5000.

--outputType=json Optional. This parameter control the size of the batches of variants per query, This number is proportional to the memory used and inversely proportional to the time. This value can not be greater than 5000.

Possible choices: json, VCF, AVRO-GA4GH, AVRO-OPENCGA

3.2.2 Examples

Get all variants in chromosome 1 with a maf in the database less than 0.05:

```
pyCGAVariantFetcher [sid] --host [hostname] --studyID [studyID] --chromosome 1 --maf < 0.05
```

Get all nonsense variants in a certain gene panel with a phylop conservation score more than 0.2:

```
panel=PAX6,B3GALTL,SOX2,MFRP,RAX,BCOR,OTX2,SIX6,BMP4,ALDH1A3,COL4A1,BMPR1A,HCCS,CYP1B1,RAB3GAP1,SHH,V
pyCGAVariantFetcher [sid] --host [hostname] --studyID [studyID] --consequenceType SO:1000062 --gene S
```

Get all variants in a certain region with an alternate frequency in AMR population for 1000 Genomes less than 0.1:

```
pyCGAVariantFetcher [sid] --host [hostname] --studyID [studyID] --region 1:1100000-1300000 --alternat
```

Note: If you have doubts of how to get and use the ids in OpenCGA, pleas read *Id Considerations in OpenCGA*

Note: Please notice this program is fetching variants from a DB which can contain millions of variants. Although the DB is indexed to obtain the best performance, several general queries are not supported to be fast. In this way, for example, if you try to fetch variants filtering only by type of variants (i,e SNVs), the result will be very slow, but if you query for a type of variant in a specific region will be very fast.

So we recommend, use at least one of these filters in your queries:

- ids
- chromosome
- region
- gene
- genotype
- consequenceType
- xref
- alternate_frequency
- reference_frequency
- maf
- mgf

CHAPTER

FOUR

DEVELOPER DOCUMENTATION

4.1 How to implement new web services parsers

inf