Name: pedigree_impala_analysis.py
Document Version: 1.0
Date: 27th November 2015
Author: Hrishikesh Lokhande (hlokhand@systemsbiology.org)

Description

The candidate script (pedigree_impala_analysis.py) is an annotation program that allows a user to analyze variants in a pedigree based upon a particular inheritance mode. The user can use different inheritance modes to run the script, control results by providing various thresholds in the user specified region (jupyter) or in the configuration file. The candidate script also annotates a variant with a stress score, which is obtained by processing the target inheritance pattern and the observed pattern w.r.t to an allele (Segregation class). The variant information is loaded into platform specific tables on the p7_platform database on impala. Current Annotation include: Kavair, CMS, UCSC_known_genes, CADD, Clinvar and SNPEff(depending upon availability on Impala).

Please follow along to get the usage information of the script.

User inputs:

Users can change or add inputs to the program by editing the user specified region of the program (Only applicable to the Jupyter version) or the configuration file.

Description of the options:

Stress and stress_cutoff:
Stress is the genetic stress calculated as hamming distance between the observed genotype vector and the target pattern. A stress of 0 represents that the observed and expected genotypes are the same.

Kaviar and kaviar_cutoff:
Kaviar database allows variant frequency lookups from the kaviar_isb table on the impala system. Since the goal of the candidate script is to analyze and annotate rare variants, we are currently using a cutoff of 3% or 0.03.

score_cutoff:
This is the overall candidate score that is calculated using dann_score, segregation score, allele frequency, max quality score etc. A detailed description of how the overall candidate score is calculated is given below.

subject_list:
This is the inclusion order of the pedigree members. It is important that the inheritance target pattern is designed considering the inclusion order.

target_patterns:
The user can specify one or more inheritance patterns to the script.
Eg: --target_patterns <aaabab>, <abaaab>

Paternal inheritance=abaaab
Maternal_inheritance=aaabab
Simple_recessive=ababbb
Denovo = aaaaab
(The above is true for a pedigree with inclusion order of Father, Mother and a child (NB), the target pattern and the inclusion order can be changed to analyze bigger pedigrees.)

outdir:
Results would be written in this directory.

user_database:
The script creates a temporary table using the inclusion order, this table is further used as a base table to write outer joins on all annotation tables. This table is dropped when the analysis is over.

Restrict:
This could be used to restrict the analysis to a particular chromosome. Please leave this option blank if you want to analyze all chromosomes.

Dependencies:

The candidate script can run both on all glados nodes and on different famgen servers, there are certain dependencies that are essential for the script to work.

Impyla:
Impyla Python package works as a communicator between python and impala it uses the standard protocol as the ODBC/JDBC drivers. Please refer to the follow https://blog.cloudera.com/blog/2014/04/a-new-python-client-for-impala/ to install implya on your system.

Python packages:
Future and collections package from http://python-future.org/quickstart.html and https://docs.python.org/2/library/collections.html respectively. This is essential as all the print functions are using this package.

Segregation class:
Segregation score is an important annotation of the script; please make sure that the segregation.py class is imported inside the main script. (from segregation import segregation)

Databases and tables:

Following are the names of the databases and their tables that the script uses to retrieve annotation and variant information from.

p7_ref_grch37:
This database houses all of the annotation tables. Kaviar, Clinvar,ucsc_known_gene,CADD,DANN,CMS etc.

p7_platform:
This databases houses variant information tables from different platforms, for current implementation we are using wgs_illumina_variant table to get variant information for all 103 families that were sequenced by Illumina.

User_database.temp:
This table would hold the variant information temporarily. Joins would be made on this table.

Overview of the script:

1. The script starts by creating a temporary table by querying the variant table (wgs_illumina_variant or wgs_comgen variant based upon the platform) using the inclusion order; this operation should take about 20-30 mins to complete.
2. Once the temporary table is created an outer join is run on annotation tables like Kaviar, CMS, UCSC_genes, CADD/DANN, Clinvar etc. This join is run separately for each chromosome as writing a join with all chromosomes is computationally expensive. The results from the join are then sorted by position. (Joining and processing results can take anywhere from 5-20 mins depending upon length of the chromosome and target patterns supplied to the script)
3. Next step involves re-creation of all the genotypes for every position and for all members. Missing genotypes (./.) are converted to homozygous reference calls for now.
4. Once all variants for every position are identified a segregation score is calculated for each variant based upon the inheritance pattern and the observed pattern.
5. Overall candidate score is then calculated for each variant using allele frequency, quality score, segregation score and DANN score.
6. Finally variants are printed out to an out file depending upon the users cutoff for allele frequency, segregation score and overall candidate score. Results for each inheritance mode are written in separate files.
7. Once all chromosomes are processed the script drops the temp table.

Overall candidate score:

The overall candidate score allows ranking of individual candidate variants. This score is calculated based on queried allele frequency, max quality score seen at a position, DANN score, segregation score and CMS. In the current implementation more emphasis is given to pathogenic DANN scores, for higher DANN scores the overall candidate score is adjusted to a lower value.

Details: (For reference please look at the function overall_candidate_score)

1. The queried allele frequency is modified using the **QAF_score_modifier** function, for a frequency value from 0 to 0.05 this function is likely to return values from 1 to 1.3.
2. DANN score for a variant is also modified using **dann_score_modifier** function, this function returns values 0.1,0.5,0.7 and 1 for a dann score of 0.995 -1, 0.98-0.995,0.93-0.98 and <0.93 respectively.
3. The max quality score is also modified using **quality_score_adjustment** function, this returns a 0 for a quality score of >50, returns a max of 1 and min 0 for a quality score >35 and returns a 4 for anything below 35. (This ensures that candidates with good quality score are retained)
4. Finally, the score is calculated as the product of all input parameters, a value of 1 is added to each parameter before the multiplication.
5. A candidate score of 0.1-0.5 usually indicates that a variant has a good quality score, has a low allele frequency and a higher pathogenic DANN score.

Usage: For jupyter version the user would have to edit the top "user assigned region" of the code.
To run the program from the terminal please type the following:

pedigree_impala_analysis.py --targets <comma sep targets> --config <impala.config> --outdir <ab>

Result:

The Table below provides brief interpretations of the result column headers:

Header	Example Result for 1 variant	Interpretation
Chromosome	11	Chromosomal location of variant
Position	2905353	Base pair position where the first base is present
Reference	T	Hg19 reference allele at this position
Candidate	C	Candidate variant observed at this position
Genotypes	TT TC TC	Genotypes of all members at this position
Genotype vectors	aaabab	Genotype vectors at this position w.r.t to Candidate allele
Inheritance mode stress	0	Indicates the origin of allele
Allele count	2	Count of the candidate allele in the pedigree
Maximum Sequence quality score	211	Maximum quality score for genotype
Minimum Sequence quality score	158	Minimum quality score for genotype
Average Sequence quality score	184	Average quality score for genotype
Kaviar	0.000026	Queried allele frequency (Frequency of C in this case)
DANN	0.9968	Pathogenic DANN score for the candidate variant
CMS	0	Occurrence in commonly mutation segment(0-No/1-Yes)
Clinvar_sig	5	Clinvar significance score for the candidate variant
Gene name	CDKN1C	Gene spanning this variant
Overall candidate score	0.20101	Candidate score or rank for this variant.