**Genome Annotator documentation**

1. Process overview

**Gene annotations**

Index gene annotations

Annotations exists?

NO

Download copy

Yes

**Genome**

Has local index?

NO

Download copy

yes

**Variations Input File**

Read and parse SNP line

Note: can be done in chunks for very large files

More SNPs ?

**SNP Validation:**

Validate SNP against genome and set flags

Note: this is a QA step

**SNP Effect Assignment :**

Assign SNP effect using gene annotations and the genome

Note: this is where we assign the SNP functional implication, amino acid and codon effect, …

NO

Done

2. Data sources description

**A. The genome**

The annotator takes the organism assembly version as one of the inputs. The program will then use the chromosome data associated with the specified organism version to validate/get the reference allele call. For coding SNPs (synonymous and non synonymous), the program also uses the chromosome data to get the codon sequence associated with the SNP. When used as getFlanking or getSequence, the annotator uses the chromosome data to extract the flanking sequence of SNPs, or to extract transcript sequence (exons, CDS, protein,..).

The annotator indexes the genome locally to generate a one-liner index for each chromosome where the chromosome file name has the format xx.dat - xx is the chromosome name (example X.dat, 1.dat). The program is being modified to accept the standard fasta format instead of the .dat although the .dat is the actual index format. The \*.dat format is generated by a script that takes the chromosome fasta file and converts it into a \*.dat index file. This script will be included with the annotator to create indexes locally when needed. In addition, a configuration file will allow the user to specify the following information regarding the genome:

1. a path to where chromosome data are stored locally

2. the chromosome file format - [mm]xx.fa where mm is the file prefix if any

The user can browse and download genomes from different public servers using some features of the annotator

**B. Variation input file**

The annotator takes the variation file as one of the arguments . Currently , the program expects the first line of the input file to be the header line with some program recognizable field names. As it is implemented, the program will work fine with variation files from the following sources:

i. any bed file -> SNP CHROMOSOME POSITION STRAND ALLELES

ii. any bed file -> SNP CHROMOSOME POSITION STRAND ALLELE ...

iii. any bed file -> RSID CHROMOSOME pos AlleleA AlleleB ...

iv. any bed ile -> ID CHROM pos REF ALT ...

The program will detect some variations in the field names. For example, chromosome field name can be on of chr,chrom, or chromosome. SNP position field name can be one of position,pos.bppos,bpposition,location,or loc ... The program has been modified to work with VCF file types. Instead of using the assumption that the first line of the SNPs file is always the header, the annotator uses a function that given a variation file , will return the header line.This implementation guarantees that files that include additional lines before the real header line are parsed correctly. The VCF has many lines in the header before the real header line. So as it implemented in previous versions, the annotator will not work properly on VCF files. The following is a brief example of a VCF file

http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-40

**C.Gene annotations**

The previous implementation of the annotator uses Ensembl database to annotate variations . The default Ensembl schema is the latest version of the specified organism assembly version. The program caches Ensembl gene annotations locally until one of these events occurs:

1. Ensembl has a new schema version for the user specified organism assembly version

2. The local copy modification date is more than 14 days old

The program has been modified to accept two more inputs:

i. the gene annotation file, and --> optional

ii. gene Prediction source/version --> optional

Gene annotation file is expected to have a header and be in gene Prediction format -(gtf format - work in progress). The program uses gene Prediction source/version input only when there is no annotation file specified. In that case, the program will extract the annotations from the specified public database server. The user can browse the list of available gene annotations for a given organism version using some features of the annotator

Note: Allowing the annotation file to become an input to the program forces us to require some additional input flags from the user. Given that the annotator was originally implemented to use Ensembl annotations, the default genomic coordinates is 1-base coordinates standard. But data from the user does not guarantee that assumption. The user should be able to specify the coordinates base of the annotations using the appropriate flag. If no flag is set then the default will be 1-base coordinate standard

3. Algorithm

The SNPs annotator validates and annotates single polymorphism variation using the gene annotations and the organism genome.

a. Data structures

- **indexing gene annotations**

 We Index gene annotations into memory using genomic region unique id. Unlike the previous implementation that relies on Ensembl local ids for transcripts and exons, the new version generates local transcript ids on the fly. Four structures index the annotations: featureIndexMap, featureList,featuresVec,and tabularData

***The featureIndexMap*** object (format:map <pair(chrom,start),<end,id>) stores ids generated for every unique genomic region. This data structure will speed up genomic position lookup using the map lower and upper bound functions on keys pair(chrom,start) .These ids will be used to build the transcript network featureList.

***The featureList*** object (format: map<id,featuresVec> stores a list of all the transcripts that share the same genomic region.

***featuresVec*** (format: vector<struct tabularData> is a vector of transripts where transcripts are structs of type tabularData.

***tabularData*** is a structure that stores transcript data parsed from the input annotations file. This structure has some easy data access functions associated with it.

**- input file data line**

Both the annotations and SNPs lines are parsed and stored into a struct data structure for easy data access. We use maps , combined with pairs, and vector to take advantage of build-in functions associated with these containers.

b. Design

**i. Current design**

    foreach SNP do:

     get list of genes that contain this SNP

     if geneList empty then: intergenic SNP, getNearestExon();display data

     else:

       foreach gene in geneList{ get transcriptlist of this gene that contain the SNP;

          foreach returned transcript :

              if SNP NOT on exon then: intronic SNP,getNearestExon();display data

               else: get the exon id/rank, compute SNP location

                  get codon,amino acid, position relative to CDS,if this SNP in coding region .

**ii. Design modifications**

   foreach SNP do:

     validate SNP and set flags

     display SNP AND GET next if user only wants validation

     else: get list of transcripts that contain this SNP

       if transcriptlist empty then: intergenic SNP, getNearestExon();display data

       else: foreach transcript in transcriptlist

            if SNP NOT on exon then: intronic SNP,getNearestExon();display data

            else:

                get the exon id/rank, compute SNP location

                get codon,amino acid, position relative to CDS,if this SNP in coding region

                display data

**4. Program Usage**

You can run the program as:

i. SNP Annotator

ii. SNP Validator

ii. getSNPFlanking sequence

ii. getTranscriptSequence

**I) - Running Genome Annotator as SNP Annotator**

Scenario 1: you have a downloaded/generated gene annotations file [annotationsfilename]

a. you have downloaded a gene annotations file (genePrediction format - 0-base starts and 1-base ends) for mm9

b. you have a SNPs file [snpfilename] (bed, vcf,..) for mm9

c. you have the genome for mm9

Program usage:

cmd: ./genomeAnnotator -v mm9 -f snpfilename -A annotationsfilename

If you want to include the codon and amino acid implication then run the annotator as

cmd: ./genomeAnnotator -v mm9 -f snpfilename -A annotationsfilename --coding

If coordinate starts in the genePrediction file follow 1-base standard then run the annotator as follow:

cmd: ./genomeAnnotator -v mm9 -f snpfilename -A annotationsfilename --coding –oneBaseStart

Scenario 2: You do not have a gene annotations file but you want to annotate your SNPs using a remote server: Ensembl, UCSC,Graber,MGI

a. you have a SNPs file for mm9

b. you have the genome for mm9 for example

c. but you do not have gene annotations file for mm9

Program usage:

i. First step check which gene annotations are available for mm9 on the remote server by running the annotator as follow:

cmd: ./genomeAnnotator -v mm9 --annotList

The above command will display all the gene annotations we currently have for mm9 in our integrated database

cmd: ./genomeAnnotator -v mm9 --annotList --host ucsc

The above command will display all the gene annotations UCSC currently has for mm9

ii. copy one of the gene annotation name say mm9-ensGene

cmd: ./genomeAnnotator -v mm9 -a mm9-ensGene -f snpfilename -F resultFineName

for the simple version of the annotator.If you want to include the codon and

amino acid implication then run the annotator as follow:

cmd: ./genomeAnnotator -v mm9 -a mm9-ensGene -f snpfilename --coding -F resultFineName

If you want to use UCSC annotations, for example refGene then run as follow:

cmd: ./genomeAnnotator -v mm9 -a refGene -f snpfilename -F resultFineName --host ucsc

If you want to include the codon and amino acid implication then run the annotator as follow:

cmd: ./genomeAnnotator -v mm9 -a refGene -f snpfilename --coding -F resultFineName --host ucsc

Scenario 3: You want the annotator to use/download the genome from a remote server

a. you have a SNPs file for mm9

b. you do not have local index for mm9 genome

c. you want the annotator to use Ensembl annotations for mm9

Program usage:

i. First step check which genomes are available :

a. on our server by running the annotator as follow:

cmd: ./genomeAnnotator -v mm9 --genomeList

OR

b. on UCSC server by running the annotator as follow:

cmd: ./genomeAnnotator -v mm9 --genomeList --host ucsc

ii. copy one of the Assembly Versions say mm9

then run the annotator as follow: ./genomeAnnotator -v mm9 --host ucsc --download

The program will download and index mm9 genome into the genomes directory under the program working directory (annotator main directory).

When you have index the genome locally, you can now run the annotator to annotate your SNPs file.

**II) - Running Genome Annotator as SNP Validator**

The scenarios are similar to the SNP Annotator's with the exception that the validator only uses the genome to run QA tests. We do not use gene annotations.

This program will validate all single position SNPs (novel and existing) from an input against the specified reference genome assembly build.

The program will also check for reference allele ,[and SNP allele ]consistency and sets the flags accordingly. The program Also tags CpG Sites, and Mutation types (transition or transversion).

cmd: ./cpp/genomeAnnotator -v mm9 -f snpfilename --validate

The result include Conflict,prog\_refAllele/prog\_concensusAllele,Is\_CpG,MutationType, FieldCountMatch, OrganismAssemblyVersion, in addition to the original fields

**III) - Running Genome Annotator to extract SNPs flanking sequences**

Gets the flanking sequence upstream and downstream from the specified position

and returns a sequence s such that s= upstream\_seq\_offset\_len -pos-downstream\_seq\_offset\_len

cmd: ./cpp/genomeAnnotator -v mm9 -f snpfilename --flanking -o 50

WHERE -o stands for the offset (in this case it is 50 base pairs up and down the SNP)

If the offset is not specified, the program returns a sequence that includes one base pair upstream and downstream of the SNP. The result include Flanking Coordinates,Flanking Sequence,OrganismAssemblyVersion, in addition to the original fields

**III) - Running Genome Annotator to extract Transcripts fasta sequences**

Generates fasta sequences of transcripts from the specified gene annotation source

Cmd: ./cpp/genomeSNPValidator -v mm9 -a mm9-ensGene --sequence -T cds

Where –T specifies the sequence type – in this case CDS sequences will be returned.

**Notes**

- mm9 is used in the examples for simplicity

- All gene annotations files should be in Bed/genePrediction format (<http://genome.ucsc.edu/FAQ/FAQformat.html#format9>)

- By default the annotator assumes SNP location coordinates in the SNPs file to follow 1-base standard

- By default the annotator assumes annotation coordinates in the genePrediction file to follow 0-base starts and 1-base ends standard

- By default if no input SNP file is provided, the annotator will be reading SNPs lines from standard input

- By default if no output file is provided, the annotator will be writing results lines to standard output , if you want the result to be stored in an output file, add the option -F outputfilename

- By default if no organism version is provided, the annotator will use the current version of mouse (we can have user set default organism)

- If coordinate starts in the genePrediction file follow 1-base standard, add the --oneBaseStart flag to the program command line

- If SNP location coordinates in the SNPs file to follow 0-base standard, add the --snpZeroBaseStart flag to the program command line

- The annotator talks to the web server via web-services.