**Health Canada *de novo* mutation identification pipelines**

**Required software**

* python 2.x
* picard
* bwa
* GATK 4.0.11 (primarily used)
* GATK 3.8 (needed for CatVariants tool in RVQS stage only)
* samtools
* bcftools
* denovogear
* vcf-subset

**Important files**

* settings.py

This file is among the most important for the running of the pipeline. To allow for ease of use, many of the variables needed throughout the pipeline are set here. When setting up the pipeline for the first time in a new environment this file will need to be edited. Generally speaking it contains a few set parameters used in the pipeline but most importantly it defines where a variety of files, directories and executables used by the pipeline are located. An important example of this is ‘indirroot’ which will be the root directory storing all of the intermediate and final results produced by the pipeline. Before starting, this directory should contain the following subdirectories:

* + 0\_fastq\_raw
  + 1\_ubam
  + 2\_align
  + 3\_mark\_dups
  + 4\_rbqs
  + 5\_dng
  + 5\_haplotypecaller
  + 6\_filtered\_DNG
  + 6\_GenomicsDBImport
  + 7\_jointgenotype
  + 8\_rvqs
  + 9\_genotyperefinement
  + 10\_filtered\_GATK
* functions.py

This file contains a few functions used by a number of different stages in the pipeline, such as functions to read PED files, check directories to ensure they exist, etc. Unless you are overhauling the pipeline these functions likely don’t need to be changed.

**Pipeline overview**

To execute each stage of the pipeline you simply have to execute the relevant script and provide the family id. For example:

./1a\_fastq\_to\_ubam.py 1004\_21

Before starting, you need to organize your raw fastq files into directories for each family within the 0\_fastq\_raw directory described above. Each family will have a directory with its family ID. Inside this directory, each individual will have their own directory (using their ID as its name). Inside each individual’s directory will be their fastq files named <sample\_ID>\_R1.fastq.gz and <sample\_ID>\_R2.fastq.gz. So, for example, for family 2011\_22, the directory <indirroot>/0\_fastq\_raw/1004\_21/ should contain the following directories:

ls 2011\_22/

2011022 2011023 2011024 2011026

and those directories should contain:

ls 2011\_22/\*

2011\_22/2011022:

2011022\_R1.fastq.gz 2011022\_R2.fastq.gz

2011\_22/2011023:

2011023\_R1.fastq.gz 2011023\_R2.fastq.gz

2011\_22/2011024:

2011024\_R1.fastq.gz 2011024\_R2.fastq.gz

2011\_22/2011026:

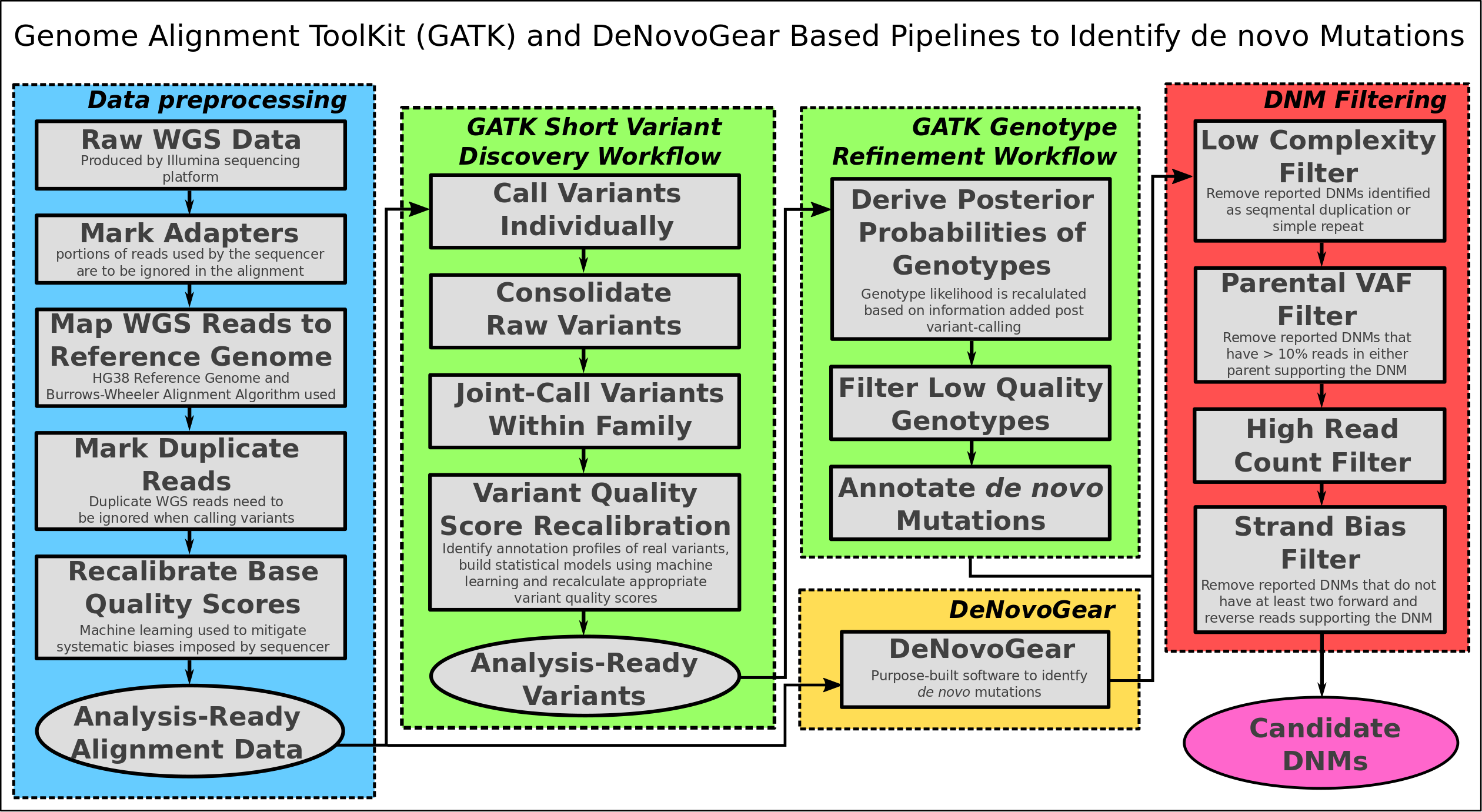
2011026\_R1.fastq.gz 2011026\_R2.fastq.gz

If these directories are properly set up then the pipeline should take care of the rest. Each stage of the pipeline will look into the appropriate directory for the input and place the output in the corresponding output directory. For nearly every file the pipeline generates it also generated a log file. The pipeline is designed so that the first line on each log file is the exact command run to create the file to which the log file corresponds. This is so that if the pipeline fails on a specific file, the entire stage does not need to be rerun but you can simply just execute the command from within the log file.

**Pipeline overview – stages**

In this section the various stages of the pipeline will be listed along with any notes made or any relevant links used while implementing it. Some of this text comes from the GATK site. The pipeline has 3 main parts:

1. Data preprocessing – based on GATK’s best practices, the data pre-processing for variant discovery workflow
2. DNM identification:
   1. DeNovoGear
   2. GATK based – GATK Short Variant Discovery Workflow followed by the GATK Genotype Refinement Workflow
3. DNM filtering – based on the filters applied in Raheleh’s “timing, rates, ...” paper:
   1. Low-complexity regions, which we defined as segmental duplications or simple repeats.
   2. Remove sites that had more than 10% of reads supporting the alternative allele in either of the parents.
   3. Removed sites whose depth was in the top 0.01% quantile in terms of read depth. For this, we assumed read depth to be Poisson distributed, with the λ parameter of the Poisson distribution equal to the mean read depth of the genome.
   4. Remove sites with less than 2 reads supporting the alternative allele on each strand.

****

**Before using the pipeline**

Need create BWA index on hg38 (already done, will need to redo if you use a different reference):

CMD: ./apps/bwa-0.7.17/bwa index hg38/Homo\_sapiens\_assembly38.fasta.gz

Real time: 6190.529 sec; CPU: 6138.804 sec

**Preprocessing Pipeline**

* Data pre-processing for variant discovery: <https://software.broadinstitute.org/gatk/best-practices/workflow?id=11165>
* This first processing step is performed per-read group and consists of mapping each individual read pair to the reference genome, which is a synthetic single-stranded representation of common genome sequence that is intended to provide a common coordinate framework for all genomic analysis. Because the mapping algorithm processes each read pair in isolation, this can be massively parallelized to increase throughput as desired.
* https://gatkforums.broadinstitute.org/gatk/discussion/6483/how-to-map-and-clean-up-short-read-sequence-data-efficiently
* The workflow reflects a lossless operating procedure that retains original sequencing read information within the final BAM file such that data is amenable to reversion and analysis by different means. These practices make scaling up and long-term storage efficient, as one needs only keep the final BAM file.

1. a) Convert fastq to uBAM (**./1a\_fastq\_to\_ubam.py <family\_ID>**)

* uses picard FastqToSam
* <https://gatkforums.broadinstitute.org/gatk/discussion/6484/how-to-generate-an-unmapped-bam-from-fastq-or-aligned-bam#optionA>
* needs a lot of temp space (hundreds of GBs)
* ~4.5-5 hours

b) Mark adapter sequences using MarkIlluminaAdapters (**./1b\_MarkIlluminaAdapters.py <family\_ID>**)

* uses picard MarkIlluminaAdapters
* MarkIlluminaAdapters adds the XT tag to a read record to mark the 5' start position of the specified adapter sequence and produces a metrics file. Some of the marked adapters come from concatenated adapters that randomly arise from the primordial soup that is a PCR reaction. Others represent read-through to 3' adapter ends of reads and arise from insert sizes that are shorter than the read length. In some instances read-though can affect the majority of reads in a sample, e.g. in Nextera library samples over-titrated with transposomes, and render these reads unmappable by certain aligners. Tools such as SamToFastq use the XT tag in various ways to effectively remove adapter sequence contribution to read alignment and alignment scoring metrics. Depending on your library preparation, insert size distribution and read length, expect varying amounts of such marked reads.
* not much RAM (~1.5 GB?: Runtime.totalMemory()=1512570880)
* ~4-4.5 hours (241.28, 242.94, 245.96, 267.29, 269.09 min)

1. Align uBAM files to ref genome (**./2\_align.py <family\_ID>**)

* [SamToFastq] | [BWA-MEM] | [MergeBamAlignment]
* CPU:
  + a few Java instances can use up to 600-1200%
  + bwa ranges between 200-800 (given 8 cores)
* little RAM (~2GB per individual)
* almost 34 hours (1856m57.738s, 2031m32.559s)
* SANITY CHECK: ensure # reads match between raw and aligned:
  + samtools view merged.bam | cut -f1 | sort | uniq | wc -l
  + samtools view raw.bam | cut -f1 | sort | uniq | wc -l

1. Mark duplicates (**./3\_mark\_duplicates.py <family\_ID>**)

* <https://gatkforums.broadinstitute.org/gatk/discussion/2799#latest>
* mainly serial, sometimes parallel
* ~26GB ram per process

1. Recalibrate base quality scores (**./4\_rbqs.py <family\_ID>**)

* <https://gatkforums.broadinstitute.org/gatk/discussion/11081/base-quality-score-recalibration-bqsr>
* detects systematic errors made by the sequencing machine when it estimates the accuracy of each base call.
* apply machine learning to model these errors empirically and adjust the quality scores accordingly
* eg. we can identify that, for a given run, whenever we called two A nucleotides in a row, the next base we called had a 1% higher rate of error. So any base call that comes after AA in a read should have its quality score reduced by 1%
* process involves two key steps:
  + the BaseRecalibrator tool builds a model of covariation based on the input data and a set of known variants, producing a recalibration file
  + the ApplyBQSR tool adjusts the base quality scores in the data based on the model, producing a new BAM file.
* running the tools: <https://gatkforums.broadinstitute.org/gatk/discussion/2801/howto-recalibrate-base-quality-scores-run-bqsr>
* hg38 files to use: <https://software.broadinstitute.org/gatk/documentation/article.php?id=1247>
  + The most recent dbSNP release (build ID > 132)
  + Mills\_and\_1000G\_gold\_standard.indels.b37.vcf
  + 1000G\_phase1.indels.b37.vcf (currently from the 1000 Genomes Phase I indel calls)

**DNM Identification - DeNovoGear**

1. DeNovoGear (**./5\_dng.py <family\_ID>**)

* dependencies:
  + Recent C++ compiler, supporting C++11 (eg. gcc 4.8.1+ or clang 3.3+)
  + CMake 3.1+ when compiling http://www.cmake.org/download/#latest (installed with yum)
  + HTSlib 1.2+ http://www.htslib.org/ (came with Samtools)
  + Eigen 3 http://eigen.tuxfamily.org/ (downloaded)
  + Boost 1.47+ http://www.boost.org/ (installed with yum)
* v1.1.1: https://github.com/denovogear/denovogear/releases - could not build
* development branch: https://github.com/denovogear/denovogear/archive/develop.tar.gz
* pedigree file: https://gatkforums.broadinstitute.org/gatk/discussion/7696/pedigree-ped-files
  + The PED file is a white-space (space or tab) delimited file. The first six columns are mandatory:
    - Family ID
    - Individual ID
    - Paternal ID
    - Maternal ID
    - Sex (1=male; 2=female; other=unknown)
    - Phenotype

**DNM Identification – GATK**

* What is the best way to find de novo mutations in trios <https://gatkforums.broadinstitute.org/gatk/discussion/10552/what-is-the-best-way-to-find-denovo-mutations-in-trios>
* Germline short variant discovery
  + <https://software.broadinstitute.org/gatk/best-practices/workflow?id=11145>
  + <https://gatkforums.broadinstitute.org/gatk/discussion/11145/germline-short-variant-discovery-snps-indels>

1. HaplotypeCaller (**./5\_haplotypecaller.py <family\_ID>**)

* Call germline SNPs and indels via local re-assembly of haplotypes
* <https://software.broadinstitute.org/gatk/documentation/tooldocs/current/org_broadinstitute_hellbender_tools_walkers_haplotypecaller_HaplotypeCaller.php>
* running Single-sample GVCF calling without allele-specific annotations: <https://gatkforums.broadinstitute.org/gatk/discussion/4639/x-chromosome-gentyping>

1. Consolidate GVCFs (**./6\_GenomicsDBImport.py <family\_ID>**)

* consolidating the contents of GVCF files across multiple samples in order to improve scalability and speed the next step, joint genotyping
* <https://software.broadinstitute.org/gatk/documentation/article?id=11813>
* <https://gatkforums.broadinstitute.org/gatk/discussion/10061/using-genomicsdbimport-to-consolidate-gvcfs-for-input-to-genotypegvcfs-in-gatk4>

1. Joint-Call Cohort (**./7\_jointgenotype.py <family\_ID>**)

* GenotypeGVCFs merges gVCF records that were produced as part of the Best Practices workflow for variant discovery. This tool performs the multi-sample joint aggregation step and merges the records together in a sophisticated manner: at each position of the input gVCFs, this tool will combine all spanning records, produce correct genotype likelihoods, re-genotype the newly merged record, and then re-annotate it.
* <https://software.broadinstitute.org/gatk/documentation/tooldocs/3.8-0/org_broadinstitute_gatk_tools_walkers_variantutils_GenotypeGVCFs.php>

1. Filter Variants by Variant (Quality Score) Recalibration (VariantRecalibrator, ApplyRecalibration) (**./8\_rvqs.py <family\_ID>)**

* uses machine learning to identify annotation profiles of variants that are likely to be real, and assigns a VQSLOD score to each variant that is much more reliable than the QUAL score calculated by the caller
* <https://software.broadinstitute.org/gatk/documentation/tooldocs/3.8-0/org_broadinstitute_gatk_tools_CatVariants.php>
* <https://gatkforums.broadinstitute.org/gatk/discussion/2805/howto-recalibrate-variant-quality-scores-run-vqsr>
* <https://gatkforums.broadinstitute.org/gatk/discussion/1259/which-training-sets-arguments-should-i-use-for-running-vqsr>
* <https://software.broadinstitute.org/gatk/documentation/article.php?id=1259>
* <https://gatkforums.broadinstitute.org/gatk/discussion/11084/variant-quality-score-recalibration-vqsr>
* <https://software.broadinstitute.org/gatk/documentation/article?id=39>
* <https://gatkforums.broadinstitute.org/gatk/discussion/3952/variantrecalibrator-no-data-found>
  + added '--maxGaussians 4' to SNP VR to avoid 'no data' error

1. Genotype Refinement Workflow (**./9\_genotyperefinement.py <family\_ID>**)

* <https://gatkforums.broadinstitute.org/gatk/discussion/comment/52571/#Comment_52571>
* <https://gatkforums.broadinstitute.org/gatk/discussion/4723/genotype-refinement-workflow>
* <https://gatkforums.broadinstitute.org/gatk/discussion/4727/howto-run-the-genotype-refinement-workflow>
* <https://software.broadinstitute.org/gatk/documentation/article?id=11074>
* CalculateGenotypePosteriors only works on biallelic sites - use SelectVariants
* to filter 1000G\_phase1.snps.high\_confidence.hg38.vcf.gz
* <https://gatkforums.broadinstitute.org/gatk/discussion/5071/calculategenotypeposteriors-error>

**DNM Filtering**

1. and 10. filtering results (**./6\_filter\_dng.py <family\_ID> and ./10\_filter\_GATK.py <family\_ID>**)

* exactly the same as Raheleh’s “Timing, rates,...” paper
* Low complexity regions
  + Simple repeats: <http://genome.ucsc.edu/cgi-bin/hgTables?hgsid=678905581_zUZiKkQeJLMCUGv6Fy2grdaW4gMG&boolshad.hgta_printCustomTrackHeaders=0&hgta_ctName=tb_simpleRepeat&hgta_ctDesc=table+browser+query+on+simpleRepeat&hgta_ctVis=pack&hgta_ctUrl=&fbQual=whole&fbUpBases=200&fbDownBases=200&hgta_doGetBed=get+BED>
  + Segmental dups: <http://genome.ucsc.edu/cgi-bin/hgTables?hgsid=678905581_zUZiKkQeJLMCUGv6Fy2grdaW4gMG&boolshad.hgta_printCustomTrackHeaders=0&hgta_ctName=tb_simpleRepeat&hgta_ctDesc=table+browser+query+on+simpleRepeat&hgta_ctVis=pack&hgta_ctUrl=&fbQual=whole&fbUpBases=200&fbDownBases=200&hgta_doGetBed=get+BED>
* Calculating mean read depth:
  + Run picard CollectAlignmentSummaryMetrics on each bam file (takes roughly 2 hours). Extract PF\_ALIGNED\_BASES for both forward and reverse reads. Divide this by number of bases in hg38 (3209286105, total size (including N bases) taken from: <http://genomewiki.ucsc.edu/index.php/Hg38_27-way_Genome_size_statistics>)
* This step uses a data structure called an interval tree. The implementation given assumes these trees have been created, but has code to create the trees if necessary. The calls to these functions have been commented out.