ctDNA pypeliner

Circulating tumour DNA (ctDNA) is fragmented DNA derived from tumour cells found in blood plasma. These DNA fragments contain the same mutations found in the tumour and can be detected using DNA sequencing technologies. The ability to find mutations in ctDNA allows for liquid biopsies to be performed in order to track tumour progression throughout treatment. Since a liquid biopsy only requires a blood draw, it is preferable to more invasive procedures. The caveat of monitoring through liquid biopsies is that ctDNA makes up only a small percentage of cell-free DNA (cfDNA), which is all the DNA found circulating in the bloodstream, resulting in very low frequency mutations.

ctDNA pypeliner is a workflow designed to call somatic single nucleotide variants (SNVs) and insertion/deletion mutations (INDELs) by using five other publicly available SNV/INDEL detection tools, MutationSeq (https://github.com/shahcompbio/mutationseq), DeepSNV (https://bioconductor.org/packages/release/bioc/html/deepSNV.html), LoLoPicker (https://github.com/jcarrotzhang/LoLoPicker), Strelka2 (https://github.com/Illumina/strelka), and VarScan (http://varscan.sourceforge.net/). Since ctDNA SNVs are low frequency, these mutations are difficult to detect and differentiate from errors that may occur during the sequencing process. To reduce the number of false positives, ctDNA pypeliner uses the individual results from each mutation detection tool and performs a consensus amongst those results to form a single list of higher probability mutations. To minimize the effects of sequencing errors, the workflow requires targeted, high coverage sequencing (~1000x coverage).

Dependencies

ctDNA pypeliner makes use of Conda to manage source packages. The tools that are used by the workflow are mostly available as Conda packages. The setup procedure of ctDNA pypeliner and the Conda environment is outlined here:
Clone ctDNA_pypeliner:

```
git clone https://github.com/shahcompbio/ctDNA_pypeliner.git
```

```
cd ctDNA_pypeliner
```

Then create an environment with the required packages:

```
conda create --name ctDNApypeliner --file conda_packages.txt
```

Activate the environment:

```
source activate ctDNApypeliner
```

ctDNA_pypeliner uses LoLoPicker and Annovar which are not available through conda. To setup LoLoPicker:

```
git clone https://github.com/jcarrotzhang/LoLoPicker.git
cd LoLoPicker
python setup.py install
```

To setup Annovar, follow the download procedures here: http://annovar.openbioinformatics.org/en/latest/user-guide/download/

Add the ctDNA pipeline into the current site packages:

```
python setup.py install
```

The ctDNA pypeliner code is written in Python and uses Pypeliner to manage the workflow.

Input

ctDNA pypeliner takes in two main input files. The first is the main input file that contains paths to unaligned matched normal and tumour/plasma sample fastQ files along with other metadata about the samples. This input file is in YAML format and an example is shown here:

```
PATIENT_ID:
  normal:
    NORMAL_SAMPLE_ID:
      fastq1: /path/to/fastqfile/normal_sample_L001_R1_001.fastq.gz
      fastq2: /path/to/fastqfile/normal_sample_L001_R2_001.fastq.gz
      run: Run002
      sample_status: Normal
      type: saliva
  tumour:
    TUMOUR_SAMPLE1_ID:
      fastq1: /path/to/fastqfile/tumour_sample1_L001_R1_001.fastq.gz
      fastq2: /path/to/fastqfile/tumour_sample1_L001_R2_001.fastq.gz
      run: Run007
      sample_status: Treatment naive Primary Sx
      type: TNBC FFPE
    TUMOUR_SAMPLE3_ID:
      fastq1: /path/to/fastqfile/tumour_sample3_L001_R1_001.fastq.gz
      fastq2: /path/to/fastqfile/tumour_sample3_L001_R2_001.fastq.gz
      run: Run002
      sample_status: Pre-Surgery/Post-Chemo
      type: plasma baseline
```

This exact format is required for the workflow to differentiate tumour/plasma samples from normal tissue samples to tell somatic mutations from germline mutations. The workflow will output results in directories in the structure of `PATIENT_ID/SAMPLE_ID`

The second input file that is required is a configuration file also in YAML format. This file contains various directory paths that are required by the workflow. A template is shown here:

```
reference_genome: '/path/to/reference/GRCh37-lite.fa'
results_dir: '/path/to/results/'
bam_directory: '/path/to/bams/'
bed_file: '/path/to/beds/GRCh37.bed'
r_script_dir: '/path/to/ctDNA_pypeliner/ctDNA/r_scripts/'
museq_python:
'/shahlab/pipelines/virtual_environments/museq_pipeline/bin/python'
museq_classify: '/path/to/mutationSeq_4.3.7_python2.7/classify.py'
museq_deep_model: '/path/to/mutationSeq_4.3.7_python2.7/model_deep_v0.2.npz'
museq_config: '/path/to/mutationSeq_4.3.7_python2.7/metadata.config'
snv_vcf_template: '/path/to/ctDNA_pypeliner/template_snv.vcf'
indel_vcf_template: '/path/to/ctDNA_pypeliner/template_indel.vcf'
annovar: '/path/to/annovar/'
annovar_humandb: '/path/to/annovar/humandb/'
```

The Workflow

Alignment

The first step of the workflow is to align the sequencing reads from fastQ to sorted BAM files. Since alignment of reads does not depend on the type of sample, all samples (normal and tumour/plasma) can be aligned in parallel. From there, if a patient contains more than one normal sample (for example, one sample from normal tissue and one from buffy coat), the normal sample BAMs are merged into a single BAM file.

Variant Calling

Each variant calling tool takes in matched tumour and normal BAM files as input along with other parameters (manifest (BED) file, reference genome, etc.). The tools operate independently of each other and can be run in parallel. The parameters (such as minimum read depth, maximum normal variant allele frequency, minimum tumour allele frequency, base quality, mapping quality) of each tool are tuned where appropriate to be less stringent to allow for low frequency mutations to be called. The outputs of each tool are either in VCF format or tsv format. These output files can be found in the appropriate result directory for the patient and sample.

Consensus

After variant calling is complete by each tool, a union is performed on each sample using the tool outputs. Each mutation called by each tool contains a scoring

metric. For LoLoPicker, DeepSNV, and VarScan this is a p-value while Strelka uses its own quality score (QSS) out of a maximum 3070 and MutationSeq uses a probability value out of 1. For a mutation to appear in the workflow output file, it must be called by at least two out of five tools and must also have an associated score that passes a threshold. The threshold for p-value scores is 0.0005, MutationSeq score threshold is 0.65, and Strelka score threshold is 200. These thresholds can be adjusted during execution of the workflow. Raw read counts of mutations that pass the consensus filters are then obtained from the aligned BAM file and only those with read counts above 1000 (also adjustable) are kept. These mutations are written into output TSV files that contain the scoring metric from the tools that called the mutation.

Annotation

Finally, the resulting mutations are annotated using ANNOVAR (refGene and cytoBand). The annotations describe the affected gene and protein changes caused by the mutation. Annotated mutations are written in TSV format and VCF format.

Output files

Output files are created in the `results_dir` specified in the configuration file and are organized into subdirectories according to patient_id. For each patient tumour sample, ctDNA_pypeliner will output 3 files for SNVs and 3 files for INDELS. One TSV, one TXT, and one VCF for each type of mutation. The TSV and VCF output files contain scoring information from each respective tool. Those scores are specified as follows:

- DeepSNV: The corrected p-value
- LoLoPicker: The corrected p-value
- MutationSeq: The probability score of mutation
- Varscan: The somatic p-value for Somatic/LOH events
- Strelka: Quality score for any somatic snv

Only VarScan and Strelka call INDEL mutations, so indel output files will only contain VarScan and Strelka scoring

Tab-Separated Values (TSV) file

The output TSV file clearly states the score factors of each SNV calling tool, read depth, and allele counts/frequencies at each mutation site. This output is unannotated.

Below is a sample header from an SNV TSV file:

```
chr  pos ref alt count deepSNV LoLoPicker  VarScan MutationSeq Strelka
N_coverage  N_A N_C N_G N_T N_N N_vaf T_coverage  T_A T_C T_G T_T T_N T_vaf
```

INDEL TSV file:

```
chr pos ref alt VarScan Strelka N_coverage  N_ref N_alt N_vaf T_coverage  T_ref
T_alt T_vaf
```

TXT file

The output TXT file contains annotation information of each mutation site. It does not contain scoring information from the tools, nor does it contain information about read depth and allele count/frequency.

Below is a sample header from an output TXT file:

```
Chr Start End Ref Alt Func.refGene  Gene.refGene  GeneDetail.refGene
ExonicFunc.refGene  AAChange.refGene  cytoBand
```

VCF file

The output VCF file contains both annotation information as well as the information found in the TSV output file (scoring information, read depth, allele counts and frequencies)

Below is a sample meta information (header) from an output VCF file:

```
##fileformat=VCFv4.1
##source=ctDNA_pypeliner
##INFO=<ID=COUNT,Number=1,Type=Integer,Description="Number of consensus calls">
##INFO=<ID=DSNV,Number=1,Type=Float,Description="p-value from DeepSNV">
##INFO=<ID=LLP,Number=1,Type=Float,Description="p-value from LoLoPicker">
##INFO=<ID=VS,Number=1,Type=Float,Description="p-value from VarScan">
##INFO=<ID=MS,Number=1,Type=Float,Description="Probability score from
MutationSeq">
##INFO=<ID=STR,Number=1,Type=Float,Description="Quality Score from Strelka">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=AU,Number=1,Type=Integer,Description="Read count of nucleotide A">
##FORMAT=<ID=CU,Number=1,Type=Integer,Description="Read count of nucleotide C">
##FORMAT=<ID=GU,Number=1,Type=Integer,Description="Read count of nucleotide G">
##FORMAT=<ID=TU,Number=1,Type=Integer,Description="Read count of nucleotide T">
##FORMAT=<ID=NU,Number=1,Type=Integer,Description="Read count of null">
##FORMAT=<ID=VAF,Number=1,Type=Float,Description="Variant allele frequency">
```

```
##INFO=<ID=ANNOVAR_DATE,Number=1,Type=String,Description="Flag the start of
ANNOVAR annotation for one alternative allele">
##INFO=<ID=Func.refGene,Number=.,Type=String,Description="Func.refGene
annotation provided by ANNOVAR">
##INFO=<ID=Gene.refGene,Number=.,Type=String,Description="Gene.refGene
annotation provided by ANNOVAR">
##INFO=<ID=GeneDetail.refGene,Number=.,Type=String,Description="GeneDetail.refG
ene annotation provided by ANNOVAR">
##INFO=<ID=ExonicFunc.refGene,Number=.,Type=String,Description="ExonicFunc.refG
ene annotation provided by ANNOVAR">
##INFO=<ID=AAChange.refGene,Number=.,Type=String,Description="AAChange.refGene
annotation provided by ANNOVAR">
##INFO=<ID=cytoBand,Number=.,Type=String,Description="cytoBand annotation
provided by ANNOVAR">
##INFO=<ID=ALLELE_END,Number=0,Type=Flag,Description="Flag the end of ANNOVAR
annotation for one alternative allele">
#CHROM  POS ID  REF ALT QUAL  FILTER  INFO  FORMAT  NORMAL  TUMOR
```

Execution

Running the workflow

Locally

```
ctdna_pypeliner --input_yaml /path/to/input.yaml --config /path/to/config.yaml
--map_q 25 --base_q 15 --p_cutoff 0.0005 --T_vaf_cutoff 0.004 --N_vaf_cutoff 0
--museq_cutoff 0.65 --tmpdir /path/to/tmp/ --pipelinedir /path/to/pipeline/
--submit local --maxjobs 4
```

On shahlab cluster

```
ctdna_pypeliner --input_yaml /path/to/input.yaml --config /path/to/config.yaml
--map_q 25 --base_q 15 --p_cutoff 0.0005 --T_vaf_cutoff 0.004 --N_vaf_cutoff 0
--museq_cutoff 0.65 --tmpdir /path/to/tmp/ --pipelinedir /path/to/pipeline/
--submit asyncqsub --nativespec ' -V -hard -q shahlab.q -l h_vmem={mem}G -P
shahlab_high -S /bin/bash' --maxjobs 128 --context_config /path/to/context.yaml
```

Options

```
 --input_yaml          Input filename
  --config              Configuration filename
  --map_q               Minimum mapping quality
  --base_q              Minimum base quality
  --p_threshold         Maximum p_value
```

```
  --museq_threshold     Minimum MutationSeq score
  --N_vaf_threshold     Maximum normal variant allele frequency
  --T_vaf_threshold     Minimum tumour variant allele frequency
  --umi_trim            Set flag to true to trim trailing and leading UMI from
reads
```

Pypeliner workflow arguments

```
 --tmpdir TMPDIR       location of temporary files
  --pipelinedir PIPELINEDIR
                        location of pipeline files
  --loglevel {CRITICAL,ERROR,WARNING,INFO,DEBUG}
                        logging level for console messages
  --submit SUBMIT       job submission system
  --submit_config SUBMIT_CONFIG
                        job submission system config file
  --nativespec NATIVESPEC
                        qsub native specification
  --storage STORAGE     file storage system
  --storage_config STORAGE_CONFIG
                        file storage system config file
  --maxjobs MAXJOBS     maximum number of parallel jobs
  --repopulate          recreate all temporaries
  --rerun               rerun the pipeline
  --nocleanup           do not automatically clean up temporaries
  --interactive         run in interactive mode
  --sentinel_only       no timestamp checks, sentinal only
  --context_config CONTEXT_CONFIG
                        container registry credentials and job context
                        overrides
```