Pipeliner v 0.2.0

Evaluating the performance of bioinformatics' pipelines for Next Generation re-Sequencing

Bruno Nevado and Miguel Perez-Enciso

November 22, 2013

Contents

1 Introduction

Current deluge of Next Generation Sequencing (NGS) data has been accompanied by an increasing number of software and pipelines with a variety of options that are not necessarily well documented and whose actual performance or influence in the whole process is not well known. Pipeliner is a highly customisable but friendly GUI environment which allows comparing the performance of alternative NGS software, as well as exploring the influence of the choice of parameters in analysis of NGS datasets.

Pipeliner is written in C++ (www.cplusplus.com) and is a tool to compare alternative experimental designs and bioinformatics' pipelines for NGS re-sequencing studies. Pipeliner consists of two separate executables: a Graphical User Interface used for setting up analysis' parameters and plotting results; and a command-line tool used during the analysis to convert file formats, compute results, etc.

As the aim is to evaluate the performance of alternative pipelines for NGS analysis, an analysis with Pipeliner will typically require several other software packages to simulate the input datasets and analyse the NGS data.

The main steps involved in an analysis with Pipeliner are:

1. Defining the study system and experimental design. Making use of (coalescent or individual-based) simulations, the user can specify, e.g., how many individuals to sample, the evolutionary distance from the population sampled to the reference genome available, the nucleotide diversity or recombination rate of the population, patterns of selection or demographic changes, sub structuring, etc.
2. Simulating the NGS experiment, i.e. generating the NGS short reads from each individual in the population, according to a specific NGS technology and predefined average depth per individual.
3. Aligning the NGS short reads to the reference genome.
4. Performing variant calling and quality control.
5. Estimating a number of statistics (recovery, accuracy, false discovery rate, etc) summarising the performance of the pipeline defined.
6. Estimating population genetic parameters (theta, pi, neutrality tests) from the simulated NGS datasets.

All steps are easily customisable by the user, in terms of which software and which options to use. Further, Pipeliner allows the user to easily plot and compare the results obtained using different pipelines or different values for a range of parameters involved in the analysis.

If you use Pipeliner in your research results, please cite our paper:

Nevado B. and Perez-Enciso M. (*in preparation*). Pipeliner: evaluating NGS methods.

2 Obtaining and installing Pipeliner

2.1 Pipeliner

Pipeliner can be obtained from it's website. The source code is available for download, and pre-compiled executables are available for some versions of Mac and Linux OS. For the command-line tool, a C++11 compiler is needed, while for the GUI the QT libraries must also be installed (www.qt-project.org).

Instructions on how to set up Pipeliner in Linux and Mac OS machines are provided together with the program.

2.2 Additional software

Pipeliner uses different software to both simulate and analyse NGS datasets. By default, Pipeliner uses the following software:

1. **ms** to simulate the population of interest and the reference using the coalescent (website).
2. **art_illumina** to simulate NGS reads (website).
3. **bwa** to align the short reads (website).
4. **samtools** for variant calling, including the utilities bcftools and vcfutils.pl (website).
5. **picard-tools**, specifically the tools AddOrReplaceReadGroups and CreateSequenceDictionary (website).
6. **mstatspop** to calculate variability and neutrality tests, optional (website).
7. **R** to plot the results (website).

These programs must be installed before doing the analysis in Pipeliner. Furthermore, any of the steps involved in the analysis can be performed with alternative software (see *User-defined steps section*).

If these programs are installed outside the computer's path, the location of the executables needs to be specified in the *Run settings* section of PipelinerGUI.

2.3 OSes

Pipeliner is available for Linux and Mac computers.

3 Pipeliner GUI

Upon starting PipelinerGUI, the user is presented with four tabs:

1. Pipeline
2. Run settings
3. Bash file
4. Plots

The *Pipeline* tab is used to set up the analysis, i.e. defining the sampling design, study system, and bioinformatics' analysis. The *Run settings* tab is used to define paths, number of replicates, and other run-time options. The *Bash file* allows previewing, editing and saving the pipeline's commands into a file. This file can then be executed, locally or remotely (e.g. on a compute cluster) and once finished, the *Plots* tab can be used for summarising and plotting of results.

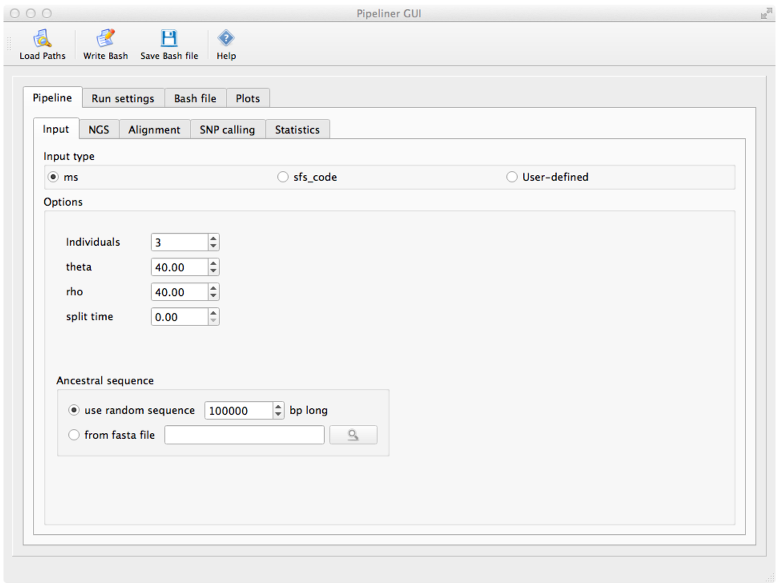
3.1 Pipeline

Used to define the both the input sequences to use in the analysis, the bioinfromatics' pipeline, and which statistics to calculate.

3.1.1 Input

In the input step, the user defines how to obtain the DNA sequences for analysis. These include two haplotypes per diploid individual and one haplotype to be used as reference genome. Three input modes are available to achieve this: ms coalescent simulation, sfs_code forward simulation, and User-defined commands.

- ms coalescent simulation: The user chooses settings to simulate the population of interest using a subset of options from the program ms.



Input type

[✓ ms] sfs_code User-defined

Options

Individuals	3
theta	40.00
rho	40.00
split time	0.00

Ancestral sequence

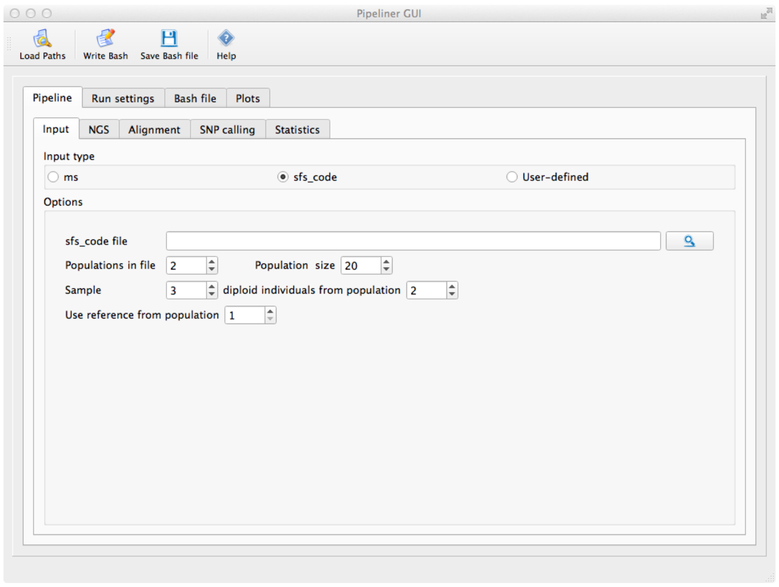
[✓ use random sequence] 100000 bp long

from fasta file

Options available:

- Number of individuals sampled.
- Variability and recombination rates.
- Length of the simulated region.
- Time of divergence between the population of interest and the reference.
- Ancestral sequence upon which the SNPs obtained with coalescent simulations will be applied (provided by the user, or obtained by setting the length and choosing the random sequence option, which will generate a sequence with uniform number and distribution of the four bases a, c, g, t).

• SFS_code: The user provides an output file from the forward simulator program sfs_code (website). Note that, **if simulating more than 1 population, all populations must have the same number of individuals sampled**.



Input type

ms [✓ sfs_code] User-defined

Options

sfs_code file

Populations in file 2 Population size 20

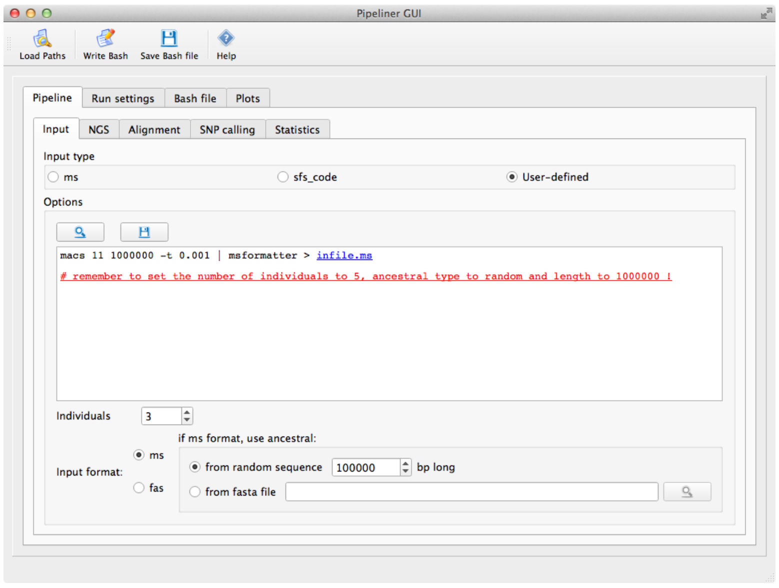
Sample 3 diploid individuals from population 2

Use reference from population 1

Options available:

- Location of sfs_code output file.
- Number of populations simulated.
- Number of diploids simulated per population (must be the same for every population).
- Number of diploid individuals to use in the analysis.
- Which population to analyse. The n diploid individuals will be taken from this populations, starting with the second diploid individual in the population (the first diploid individual is *reserved* for using as reference).
- Which population to use as the reference. Pipeliner will take the first sequence from this population and use it as reference genome for alignment, SNP calling etc...

- User-defined: Other software can be used to obtain the input data, by providing a command line specifying how to simulate it. This command line must be executable directly, i.e. it should work if it is copied and pasted into a terminal window. It should work with any simulation software insofar as the result is in either fasta or ms-like format. Whichever population scenario the user selects, **the first haplotype in the input file will be used as the reference**, and the remaining haplotypes represent the population, with **each 2 consecutive haplotypes treated as 1 diploid individual**. If using ms-like input, the user must also define which ancestral sequence to use - random or user provided. *See the User-defined steps section for details.*



Input type: ms [✓ User-defined] sfs_code

Options

```
macs 11 1000000 -t 0.001 | msformatter > infile.ms
# remember to set the number of individuals to 5, ancestral type to random and length to 1000000 !
```

Individuals: 3

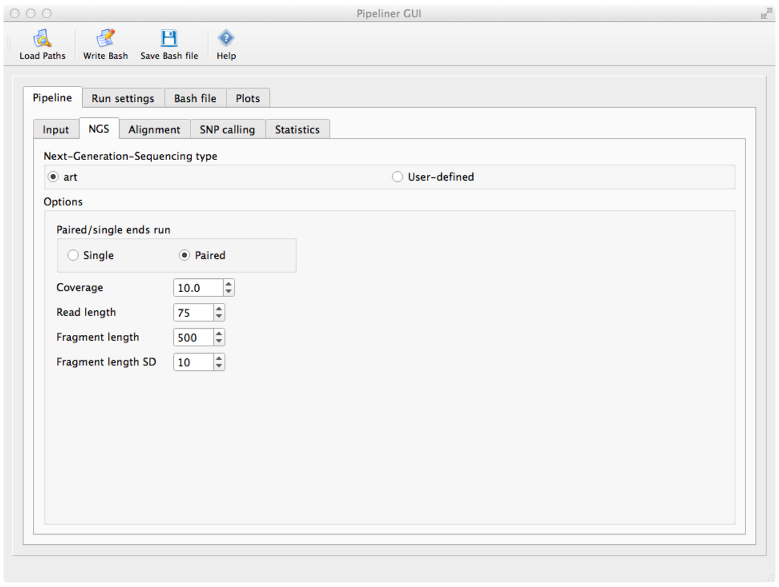
Input format: [✓ ms] fas

if ms format, use ancestral: [✓ from random sequence] 100000 bp long; from fasta file

3.1.2 NGS simulation

Used to obtain the NGS short reads for each diploid individual analysed. At the end of this step, there should be one (or two, if simulating paired-end sequencing) fasta-quality files per individual. Two options currently available: default art_illumina and User-defined.

- Default art_illumina: Use the program ART with Illumina technology and the built-in empirical profiles.



Next-Generation-Sequencing type: [✓ art] User-defined

Options

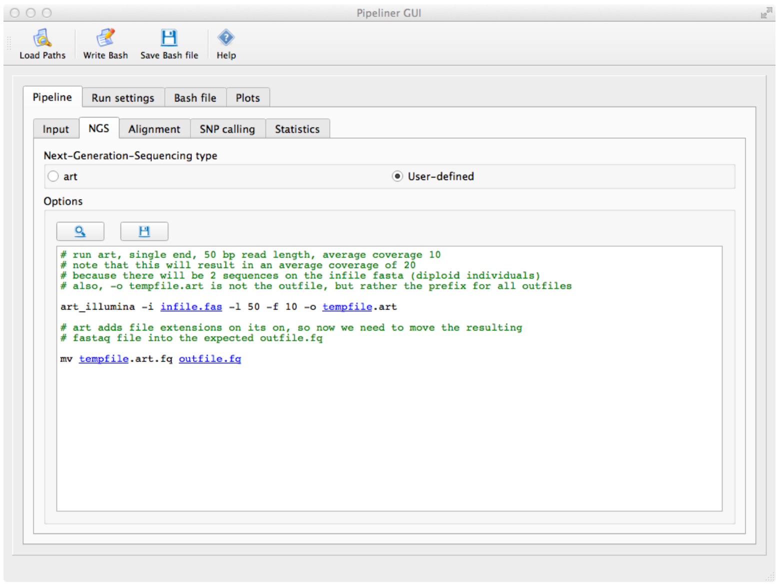
Paired/single ends run: Single [✓ Paired]

Coverage: 10.0
Read length: 75
Fragment length: 500
Fragment length SD: 10

Options available:

  - Single-end or paired-end run.
  - Average coverage to use per diploid individual.
  - Length of the short reads.
  - Length of the fragments sequenced from both ends, and the standard deviation of this value (SD). Only for paired-end runs.

- User-defined: Allows using alternative software to obtain the NGS short reads for each individual. This can be done by using alternative software (e.g. simseq or FluxSimulator) or by using 454 or solid profiles from ART (note that the correct binaries for ART-solid and ART-454 will need to be installed on the computer). *See the User-defined steps section for details.*



Next-Generation-Sequencing type

art [✓ User-defined]

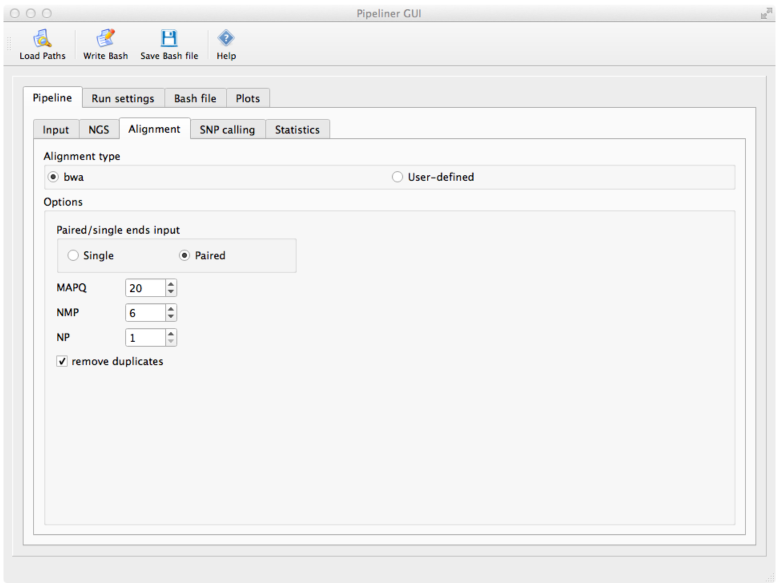
Options

```
# run art, single end, 50 bp read length, average coverage 10
# note that this will result in an average coverage of 20
# because there will be 2 sequences on the infile fasta (diploid individuals)
# also, -o tempfile.art is not the outfile, but rather the prefix for all outfiles

art_illumina -i infile.fas -l 50 -f 10 -o tempfile.art

# art adds file extensions on its on, so now we need to move the resulting
# fastaq file into the expected outfile.fq

mv tempfile.art.fq outfile.fq
```

3.1.3 Alignment

Used to specify how to align the NGS short reads from each individual to the reference sequence. This step should start from the fasta-quality files obtained before, and finish with a single bam formatted file (bam/sam file specifications) for each individual. Two options available: default using bwa, or User-defined step.

- Default bwa: Uses the software BWA to perform the alignment of short reads.



Alignment type

(●) bwa ○ User-defined

Options

Paired/single ends input

○ Single (●) Paired

MAPQ 20

NMP 6

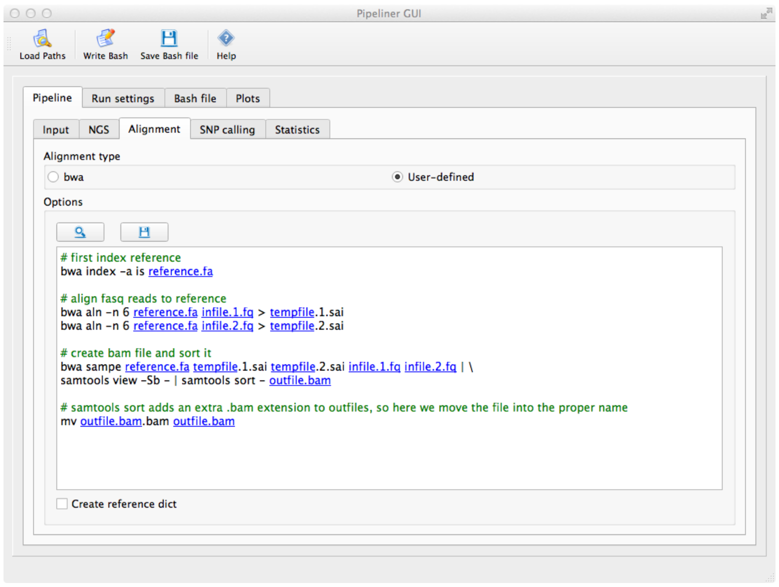
NP 1

☑ remove duplicates

Options available:

- Input type -single or paired-ends. Should match the output of the NGS simulation step.
- Mapping quality threshold - reads with mapping quality below this value will be discarded.
- Number of mismatches allowed between each read and the reference.
- How many processors to use while doing the alignment.

• User-defined: Use alternative alignment software, or use options from BWA that are not available in the default section. *See the User-defined steps section for details.*



Alignment type

bwa [✓ User-defined]

Options

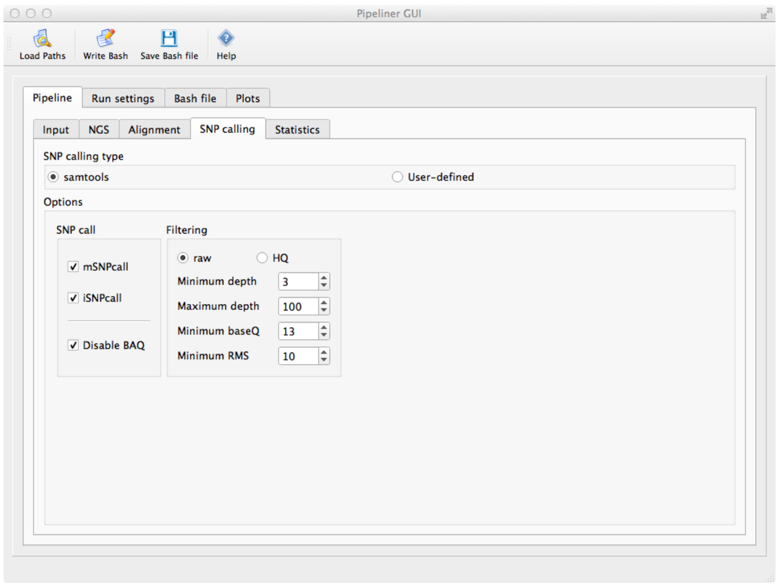
```
# first index reference
bwa index -a is reference.fa

# align fasq reads to reference
bwa aln -n 6 reference.fa infile.1.fq > tempfile.1.sai
bwa aln -n 6 reference.fa infile.2.fq > tempfile.2.sai

# create bam file and sort it
bwa sampe reference.fa tempfile.1.sai tempfile.2.sai infile.1.fq infile.2.fq | \
samtools view -Sb - | samtools sort - outfile.bam

# samtools sort adds an extra .bam extension to outfiles, so here we move the file into the proper name
mv outfile.bam.bam outfile.bam
```

☐ Create reference dict

3.1.4 SNP calling

Used to define SNP calling and filtering options. This step starts from the aligned short-reads obtained before (.bam files) and finishes with a .vcf format file (vcf file specifications). SNP calling can be done separately for each individual (iSNPcall) as well as jointly for all individuals under analysis (multiple-sample SNP calling, mSNPcall). Two options available: default using samtools, or User-defined step.

- Default samtools: Use the software samtools and bcftools to perform the SNP calling and filtering.



SNP calling type: [✓ samtools] User-defined

Options

SNP call:
- ✓ mSNPcall
- ✓ iSNPcall
- ✓ Disable BAQ

Filtering: [✓ raw] HQ

Minimum depth	3
Maximum depth	100
Minimum baseQ	13
Minimum RMS	10

Options available:

– SNP calling using each individual's data separately, and/or using all individual's data at the same time (multi-sample SNP calling)

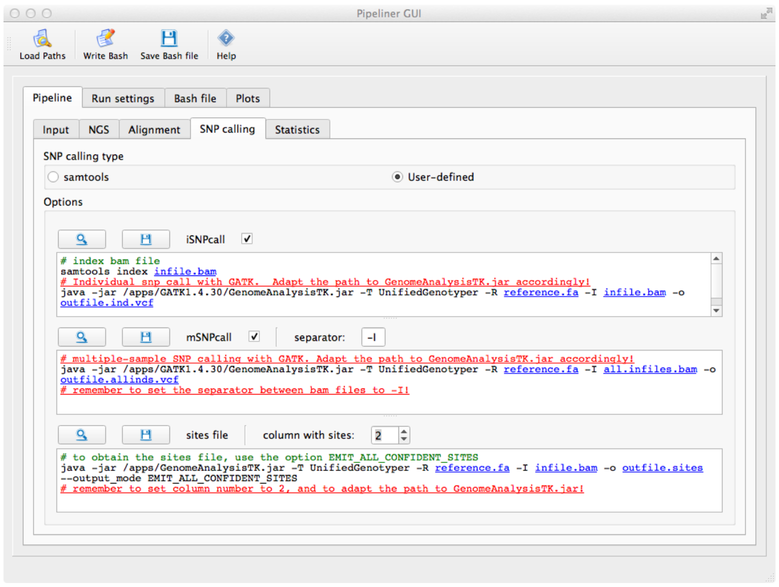
– Disable Base Alignment Quality (BAQ) option of samtools - leaving BAQ enabled should be useful in removing false positive SNP calls nearby indels.

– Minimum and maximum coverage to call SNPs (i.e. sites with less or more than these values will be excluded from the results).

– Minimum base quality of reads (reads below this threshold are not used for SNP calling).

– Minimum RMS mapping quality for SNPs (for filtering SNP calls using vfcutils.pl varFilter).

- User-defined: Define alternative SNP calling software, or perform SNP calling and filtering with extra options of samtools/bcftools not available in the default panel. *See the User-defined steps section for details.*



SNP calling type

○ samtools [✓ User-defined]

Options

iSNPcall ✓

```
# index bam file
samtools index infile.bam
# Individual snp call with GATK.  Adapt the path to GenomeAnalysisTK.jar accordingly!
java -jar /apps/GATK1.4.30/GenomeAnalysisTK.jar -T UnifiedGenotyper -R reference.fa -I infile.bam -o
outfile.ind.vcf
```

mSNPcall ✓ separator: -I

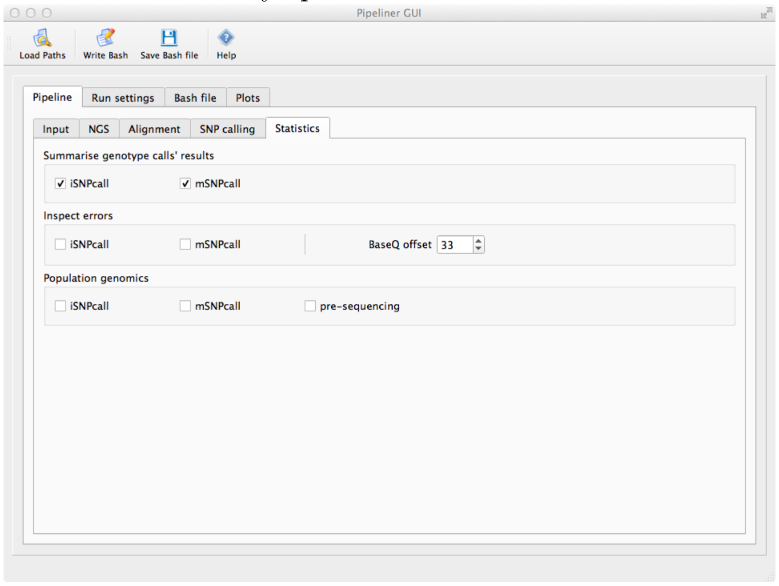
```
# multiple-sample SNP calling with GATK. Adapt the path to GenomeAnalysisTK.jar accordingly!
java -jar /apps/GATK1.4.30/GenomeAnalysisTK.jar -T UnifiedGenotyper -R reference.fa -I all.infiles.bam -o
outfile.allinds.vcf
# remember to set the separator between bam files to -I!
```

sites file column with sites: 2

```
# to obtain the sites file, use the option EMIT_ALL_CONFIDENT_SITES
java -jar /apps/GenomeAnalysisTK.jar -T UnifiedGenotyper -R reference.fa -I infile.bam -o outfile.sites
--output_mode EMIT_ALL_CONFIDENT_SITES
# remember to set column number to 2, and to adapt the path to GenomeAnalysisTK.jar!
```

3.1.5 Statistics

In this section the user defines how to summarise the performance of the pipeline defined. Three main summary options are available:



Summarise genotype calls' results

✓ iSNPcall ✓ mSNPcall

Inspect errors

☐ iSNPcall ☐ mSNPcall BaseQ offset 33

Population genomics

☐ iSNPcall ☐ mSNPcall ☐ pre-sequencing

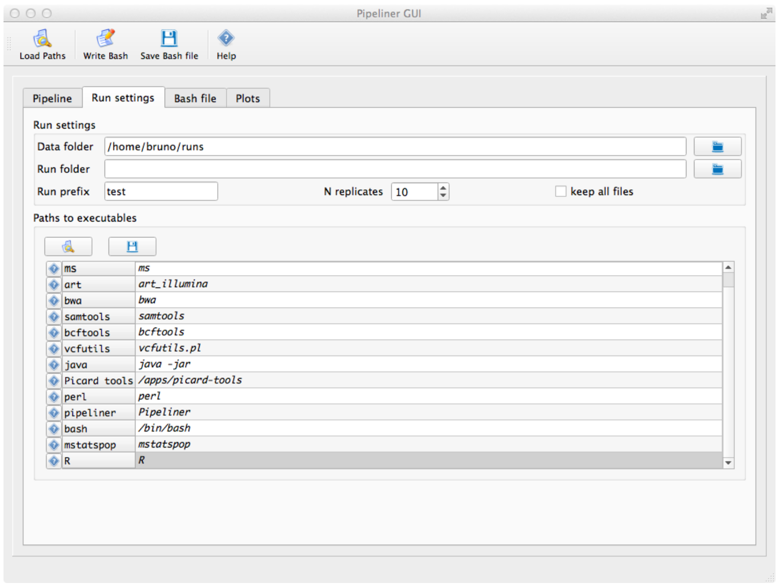
- Summarise genotype calls' results. Enabling this option will report, for each individual of each replicate, a number of statics concerning the power and accuracy of the pipeline defined. The statistics reported include **Recovery** - percentage of original sites that are correctly identified in the pos-sequencing datasets; **Accuracy** - percentage of sites that pass the different filters, i.e. are included in the list of confident sites, and are correctly identified in the pos-sequencing datasets; and **False Discovery Rate** - percentage of genotype calls performed that are incorrect. These statistics are reported for each individual analysed, and are reported separately for invariable positions - where the individual is homozygous for the reference allele - homozygous SNPs - where the individual is homozygous for the alternative allele - and heterozygous SNPs - where the individual carries one reference and one alternative allele. In addition, Pipeliner can report the frequency of different outcomes, e.g. the percentage of heterozygous SNPs that are incorrectly identified as homozygous for the reference allele or homozygous for the alternative allele, etc. *Enabling this option will create a summary file - or two if both iSNPcall and mSNPcall are enabled - that can be loaded into PipelinerGUI's Plot section after the analysis is finished.*

- Inspect errors. If enabled, Pipeliner will report, for each site that is incorrectly identified in the pos-sequencing datasets, the error type - homozygous SNP called heterozygous, heterozygous SNP called invariable etc.. - the original configuration i.e. the alleles present in reference and individual, and the mpileup line. The mpileup line shows the reference allele, the alternative allele, the bases seen covering the site and their base qualities, and can be used to pinpoint specific errors incurred by the pipeline defined. *This option is only available when running the analysis in a Linux machine, and the results must be checked manually using a text editor.*

- Population genomics. If selected, Pipeliner will calculate the following population genetics' statistics: population variability (theta), nucleotide variability (pi), Tajima's D, Fu and Li's D and Fay and Wu's H. Neutrality tests are calculated using the reference genome as outgroup. Statistics are calculated using the program mstatspop, and can be estimated for each SNP calling method defined in the pipeline - iSNPcall and mSNPcall - as well as with the original, pre-sequencing datasets i.e. without errors nor missing data. *Results will be written into a summary file that can be loaded into the Plot section of PipelinerGUI, together with the results of genotype calls.*

3.2 Run settings

The Run settings tab is used to specify several run-time options.



Run settings

Data folder: /home/bruno/runs

Run folder:

Run prefix: test

N replicates: 10

☐ keep all files

Paths to executables

ms	*ms*
art	*art_illumina*
bwa	*bwa*
samtools	*samtools*
bcftools	*bcftools*
vcfutils	*vcfutils.pl*
java	*java -jar*
Picard tools	*/apps/picard-tools*
perl	*perl*
pipeliner	*Pipeliner*
bash	*/bin/bash*
mstatspop	*mstatspop*
R	*R*

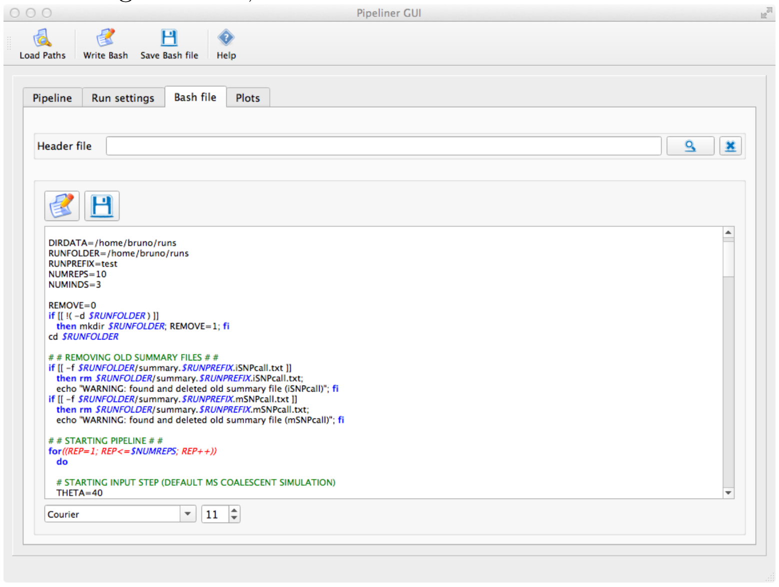
Options available:

- Data folder: where to store the result files. The folder defined here must exist prior to running the analysis.
- Run folder: where to run the analysis. Default: same as Data folder. Under some computer configurations, it might be convenient to run the analysis on a different folder (e.g. on the computer's temporary folder) and copy back the results to Data folder once the analysis finishes. If Run folder does not exist, it will be created by Pipeliner, and deleted once the analysis finishes.
- Run prefix: the prefix used for all files created during the analysis, except summary files. This is useful for cleaning up once the run finishes, or to manually inspect the resulting files from the analysis. For summary files, the run prefix is included in the middle of the file' name.
- Number of replicates: Number of times to run the entire pipeline before summarising the results.
- Keep all files. The default behaviour is to delete all files except summary and log files at the end of the run. If this option is checked, all files (fasta, fasta-quality, .bam files, .vcf files, as well as all temporary files) will be available in the Data folder once the run finishes.

In addition, the Run settings tab is also used to specify the location of the different executables needed to simulate and analyse the NGS datasets, including the simulation tools selected, alignment and SNP calling software, the mstatspop software and the command-line Pipeliner tool, etc. If these programs are installed on the computer's path, the absolute path is not needed, otherwise it must be defined in full. Once defined, the location of the different executables can be saved into a file, for quick setting up in subsequent analysis.

3.3 Bash file

Once the Pipeline and Run settings are defined, pressing the *Write Bash* button will write the commands into the Bash file section, where they can be previewed and edited, prior to saving into a file. This file can then be executed either locally or in a remote computer - exact way to run the commands in the file will depend on the computer system. For runs using a computer cluster, there is also the option to load a header file, which can contain e.g. which nodes to use, or how much memory to allocate to the job. The options will depend on the cluster where the analysis will be performed, its job-handling software, etc.



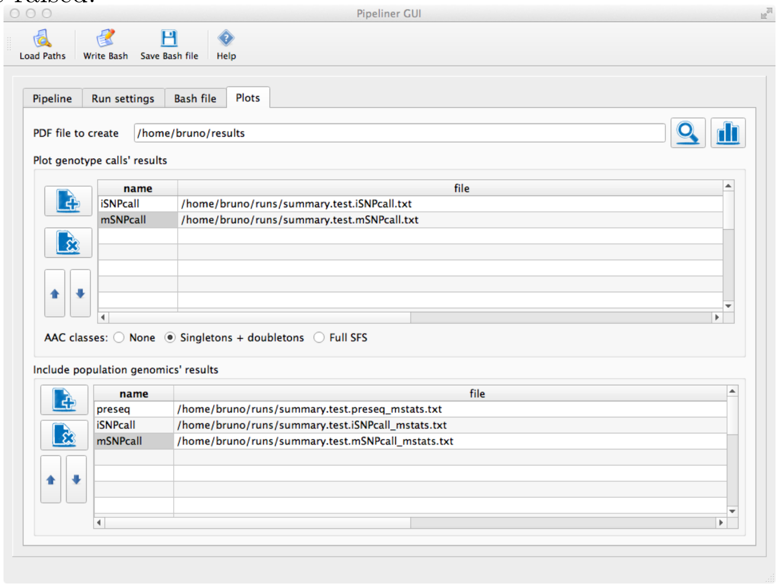
```
DIRDATA=/home/bruno/runs
RUNFOLDER=/home/bruno/runs
RUNPREFIX=test
NUMREPS=10
NUMINDS=3

REMOVE=0
if [[ !( -d $RUNFOLDER ) ]]
  then mkdir $RUNFOLDER; REMOVE=1; fi
cd $RUNFOLDER

# # REMOVING OLD SUMMARY FILES # #
if [[ -f $RUNFOLDER/summary.$RUNPREFIX.iSNPcall.txt ]]
  then rm $RUNFOLDER/summary.$RUNPREFIX.iSNPcall.txt;
  echo "WARNING: found and deleted old summary file (iSNPcall)"; fi
if [[ -f $RUNFOLDER/summary.$RUNPREFIX.mSNPcall.txt ]]
  then rm $RUNFOLDER/summary.$RUNPREFIX.mSNPcall.txt;
  echo "WARNING: found and deleted old summary file (mSNPcall)"; fi

# # STARTING PIPELINE # #
for((REP=1; REP<=$NUMREPS; REP++))
  do

  # STARTING INPUT STEP (DEFAULT MS COALESCENT SIMULATION)
  THETA=40
```

3.4 Plots

The Plots tab is used to plot and compare the output of runs with different programs/settings (though it can also be used to summarise the results of a single run).

Each analysis with Pipeliner will output one or several summary result files named summary.*runprefix*.(...).txt. The files ending in "...runprefix.iSNPcall.txt" and "...runprefix.mSNPcall.txt" contain the raw results of genotype calls, and can be loaded into the upper section of the Plots tab. The files ending in "...runprefix.iSNPcall_mstats.txt", "...runprefix.mSNPcall_mstats.txt" and "...runprefix.preseq_mstats.txt" contain the results of the population genetics' estimates, and can be loaded into the lower section of the Plots tab. For both cases, the name column, to the left of the files' names, can be used to label the resulting plots. If no names are entered, the results are labelled with numbers, according to the order in these sections.

Results concerning the Accuracy of genotype calls can also be summarised according to Alternative Allele Count, i.e. the number of alternative bases, for a given SNP, present in the population. AAC will affect Recovery and Accuracy values even in the case of individual sequencing, if the SNP calling step is performed jointly for all individuals. No difference should be found when doing SNP calling individually for each diploid individual. The three options for AAC plots are: *none*, i.e. do not create plots with AAC results; *singletons + doubletons*, will create plots with 3 classes - singletons of the alternative allele, doubletons for the alternative allele, and all remaining mutations; and *full Site-Frequency-Spectrum*, which will use one class for each possible AAC (singletons, doubletons ... up to fixed for alternative allele). Plots of AAC are only possible for analysis with the same number of diploid individuals sampled - otherwise an error will be raised.



PDF file to create: /home/bruno/results

Plot genotype calls' results

name	file
iSNPcall	/home/bruno/runs/summary.test.iSNPcall.txt
mSNPcall	/home/bruno/runs/summary.test.mSNPcall.txt

AAC classes: None [✓ Singletons + doubletons] Full SFS

Include population genomics' results

name	file
preseq	/home/bruno/runs/summary.test.preseq_mstats.txt
iSNPcall	/home/bruno/runs/summary.test.iSNPcall_mstats.txt
mSNPcall	/home/bruno/runs/summary.test.mSNPcall_mstats.txt

4 Pipeliner command line tools

Pipeliner's command line tools is a collection of commands to change file formats, summarise results, etc. To see a complete list of available commands and options, call Pipeliner's command line tool with the command "help".

5 A guided run

This short tutorial shows how to perform an example analysis in Pipeliner.

1. Download and install Pipeliner and default software (see section 2). Start PipelinerGUI by double-clicking the executable.

2. In the **Pipeline** > **Input** tab: Set *Input type* to ms, then choose 5 Individuals, and 100 *theta* and *rho*. Set the ancestral sequence type to random, and 100 000 bp long. Choose *Split time* of 0.

3. In the **Pipeline** > **NGS** tab: Set the run to *Paired-ends*, *coverage* to 6x and *read length* to 75 bp.

4. In the **Pipeline** > **Alignment** tab: Set the input to *Paired-ends*, *mapping quality* to 20 and number of mismatches (*NMP*) to 6.

5. In the **Pipeline** > **SNP calling** tab: Select both *iSNPcall* and *mSNPcall* modes and check the *disable BAQ* option. In the filtering section choose *minimum depth* of 3 and *maximum depth* of 100, *minimum base quality* of 13 and *minimum RMS* of 10.

6. In the **Pipeline** > **statistics** tab: check both *iSNPcall* and *mSNPcall* in the *Summarise genotype calls' results* section, and all three boxes in the *Population genomics* section.

7. In the **Run settings** tab: Select a folder where the analysis should be performed (*Data folder*). Leave *Run folder* blank, then choose a prefix for this test run, and set number of replicates to 10. Uncheck the *keep all files* option. In the section *Paths to executables*, set the paths to the different executables, and remember to save them for future analysis (use the *Save paths to file* button)

8. In the **Bash file** tab: press the *Write bash* button. The bash commands should now appear in the text editor. If any settings are incorrect, an error will be raised with information on which error and how to correct it. Otherwise, the pipeline should be correctly defined, and the analysis can be started. Press the *Save bash* button, and write the commands into a file, preferably in the Data folder defined before. This Bash file can now be executed from the terminal.

   There are several ways to execute a bash file, perhaps the simplest is to give it executable mode (type *chmod +x filename.sh*) and then run it with *./filename.sh > out 2> err*. The pipeline defined in the PipelinerGUI will now be executed.

9. Wait until the run finishes, and then check the files *err* and *out*, which are simply text files with the log of the run. In case the run encountered some errors, they should be present in the file *err*. The file can be quite long, and errors difficult to spot, but using a text editor's *search* function (or the *grep* function in terminal), look for the words "ERROR", "fault" and "fail". If present, these likely mean

something went wrong, and will normally be followed by some information on what went wrong. This can be missing executables, wrong file formats, or any other of a myriad of problems.

10. If no errors where encountered during the analysis, there should be 5 files in the Data folder named summary.*runprefix.*(...).txt, and we can proceed to summarise and plot the results. Get back to PipelinerGUI and choose the **Plots** tab. In the upper section of this tab, press the *Add file* button, navigate to the Data folder, and select the two appropriate summary files. In the *name* column, enter either iSNPcall or mSNPcall, according to which summary file is present on the *file* column.

    In the *AAC classes*, choose the *singletons + doubletons* option.

    In the lower section of the Plots tab add the summary files (...)preseq_mstats.txt, (...)iSNPcall_mstats.txt and (...)mSNPcall_mstats.txt, and label then accordingly in the *name* column.

    Now select the file to save the PDF results to, and click *Plot results.* After a brief moment, the resulting pdf file should open automatically with the default PDF viewer of the computer. This PDF will contain a graphical overview of the performance of both individual and multiple-sample SNP calling at the coverage and settings defined earlier, and the population genetics' statistics estimated with the pre-sequencing datasets, and each of the SNP calling results. In the same folder as this PDF, there will also be a file with the commands used to plot these results, which can be loaded into the R package for further refinement of the plots.

11. Once this is done, it is possible to re-run the analysis using different parameter values, either in the input options, or any of the other tabs. For instance, using different average depth (NGS simulation tab), to see how the accuracy, FDR etc change with different coverages. Or choosing a different number of individuals, more stringent options in SNP calling, etc... Then, rerun the analysis, and load the new summary files, together with the previous ones, into the Plot results tab. Plotting the new results will provide a quick graphical overview of the performance of the different options/pipelines.

6 User-defined steps

For every step of the analysis performed by Pipeliner (generating the population and reference, obtaining the NGS short reads, aligning them, and calling variants) there is the option to use User-defined steps. **These User-defined steps allow using any software, insofar as standard data formats are used (fasta, fastaq, bam, vcf)**. This functionality is at the core of Pipeliner, and provides the required flexibility to remain up-to-date with the fast-pace with which new software and methods are devised to analyse NGS data.

All User-defined steps are implemented in a similar way (although each expects different input/output files). Overall, the user must provide Pipeliner with the command-line options (which must be executable from the terminal) required to use the software chosen for the analysis, by entering these steps in the appropriate fields of each User-defined option. The user can specify sequential commands, by writing them on different lines within the User-defined fields. Linux redirection and pipe operators can also be used.

Within these command-lines, the user must include **'tags'** which represent the **input** and **output** files expected by Pipeliner and needed for the corresponding step. In addition to input and output files, tags can be used to denote the **reference genome** (available in some User-defined steps). Finally, the user might want to use **temporary files**, e.g. when the analysis includes more than 1 command, and the output from the first command is needed to run the second (see the examples below). The tags are entered simply by writing their names in the text fields, and should be automatically recognised and highligthed by PipelinerGUI.

With all User-defined steps, there is the possibility to save into or load from a text file the commands to be used, and comments can be included by preceding any line with a #. The folder /examples within Pipeliner's distribution contains a few examples in this format.

Note that **any command written into the User-defined steps will be executed directly on the terminal using the current user's privileges**. Thus care should be taken to avoid using commands that could damage your computer or files - such as removing unwanted files, formatting disks, etc...

1. **User-defined Input step**: In the input tab, the user can specify different software to obtain the population and the reference genome for the analysis. The software must provide a ms-like or a fasta-formatted file, in which the first line will be used as the reference genome, and each 2 consecutive lines thereafter as diploid individuals. Tags available for this step are:

   a) **infile.ms** : the input file in ms-like format.

   b) **infile.fas** : the input file in fasta format.

   c) **tempfile** : any temporary file needed.

   In addition, the user will need to define (in the appropriate fields) the number of diploid individuals which will be simulated and, if using ms-like input, which sequence to use as ancestral.

   Some examples (these examples are provided, as plain-text formatted files, in the folder /examples of Pipeliner's distribution):

- User-defined coalescent simulation with the msms program. Assuming the executable for msms is installed in /home/jsmith/applications, the following line (entered in the User-defined section of the input tab) would simulate a population of 5 diploid individuals (plus reference sequence) which have experienced constant population size and weak selection (example 5.1.1 in msms user's manual):

```
/home/jsmith/applications/msms -N 100000 -ms 11 1 -t 50 -SAA 200 -SaA 100
  -SF 1e-2 > infile.ms
```

- User defined coalescent simulation with the macs program. Assumes macs is installed and in your path, will simulate a 1 Mb genomic region for 5 diploid individuals.

```
macs 11 1000000 -t 0.001 | msformatter > infile.ms
```

2. **User-defined NGS simulation step** (from fasta to fasta-quality files): In the NGS simulation step, the user can define alternative commands to obtain the NGS short-reads. These commands will be executed once for each individual under analysis. The allowed tags on this step are:

    a) **infile.fas**: the fasta-format file containing 2 sequences for the current individual.

    b) **outfile.fq**: the fasta-quality output file containing the short-reads for the current individual. Using this tag will make Pipeliner expect a single-end NGS experiment.

    c) **outfile.1.fq** and **outfile.2.fq**: for paired-end NGS experiments, these two files should contain the forward and reverse direction short reads for the current individual. If used, both tags must be specified. Also, these tags and outfile.fq are mutually exclusive (and will raise an error if present on the same step).

    d) **tempfile** : any temporary file needed.

    e) **ind_number** : current individual number.

    Some examples:

    - User defined NGS simulation step using art. This example shows how to deal with programs that output data into non-customisable files. For instance, using art_illumina the user only specifies the prefix for the output files, and the program adds extension to this prefix to obtain the full name for output files. We assume art_illumina is the name of the executable, which is installed in the computer's path. The first line below runs art with average 10x depth per individual, 75 bp reads etc, and resulting files are named tempfile.1.fq and tempfile.2.fq... In the second and third lines, these files are then renamed so that Pipeliner can proceed with them for the analysis.

```
art_illumina -i infile.fas -p -m 500 -s 10 -l 75 -f 10 -o tempfile
cp tempfile1.fq outfile.1.fq
cp tempfile2.fq outfile.2.fq
```

3. **User-defined alignment step** (from fasta-quality to bam files): In the User-defined alignment step, the user must provide commands to align the NGS short reads into a single bam file for each individual. The allowed tags are:

   a) **infile.fq** or **infile.1.fq** + **infile.2.fq**: the fasta-quality files containing the short reads for the current individual. Only the first or the two last should be present, depending on whether a single- or paired-end NGS experiment is being simulated.

   b) **outfile.bam**: the bam-format file containing the aligned reads for the current individual.

   c) **reference.fa**: the fasta-formatted, interleaved reference genome to use to align the short reads to.

   d) **tempfile** : any temporary file needed.

   e) **ind_number** : current individual number.

   e.g. using bwa:

```
bwa index -a is reference.fa
# align fasq reads to reference
bwa aln -n 6 reference.fa infile.1.fq > tempfile.1.sai
bwa aln -n 6 reference.fa infile.2.fq > tempfile.2.sai

# create bam file and sort it. samtools sort adds an extra .bam
# extension to outfiles, so here we again use a tempfile
bwa sampe reference.fa tempfile.1.sai tempfile.2.sai infile.1.fq infile.2.fq | \
  samtools view -Sb - | samtools sort - tempfile
mv tempfile.bam outfile.bam
```

4. **User-defined SNP calling and filtering step** (from bam to vcf files): In this step, the user must define the commands to call variants from the aligned short-reads files in bam format. There are three sub-steps, including the individual SNP calling, the multiple-sample SNP calling (with all individuals together) and the 'sites' step (see below). The first two steps are optional (at least one must be present), while the 'sites' step is always required.

   a) **Individual SNP calling**: commands to perform SNP calling and filtering for each individual separately. Should start from the bam file for the current individual, and finish with a single vcf file for each individual. The allowed tags here are:

i. **infile.bam**: the current individual's bam file.

ii. **outfile.ind.vcf**: resulting variant-calls for the current individual.

iii. **reference.fa**: the fasta-formatted, interleaved reference genome to use to align the short reads to.

iv. **tempfile** : any temporary file needed.

v. **ind_number** : current individual number.

Some examples:

- Individual SNP calling with GATK (plain-text example provided in folder /examples of Pipeliner's distribution):

```
# index bam file
samtools index infile.bam
# Individual snp call with GATK.  Adapt the path to GenomeAnalysisTK.jar accordingly!
java -jar /apps/GATK/GenomeAnalysisTK.jar -T UnifiedGenotyper -R reference.fa \
  -I infile.bam -o outfile.ind.vcf
```

b) **Multiple-sample SNP calling**: commands to perform SNP calling with all individuals at the same time. The outcome of this step should be a single vcf file, containing the variant calls for all individuals. These variant calls will be filtered with the .sites file obtained in the next step (see below). Allowed tags are:

i. **all.infiles.bam**: list of bam files for all individuals. By default, the list of files will be printed separated by a single space, but the field *separator* can be used to specify otherwise. For instance, GATK expects a "-I" in between each bam file name.

ii. **outfile.allinds.vcf**: vcf file containing variant calls for all individuals.

iii. **reference.fa**: the fasta-formatted, interleaved reference genome to use to align the short reads to.

iv. **tempfile** : any temporary file needed.

Some examples:

- Multiple sample SNP calling with GATK (plain-text example provided in folder /examples of Pipeliner's distribution). Note that tempfile.newref.fa points to the new reference sequence, used in the previous step

```
# multiple-sample SNP calling with GATK. Adapt the path to GenomeAnalysisTK.jar !
java -jar /apps/GATK/GenomeAnalysisTK.jar -T UnifiedGenotyper -R reference.fa \
  -I all.infiles.bam -o outfile.allinds.vcf
# remember to set the separator between bam files to -I!
```

c) **Obtain .sites file**: in order to obtain accurate estimates of Recovery and Accuracy, and of population genetics' statistics, it is imperative to obtain a list of "confident sites". This list should contain all sites that pass the quality control filters, irrespective of the presence or absence of variants. Sites not present in this list are coded as missing, whereas those present are coded according to their genotype call or, if no genotype call is present, as homozygous for the reference allele. This step is performed once for each individual under analysis. Allowed tags:

   i. **infile.bam**: the current individual's bam file.

   ii. **outfile.sites**: list of sites passing all filters.

   iii. **reference.fa**: the fasta-formatted, interleaved reference genome to use to align the short reads to.

   iv. **tempfile** : any temporary file needed.

   v. **ind_number** : current individual number.

   The user must also specify (in the field to the right) in which column of the output file the site numbers are stored, e.g. in a standard vcf file the sites' numbers are on column 2.

   Some examples:

   - Obtaining the .sites file with GATK (plain-text example provided in folder /examples of Pipeliner's distribution):

```
# to obtain the sites file, use the option EMIT_ALL_CONFIDENT_SITES
java -jar /aps/GATK/GenomeAnalysisTK.jar -T UnifiedGenotyper -R reference.fa \
  -I infile.bam -o outfile.sites --output_mode EMIT_ALL_CONFIDENT_SITES
# remember to set column number to 2, and to adapt the path to GenomeAnalysisTK.jar!
```

7 Disclaimer

Without any warranty of any kind etc...

8 License

Pipeliner is distributed under the GNU General Public License version 2.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 dated June, 1991 or at your option any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied