Somatypus

A Platypus-based variant calling pipeline for cancer data

Adrian Baez-Ortega, Maximilian R. Stammnitz and Elizabeth P. Murchison

Transmissible Cancer Group, University of Cambridge

Developed by A.B-O.

February 2016

Contents

Licence

Developer: Adrian Baez-Ortega (ORCID 0000-0002-9201-4420; ab2324@cam.ac.uk)

1. What is Somatypus?

The identification of mutations — widely known as 'variant calling' — in cancer sequence data is challenging, because of the occurrence of copy number alterations, sample contamination from surrounding healthy tissue, and particular difficulties in the alignment of sequence reads to a reference (healthy) genome, among other factors. This has sparked the development of computational tools for variant calling which are specialised for working with cancer data.

Transmissible cancers, which arise in one individual and then spread clonally through the population as an invasive allograft, pose a further challenge, because there is a lack of 'normal' tissue from which healthy DNA can be obtained (there is 'host' tissue instead); this is a consequence of the arousal of the cancer lineage from cells belonging to a different, unknown individual — usually a long time ago. Until now, there was no available software tool or pipeline directed toward variant calling in transmissible cancer samples, mainly due to their extremely rare occurrence — only three diseases of this kind have been described, none of them in humans.

Because of this, we have developed Somatypus, an open-source pipeline for variant calling in cancer data, which is applicable to any kind of cancer sequence data, although it is intended, and particularly useful, for the processing of data obtained from many tumour samples that are closely related to each other, and which lack matched normal samples, such as transmissible cancer or metastasis data, because it does not require the usual 'normal-tumour' sample pairing.

Somatypus is built upon the powerful variant caller Platypus, developed at the Wellcome Trust Centre for Human Genetics (**http://www.well.ox.ac.uk/platypus**). We chose Platypus as our variant caller after proving in many tests that its calling sensitivity and specificity are comparable or better than those of other well-known

callers, besides offering run times as much as 100 times shorter than those of other tools. We have designed our pipeline around several executions of Platypus, with the aim of obtaining maximum sensitivity in the initial calling steps, followed by diverse genotyping and filtering steps which provide a highly accurate removal of false positives and dubious candidate variants, rendering a final set of high-confidence variants. The effect of virtually any step and setting in the pipeline has been tested in detail through manual assessment of random subsets of the resulting variant sets.

Although Somatypus still needs to be statistically benchmarked using standardised public datasets, we believe that it offers remarkable calling power and accuracy. Its estimated false positive rate is <2%; although the false negative rate is difficult to estimate without a benchmarking data set, we believe it to be comparable to that of other callers; we have confirmed that in many cases of missed variants, Somatypus initially calls them, and subsequently discards them based on reasonable criteria about their reliability or that of their sequence context.

Somatypus has been tested on Ubuntu (14.04.3) systems, and it should behave well on any Linux distribution. It has not been tested on Mac systems, but it might work, maybe requiring some minor code modifications. However, the installation of the software dependencies will probably differ for Mac systems, so the installation might need to be done manually.

1.1 Somatypus does…

- Call single nucleotide polymorphisms (SNPs) and short insertions/deletions (indels) from any amount of sequence alignment files (subject to memory requirements), at a probably unmatched speed.
- Filter low-quality or ambiguous variants, while preserving low-frequency (occurring in a very few samples) and aberrant-copy-number (not fitting a diploid model) variants.
- Run seamlessly from a single command.
- Resume its execution after an unexpected interruption.

1.2 Somatypus does not…

- Call long indels or structural variants.
- Manage samples in pairs, or differentiate between ‘tumour’ and ‘host/normal’ samples, or between germ-line or somatic variants. Identification of germ-line variants must be performed downstream.
- Take into account contamination between samples. Identification of variants caused by sample contamination must be performed downstream.
- Allow a customised (different parameters) or partial (only certain steps) execution — unless the source code is altered.

2. Installation

2.1 Dependencies

Somatypus has been designed as a set of Bash and Python scripts that do not require a real installation, but simply adding the software directory to the PATH environment variable, making set-up straightforward. Nevertheless, it has some minimal software dependencies:

- **Python** 2.6 or later (it has not been tested on Python 3.X).
- **Platypus** (**http://www.well.ox.ac.uk/platypus**), which in turn requires **htslib** (**http://www.htslib.org**) — we recommend installing both inside a same directory.
- **VCFtools** (**https://vcftools.github.io**), for using the `vcf-sort` command.
- The **tabix** package (**http://sourceforge.net/projects/samtools/files/tabix**), for using the `tabix` and `bgzip` commands. It should come together with htslib.

2.2 Installing Somatypus

Installing the pipeline is as easy as downloading the latest version's ZIP file (available from the GitHub repository), uncompressing it into the desired location, and run the install_somatypus.sh script. (Replace the example path with your own.)

```
/PATH/TO/somatypus-x.x/install_somatypus.sh
```

This script will automatically download and install any necessary dependencies, and will add the somatypus-x.x/src directory (which contains the pipeline scripts) to the PATH environment variable. It will stop if it detects any problem at any step of the installation. If you are unable to find the cause of the problem, then you should open this script and run the commands in it from the terminal, one by one. The script will ask you for your password, because administrator privileges are needed to complete the installation of some of the dependencies. If you do not have administration privileges in your computer (i.e. you cannot do 'sudo'), then you should follow the alternative installation steps in the INSTALL.txt file.

Once the installation process is finished, you will need to source the .profile file in your home directory, so that the changes made by the installation script are applied. After that, you should be able to run somatypus directly from the command line:

```
source ~/.profile
somatypus
```

(Just after installing Somatypus, you will be able to run it only from the terminal in which you sourced the .profile file. If you need to run Somatypus in other terminal, you must either source the .profile file again, or log out of your session and log in again.)

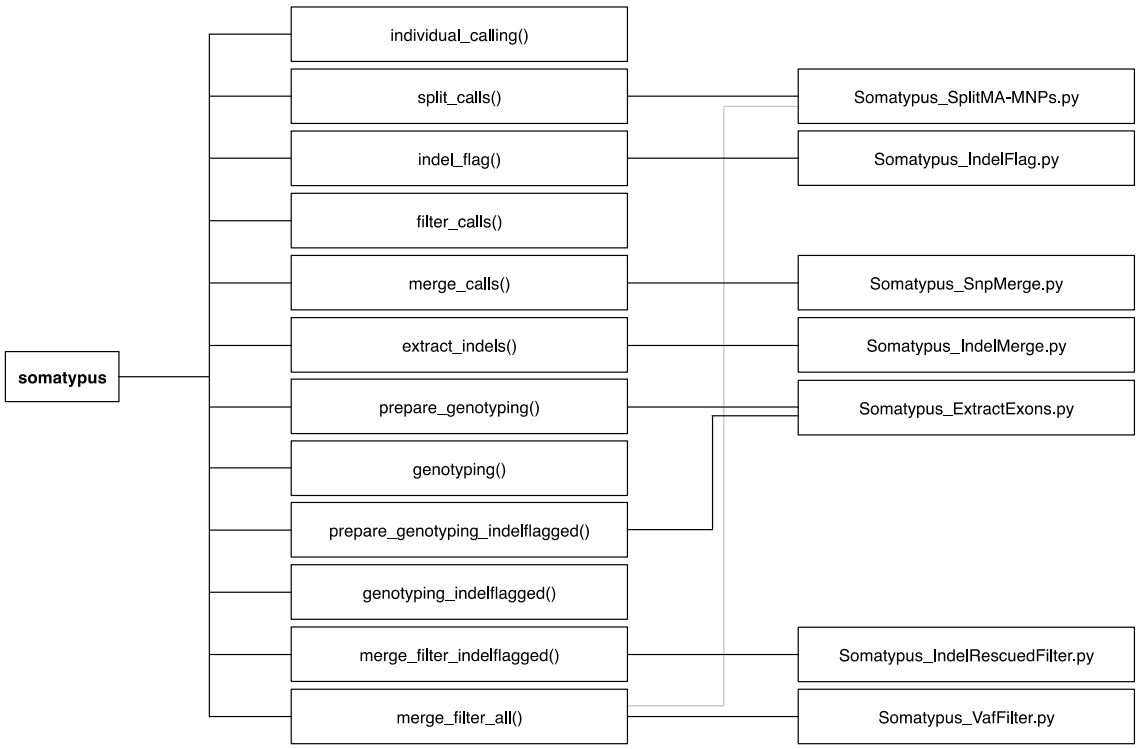
3. Description of the pipeline

3.1 Overview

The Somatypus processing pipeline is designed to produce a set of highly reliable variants (in VCF format) from a set of input alignment files (in BAM format, one file per sample) and a reference genome (in FASTA format). It can also take a set of regions wherein to perform the calling, specified in a text file. The input BAM and FASTA files must be accompanied by .bai and .fai index files, respectively. These can be generated by indexing the input files with the commands `samtools index` and `samtools faidx` (or equivalent ones), respectively.

The pipeline is structured in a modular way: one core Bash shell script, called *somatypus*, is directly called by the user, processes the input and controls the overall execution. This script contains 12 functions, each of which manage one step of the processing workflow; some functions will be called more than once, so there are more pipeline steps than functions. Some of these steps demand operations involving statistical computation and file parsing and editing; these are performed by a set of 7 Python scripts, which are called by the Bash functions when necessary.

Below is a diagram of this structure of scripts and functions; these are arranged, from left to right and from top to bottom, according to their relative execution order in the pipeline.



The major advantage of this kind of modular organisation is that, in the event that the pipeline execution is interrupted before finishing, it can be resumed from the last uncompleted step, thanks to the maintenance of a checkpoint file along the execution.

3.2 Pipeline steps

When the pipeline is called via the `somatypus` command with no options, or with `-h`, it displays a help guide including a brief description and the required inputs.

```
$ somatypus

| SOMATYPUS
| A Platypus-based variant calling pipeline for cancer data
|
| Adrian Baez-Ortega, Maximilian R. Stammnitz and Elizabeth P. Murchison
| Transmissible Cancer Group, University of Cambridge
| Developed by A.B-O. (2016)
|
| Somatypus is a computational pipeline that makes use of the powerful variant caller Platypus for
| calling germ-line and somatic SNPs and indels in sequencing data from a set of unpaired samples.
| It has been designed to offer great sensitivity and specificity, even for low-frequency somatic
| mutations found in cancer genomes. It is particularly useful in cases where there is no matched
| normal sample for each tumour sample, such as transmissible cancers or metastases.
|
| Input:
|    -i  Absolute path to folder containing the input BAM files (accompanied by BAI indices).
|    -o  Absolute path to the output folder (it will be created if needed).
|    -g  Absolute path to reference genome FASTA file (accompanied by FAI index).
|    -r  Absolute path to file of regions to use, one per line in CHR:START-END format. [OPTIONAL]
|    -c  Number of CPUs (processes) for Platypus  *(should not exceed 8 due to a bug)*. [OPTIONAL]
|
| Usage:
|    somatypus -i /path/to/bams_dir -o /path/to/out_dir -g /path/to/genome.fna
-r /path/to/regions.txt -c <1-8>
```

It is advisable that all the input paths be absolute, rather than relative. An optional regions file allows the user to define a set of genomic regions (e.g. exons) wherein to perform the calling. The regions file must be a text file containing one region per line, in CHR:START-END format (e.g. 1:1028676-1028844. The chromosome labels must match those in the reference FASTA and in the sample BAMs).

The number of CPUs is also optional (default is 1) but, if input, must be at least 1, and should not exceed 8, due to a bug in Platypus that can cause an extremely excessive memory allocation attempt (see **Known issues**).

After running the pipeline, the full log of the execution will be contained in the SOMATYPUS.log file, in the logs subfolder of the output directory (besides the logs of most of the steps).

Below is a description of each step involved in the execution of the pipeline, including the names of the scripts and functions that take part in it.

3.2.1 Initialisation

The preliminary step of the workflow is composed by a series of initial validity checks. First, the script checks that the software tools required (Platypus, VCFtools, tabix and the very Somatypus scripts) are installed and accessible from the command line. Second, the existence of all the input files is verified, as well as the presence of the necessary index (.bai and .fai) files; the value of the argument indicating the number of processors to use is also checked to be greater than 0. And third, the script looks for a checkpoint file (which is created inside the *logs* folder during the execution of the pipeline) and, if it exists (due to a previous execution in the same output folder), it reads from it the name and index of the last completed step of the pipeline, and resumes the execution from the step following it. As a consequence of this, an existing output folder that has been used in a previous run of the pipeline should be indicated only if said run ended unexpectedly and needs to be resumed.

Involved: somatypus.

3.2.2 Individual variant calling

The first step of the processing workflow performs variant calling individually on each sample (BAM file). This step is actually divided into two, because variants will be called with two different caller configurations in order to maximise sensitivity and minimise the number of false negatives. There is a checkpoint after each of these steps. The reason for considering each sample individually, instead of all together, is that Platypus, being a haplotype-based caller, tends to group the variants into haplotypes and look at the distribution of these haplotypes across the 'population', using a Hardy-Weinberg equilibrium model (most useful for diploid, non-somatic variants). This causes many rare, low-frequency alleles not to be called because they are not present in the population in a way that satisfies the restrictions of the model. By calling the variants individually in each sample (using special configurations) and then, after a filtering process, genotyping the candidate variants in all the samples at the same type, the pipeline can bypass the restrictions of the Platypus model, substantially increasing sensitivity for detecting low-frequency somatic variants, sometimes present in aberrant copy number.

Therefore, the aim here is to call as many variants as possible; after an individual filtering, the variants from all samples will be merged into a single VCF file, and then genotyped in order to obtain reliable read counts, genotypes and quality estimates by considering all samples simultaneously; a final filtering stage will follow.

The first step runs Platypus independently for each sample, using settings that are very similar to the default ones, except for two changes:

- The minimum number of reads supporting a variant in a sample for the variant to be called (`--minReads` parameter) is increased from 2 to 3. The reason is that we would not trust any variant that has been called with only 2 supporting reads (even if this was half the total number of reads in that position).

- The minimum posterior probability (as defined in the **Platypus supplementary material**) that a variant must have for being called (`--minPosterior` parameter) has been reduced from 5 to 0. This is also recommended for cancer data by the Platypus developing team, because it allows calling variants that otherwise would not be considered real, because they do not fit a diploid segregation within the sample 'population'. (Platypus employs a, which is inadequate for this type of data, although the steps and settings in this pipeline effectively avoid the restrictions of this model.) Setting the minimum posterior to 0 causes many false positives to be called at this stage, but also rescues many real somatic variants that would otherwise be overlooked.

The second calling step calls individually on each sample again, but now using an alternative, less stringent configuration. Apart for the two parameter changes noted above, we have modified the following options:

- The parameter which defines how long groups of SNPs (called MNPs) can be, as well as how many bases at both ends of reads are not considered for calling (`--minFlank`) has been reduced from 10 to 0.

- The parameter which defines how many bases at both ends of reads are forced to have base quality 0, in order to prevent calling in these unreliable regions (`--trimReadFlank`) has been increased from 0 to 10. The reason is to correct for the side effect of allowing calling variants in the first and last 10 bp of reads, caused by the aforementioned `--minFlank` parameter change.

The original purpose of this modification was to call individual SNPs instead of MNPs, making the output less redundant and more reliable; however, it turned out that these relaxed settings were able to rescue variants which had been overlooked by the default settings (together with a large number of false positive that need to be filtered out). Nevertheless, the opposite was also true, so the only apparent solution to the missing of true positives by each of the settings was to combine the output of both. Fortunately, the performance of Platypus is so high that performing the variant calling twice for every sample is scarcely time-expensive in most cases.

Involved: somatypus, individual_calling().

3.2.3 Splitting of multi-allelic and MNP calls

The set of VCF files generated by the individual calling have two features that need to be altered: the grouping of several variants occurring in the same position (and thus sharing a reference allele) into the same file line (henceforth called multi-allelic variants or calls) and the grouping of multiple SNPs into multi-nucleotide polymorphism (MNP) calls. In order to standardise the data before subsequent steps, here all the calls are transformed into bi-allelic calls (having only one reference and one alternate allele), and MNP are split into their constituent SNP calls, so that any line in the files contains either a SNP or an indel, bi-allelic call.

For multi-allelic calls, the information which is specific for each alternate allele is easily split between the resulting bi-allelic calls, so they are as accurate as the original call. However, since MNPs are considered a single variant, the data values for each one of the constituent SNPs cannot be inferred from the values corresponding to the whole MNP. As a consequence, all the SNPs coming from a concrete MNP will share the same exact data, which is now imprecise and unreliable, since it applies to the whole MNP variant. In addition, if one SNP is originally present in more than one variant, the split process will render duplicated SNP calls, each of them having different values (this was the original reason of considering using alternative calling settings). This would be a problem if the information contained in these calls were definitive; but this is not the case, as we only are interested in the variants themselves, and will extract new, reliable values from all the samples at the genotyping stage.

Involved: somatypus, split_calls(), Somatypus_SplitMA-MNPs.py.

3.2.4 Identification of SNPs near indels

Manual assessment of randomly selected called variant during the pipeline design revealed that many false positives in the resulting variant set were caused by nearby indels, which shifted the flanking sequence in a way that could not be detected by the aligner when located too close to the read end; as a consequence of this, what should have been called as an indel (and sometimes was in a portion of the reads), had instead been called as a series of false SNPs. To avoid this, the SNPs located at 5 or less base pairs from any end of an indel (not just from the position where it was called), in any of the samples, are added to an exclusion list, in order to exclude them from the standard genotyping stage and to genotype and filter them separately, using more stringent settings. These variants are hereinafter called *indel-excluded* SNPs.

Involved: somatypus, indel_flag(), Somatypus_IndelFlag.py.

3.2.5 Filtering of individual SNP calls

The variants (including indel-excluded ones) have to be quality-filtered at this point, because they include a large proportion of low-quality calls that are not interesting and whose processing would consume valuable resources. The filtering is performed individually, based on the presence of the following filter flags applied by Platypus (more information can be found in the **Platypus supplementary material**).

- **badReads**: this filter triggers when across reads supporting a variant, the median of the minimum base quality close to the focal site (default 7 bp either side, configurable using `--badReadsWindow`) is too low (default 15 or less, configurable using `--badReadsThreshold`); this identifies systematic local sequencing issues causing an excess of read errors, which are not always accurately reflected in the read quality scores. It also triggers when more than a fraction of reads are filtered out for the candidate generation stage; the default for this is 0.7 (configurable using `--filteredReadsFrac`).

- **MQ**: Platypus computes the root-mean-square mapping-quality of all reads covering the variant site, and filters the variant if this value is less than 40 (configurable using `--rmsmqThreshold`). This statistic is computed using all reads mapping to the region, before any filtering is applied. The default filter threshold works well for Stampy and BWA-mapped reads; for other mappers this threshold may need to be reduced.

- **strandBias**: this filter identifies variants whose support is skewed in terms of reads mapping to the forward and reverse strands, relative to the distribution seen in all reads. Platypus uses the distribution seen overall, rather than say a binomial distribution centred around a fraction of 0.5, because certain experimental designs can give rise to bona fide strand biases. Examples include exon capture, and mapping biases due to the existence of an anchoring point to one end of the sequence, but not the other. Specifically, the reads supporting the variant are tested against a Beta-binomial distribution with parameters α and β such that that smallest of these parameters is 20, and the parameters are such that the mean of the distribution equals the ratio observed in all reads. Variants are accepted if the p-value exceeds 0.001.

- **SC**: polymerase slippage in low-complexity regions is a known cause of spurious indel calls, and in the alignment model Platypus accounts for a higher incidence of these errors. However, in certain instances, particularly when two different but substantial low-complexity regions abut, compensating errors in both low-complexity regions can result in sequence that does not support an indel, but does support an artefactual SNP. Although these occurrences are rare, when they occur these false calls are difficult to spot, because the bases can be of high quality. To avoid calling these variants, a sequence complexity statistic is computed that measures the contribution of the two most frequent nucleotides among the 21 around a site; if this measure exceeds 95%, SNP calls are flagged as suspicious.

- **QD**: in order to avoid calling variants using data from many low quality reads, a value (the QD score) is computed that reflects the total evidence in favour of the variant per read supporting the variant. As a proxy for the total evidence, Platypus uses the Phred-scaled variant posterior as reported in the QUAL field of the VCF file. To avoid a downward bias in low-coverage samples, the Phred-scaled prior of the variant is added to this. The resulting score is divided by the number of reads that support the variant. Variants with a QD value of less than 10 (configurable using `--qdThreshold`) are flagged as suspicious.

Any call showing any of the flags above will be discarded from the individual VCF file; since a variant will be often found in many individual files, it has to be flagged in all of them in order to be effectively filtered out before the subsequent merging step, which makes this initial filter notably lax.

Because of the particularities of cancer data, we allow the following flags to be present in our samples:

- **alleleBias**: this filter identifies variants that show support in too few reads compared to the expectation under heterozygous segregation in a diploid organism.
- **HapScore**: this filter indicates that too many (>4) haplotypes are supported by the data in this region.
- **Q20**: the posterior for calls is calculated in the usual way (as prior times likelihood of the call, divided by the sum of prior times likelihood over all genotypes considered), and reported as a “genotype quality Phred score” (the Phred-scaled probability of the call being wrong, or 10 times negative 10-log of one minus the posterior) in the per-sample GQ field. Although variants are called, and included in the output VCF, if they have a Phred-scaled posterior exceeding 5, all variants with posteriors below 20 are flagged as suspicious. (This flag must be allowed in concordance with our `--minPosterior` setting for calling variants with a very low posterior.)

The remaining flags mentioned in the Platypus documentation (GOF, hp10, QualDepth, and REFCALL) were not found in any of our variant sets, and are probably not being used in the latest Platypus version, so they are not considered here. The PASS flag indicates that the variant did not trigger any filter during the calling.

Involved: somatypus, filter_calls().

3.2.6 Merging of individual SNP calls

The filtered SNP calls coming from each sample are now merged in a single VCF file, so that the same chromosome (CHROM), position (POS), reference allele (REF) and alternate allele (ALT) values will not be present in more than one call in the merged VCF. The exclusion list built in the indel-flagging step will be used here to prevent SNPs located near indels from being included in the merged file. They will be merged into a different VCF file for separate genotyping.

In order to avoid the creation of multi-allelic calls (caused by more than one variant occurring at the same position) during the genotyping stage, three merged VCFs (at most) will be generated, so that if there are three different SNPs occurring at the same location, each one of them will be considered as a different allele, and output to one of these files. There will therefore be an ‘allele 1 VCF’, an ‘allele 2 VCF’ and an ‘allele 3 VCF’, as well as other three analogous files for the indel-excluded SNP set. The aim of this is to prevent different variants from sharing certain statistics, and more importantly, sharing Platypus filter flags (which relate to each call, not to each variant), which can cause some false positives to pass our filters because they are grouped together with a true positive, sharing its flags.

It is evident that, since each call in the merged VCF comes from an undetermined particular sample, only the CHROM, POS, REF and ALT values will be reliable, the other fields being valid only for the individual sample in which the variant was called.

Because we only need to know the genomic position and the base change of the SNPs at this stage, this is not a problem.

Involved: somatypus, merge_calls(), Somatypus_SnpMerge.py.

3.2.7 Identification of high-confidence indels

Now, indels are extracted from the individual VCFs, merged and included in the allele 1 merged SNP VCF file (because all the indel-excluded SNPs have been excluded from the merged files, these indels will not share position with any of the SNPs in the file). When selecting indels, a much more stringent approach is followed: only indel calls which are bi-allelic (having only one alternate allele) and have a Platypus PASS flag (they did not trigger any quality filter during calling) are extracted from the original, individual VCF files, merged according to genomic position and base change, and appended to the allele 1 VCF. The set is also examined for indels that share position but come from different samples (that is, they never share position in the same VCF file); these are also excluded.

Involved: somatypus, extract_indels(), Somatypus_IndelMerge.py.

3.2.8 Genotyping preparation

This step involves the generation of the VCF and (optionally) regions files that are needed during the genotyping of SNPs and indels. First, each of the three merged VCF files corresponding to the three 'alleles' defined at the variant merging step needs to be sorted, compressed and indexed, in order to be readable by Platypus. Then, if there is a regions file available, the regions that contain any variant in each of the alleles are extracted to a corresponding new regions file (there are, therefore, three new regions files). This makes the genotyping of the second and third alleles, that contain much less variants than the first, considerably faster, because only the relevant regions are examined. And last, a new file containing the paths to each of the BAM files, which is needed as an input to Platypus, is created.

Involved: somatypus, prepare_genotyping(), Somatypus_ExtractExons.py.

3.2.9 Genotyping of SNPs and indels

The genotyping stage is divided in three steps (with checkpoints between them), corresponding to the genotyping of the variants belonging to the alleles 1, 2 and 3. These three Platypus runs are therefore executed serially; however, the run time of the genotyping of alleles 2 and 3 will be much lower than that of allele 1, because of a considerably smaller number of variants (and regions) to look at. The settings applied for the genotyping are the same as in the first individual calling step (`--minReads=3`, `--minPosterior=0`). In addition, Platypus is now told not to call variants from the BAM files (`--getVariantsFromBAMs=0`), but to read the variants directly from the merged, sorted, and compressed VCF file of the corresponding allele (`--source` parameter). Thus, instead of calling variants *de novo*, the processed variants contained in the VCFs will be genotyped in every sample, generating three new VCF files.

However, genotyping is usually not just as straightforward when using cancer data. Because some of the variants present in the input merged VCF will not fit well in the population model used by Platypus (one of the reasons for making the calling individual to each sample), they will not be included in the output genotyped VCF. (The percentage of these missing variants after the genotyping is normally below 5%.) In order to rescue them, a second (but quick) Platypus run is done for each allele, where the regions to consider correspond to the exact positions occupied by the missing variants; these are obtained by comparing the merged VCF and the genotyped VCF. If no missing variants are found after genotyping, this step will be omitted.

Involved: somatypus, genotyping().

3.2.10 Genotyping preparation for SNPs near indels

After the 'normal' SNPs and indels have been genotyped, separate genotyping of the SNPs near indels (or indel-excluded) is performed. It has been seen that this set of excluded SNPs contains a small but significant proportion of true positives, so a special filtering step has been designed in order to rescue these real variants that are located near indels. A small proportion (~10%) of false positives may result from this process, but they are cases where only the human eye can notice that they are artefactual SNPs caused by the presence of indels or sequencing noise.

Setting up the genotyping involves the same steps as for the rest of variants, with the only difference that the VCF files to process are the ones that contain the indel-excluded SNPs.

Involved: somatypus, prepare_genotyping_indelflagged(), Somatypus_ExtractExons.py.

3.2.11 Genotyping of SNPs near indels

The genotyping of the indel-excluded SNPs is identical to the previous genotyping (except, obviously, for the input and output files), being composed of three Platypus executions. As before, missing variants are looked for and re-genotyped, so that there will be (up to) two genotyped VCFs per allele.

Involved: somatypus, genotyping_indelflagged().

3.2.12 Merging and filtering of SNPs near indels

After all the variants have been genotyped, the indel-excluded SNPs in the (normally) six VCFs resulting from the genotyping of the three alleles are merged into a single file. Then, they are quality-filtered using the same criteria as in the individual filtering step (see **Filtering of individual SNP calls**). And finally, a specific filter is applied, which discards SNPs with median read coverage below 20, or median VAF (the proportion of reads supporting the variant at the variant locus) below 0.2 or above 0.9. The median VAF is computed considering only samples having at least three reads that support the variant in question. These thresholds have been designed to filter out most of the false positives present in the indel-excluded set, and have proven to be especially effective.

Involved: somatypus, merge_filter_indelflagged(), Somatypus_IndelRescuedFilter.py.

3.2.13 Merging and filtering of all variants

The last step involves merging all the variants (those coming from the six genotyped VCFs generated at the first genotyping step, and the filtered indel-excluded SNPs) into a single file. They are then quality-filtered using the same criteria as in the individual filtering step (see **Filtering of individual SNP calls**). This set can still contain some few 2-bp MNPs, which actually are composed of just one SNP, so the splitting script needs to be called again in order to transform these into SNPs (however, no inaccurate or redundant calls are generated this time). After this, a last filter discards those variants with a VAF above 0.9 in *all* the samples, since this are most likely sequencing errors; indels that have any flag other than PASS after the genotyping are also removed at this stage. At this stage, SNPs and indels are separated and output to two respective VCF files. Finally, the VCFs are sorted and the pipeline finishes.

Involved: somatypus, merge_filter_all(), Somatypus_SplitMA-MNPs.py, Somatypus_VafFilter.py.

3.3 Output

The output variant set is contained in the files Somatypus_SNPs_final.vcf and Somatypus_Indels_final.vcf, in the output directory. The remaining files and folders are temporary and can be deleted if not needed.

The VCF files generated by Platypus begin with a 48-line header. The last line in the header contain the names of the data columns. From line 49 onwards, each line of the output VCF contains a single variant, which is described by the following fields.

- CHROM: chromosome where the variant is located.
- POS: genomic position where the variant is located inside the chromosome.
- ID: variant identifier. In Platypus, it is always a dot.
- REF: reference allele.
- ALT: alternate allele.
- QUAL: numeric quality value of the variant, corresponding to its Phred-scale posterior. (The developers warn that it is not reliable, and we have found many real variants showing low QUAL values.)
- FILTER: Platypus filter flag. Only PASS, Q20, alleleBias and HapScore flags are accepted in the pipeline (see **Filtering of individual SNP calls**).
- INFO: contains statistical information about the variant. The meaning of each value in this field is explained in the VCF header.

- FORMAT: defines the order and meaning of each of the values included in the sample data. The sample data is always displayed as a collection of 6 sub-fiels, delimited by colons: GT:GL:GOF:GQ:NR:NV. Each sub-field is described below.
    - GT: unphased genotypes (usually, 0/0, 0/1, 1/0 or 1/1). Since the somatic variants coming from cancer samples are not necessarily bi-allelic, the use of this field is discouraged.
    - GL: genotype log10-likelihoods for ref-ref, ref-alt and alt-alt genotypes. Since the somatic variants coming from cancer samples are not necessarily bi-allelic, the use of this field is discouraged.
    - GOF: goodness-of-fit value. We have not found this value to be related with variant quality.
    - GQ: genotype quality as Phred score. Since the genotype information can be inaccurate for cancer variants, the use of this field is discouraged.
    - NR: total number of reads (read coverage) at the variant position.
    - NV: number of reads supporting the variant at the variant position. The VAF is computed as NV/NR.
- Sample data: after the FORMAT field, each column in the file contains the data of the variant for one sample (which gives the column its name), in the format described by the FORMAT field.

4. After running Somatypus

Although Somatypus has been designed specifically for calling in unpaired samples (because there is no normal sample available in transmissible cancers), its main downfall is its inability to distinguish between tumour and normal/host samples, and therefore, between germ-line variants (or variants caused due to sample contamination) and somatic mutations. This should be done in downstream processing steps; a Python script and an R script have been included (in the somatypus-x.x/utils folder) that allow:

- The extraction of the most valuable data from the VCF files, which will substantially reduce the time required for any downstream computations (ExtractVcfData.py). After the installation of the pipeline, this script can be run directly from the command line, receiving as its only argument the path to the input VCF file.
- The import of these data into R; the decomposition of the variant set between variants present in any normal/host sample, and variants present exclusively in tumours, having into account contamination between samples; and the plotting of basic VAF and coverage graphs (BasicManipulation.R). For this code to

work, the tumour samples — unlike the normal/host samples — must contain a 'T' letter in their labels. This script is just an example of how to differentiate between the two types of samples, and how to discover if a variant is present only in tumours. It should be run step-by-step from an IDE like RStudio, not directly from the command line, because it does not write the resulting variant categories to any file (if you want to have easy access to them for subsequent analyses in R, you should save them into an RData file).

5. Known issues

Although Platypus is an outstanding variant caller, it does have some bugs. Here are listed the ones that we have encountered during the development and testing of Somatypus (using Platypus 0.8.1):

- **You must ensure that the names of your samples do not overlap between each other.** If that happens, Platypus will mistake the samples because of another bug, and the log will register an error, which is accurately described by the message 'Something is screwy here'. This message necessarily means that the names of your samples overlap; for example, there can be a sample called 622-T, which overlaps with another one called 22-T. In that case, you should rename 22-T (and the rest of possibly conflicting samples) to something like 022-T. Yet another difficulty lies in knowing where Platypus takes the sample name from. Normally, it will read it from the BAM header ('SM' field), implying that you will need to re-header the files using SAMtools or a equivalent method. However, if Platypus finds any field in the SAM header which it cannot interpret (this will be reflected in the log), it will make the sample name equal to the BAM file name. If that is the case, you will only need to rename the files, which is much easier.

- If the number of CPUs exceeds 8, Platypus can try to allocate an extremely excessive memory amount, which will make the involved subprocess to be killed, but will probably not cause the execution to finish (however, some variants could be missed). This surreptitious bug can be spotted by searching for the word 'memory' (using `grep -i "memory"`) in the Platypus log files, as well as in the SOMATYPUS.log file. If the bug did not occur, you should not get any matches; however, if it did occur, you will find a line that includes the words 'memory exception'. This bug is very unlikely to happen if the number of CPUs used is 8 or less.

If you find any other bug or problem while using Somatypus, please do not hesitate in contacting the developer (ab2324@cam.ac.uk).