Abstract

Motivation: Visualizing BAM and VCF files is a common task for biologists, but they're missing a way to filter and to explore the details of each short-read or variation.
Results: In that context, we wrote an interactive java-based interface named *JfxNgs* that uses javascript snippets to filter and reformat BAM and VCF files.
Availability: https://github.com/lindenb/jvarkit/blob/master/docs/JfxNgs.md
Contact: pierre.lindenbaum@univ-nantes.fr.

JfxNgs : A BAM/VCF viewer with javascript-based filtering/reformatting functionalities.

Pierre Lindenbaum (i), Matilde Karakachoff (ii), and Richard Redon (i)

(i)L'Institut du thorax, INSERM, CNRS, UNIV Nantes
8 quai Moncousu, 44000 Nantes, France.
(ii) L'Institut du thorax, INSERM, CNRS, UNIV Nantes, CHU Nantes
8 quai Moncousu, 44000 Nantes, France.
pierre.lindenbaum@univ-nantes.fr

March 2017

1 Description

Powerful tools like the *Integrative Genomics Viewer (IGV)* (Thorvalds *et al.*, 2013) are very efficient at visualizing the features laying in a genomic region but they're not really suitable for exploring the details of each item (quality, functional annotations, ...). Biologists in our lab currently explore the VCF files using knime4bio (Lindenbaum *et al.*, 2011), a plugin we developped for the knime plateform. But by breaking the VCF structure into a table, we lose important informations like the VCF header, and we cannot use the existing programming interfaces like the *'java API for high-throughput sequencing data formats'* (htsjdk) (htsjdk, 2016) to analyze the data. Hence, we have written '*JfxNgs*' a java heavy client displaying the items of some VCF and BAM files. Nevertheles we do not see *JfxNgs* as a competitor of *IGV* but rather like a companion that will be used to get the details of each record in the file.

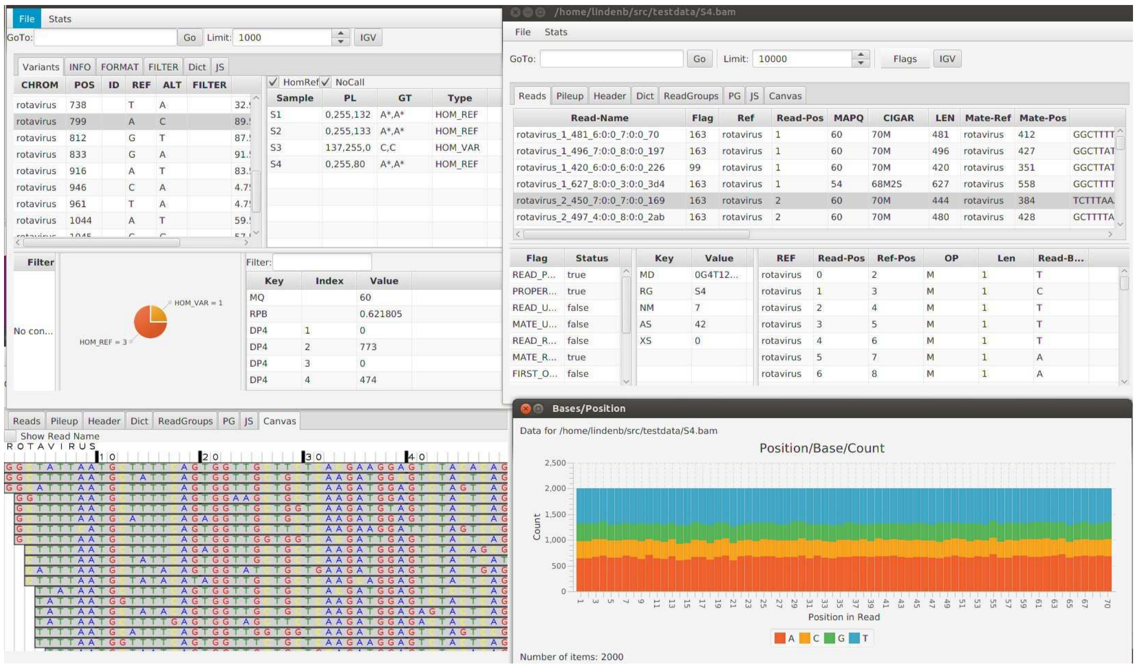
We have taken advantage that the standard edition of 'java' distributed by Oracle contains '*nashorn*', an embedded javascript engine. This engine gives the abality to manipulate java objects using simple javascript statements. When those objects are created by htsjdk (htsjdk, 2016), it gives the users a very simple way to filter or reformat a VCF or a BAM file. We had previously implemented this idea in the tools *FilterVcf* and *FilterSamReads* from the *picard* (Picard, 2017) suite and in *jvarkit* (Lindenbaum, 2015), and we have now implemented it in the *JfxNgs* interface (Fig. 1 and 2).

```
function myfilter(vc) {
    var countDP = 0;
    for(var i = 0; i < vc.getNSamples() ;++i) {
        var genotype = vc.getGenotype(i);
        if( ! genotype.hasDP() ) continue;
        var dp = genotype.getDP();
        if( dp >= 200 ) continue;
        if( ++countDP >= 2 ) return true;
        }
    return false;
    }
myfilter(variant);
```

Figure 1: A simple javascript-based filter retaining the variants having at least two genotypes with a depth lower than 200 reads: an instance of the htsjdk java class htsjdk.variant.variantcontext.VariantContext named '*variant*' is injected in the javascript context, the function 'myfilter' is invoked. It runs a loop over all the samples and count the number of genotypes having a depth ('DP') lower than 200.

```
while(iter.hasNext()) {
 var samread = iter.next();
 if( samread.getReadUnmappedFlag() ) continue;
 out.println(samread.getContig() + "\t" +
    (samread.getStart()-1) + "\t" +
     samread.getEnd());
}
```

Figure 2: Reformatting a BAM file to a BED file. An iterator named '*iter*', and scanning some instances of the htsjdk java class htsjdk.samtools.SAMRecord, is injected in the javascript context. As long as we can read a SAM record, mapped on the reference, its' genomic location is printed to '*out*', the current output stream.



CHROM	POS	ID	REF	ALT	FILTER	
rotavirus	738		T	A		32.
rotavirus	799		A	C		89.
rotavirus	812		G	T		87.
rotavirus	833		G	A		91.
rotavirus	916		A	T		83.
rotavirus	946		C	A		4.7
rotavirus	961		T	A		4.7
rotavirus	1044		A	T		59.

Sample	PL	GT	Type
S1	0,255,132	A*,A*	HOM_REF
S2	0,255,133	A*,A*	HOM_REF
S3	137,255,0	C,C	HOM_VAR
S4	0,255,80	A*,A*	HOM_REF

Key	Index	Value
MQ		60
RPB		0.621805
DP4	1	0
DP4	2	773
DP4	3	0
DP4	4	474

Read-Name	Flag	Ref	Read-Pos	MAPQ	CIGAR	LEN	Mate-Ref	Mate-Pos	
rotavirus_1_481_6:0:0_7:0:0_70	163	rotavirus	1	60	70M	481	rotavirus	412	GGCTTTT
rotavirus_1_496_7:0:0_8:0:0_197	163	rotavirus	1	60	70M	496	rotavirus	427	GGCTTAT
rotavirus_1_420_6:0:0_6:0:0_226	99	rotavirus	1	60	70M	420	rotavirus	351	GGCTTAT
rotavirus_1_627_8:0:0_3:0:0_3d4	163	rotavirus	1	54	68M2S	627	rotavirus	558	GGCTTTT
rotavirus_2_450_7:0:0_7:0:0_169	163	rotavirus	2	60	70M	444	rotavirus	384	TCTTTAA
rotavirus_2_497_4:0:0_8:0:0_2ab	163	rotavirus	2	60	70M	480	rotavirus	428	GCTTTTA

Flag	Status
READ_P...	true
PROPER...	true
READ_U...	false
MATE_U...	false
READ_R...	false
MATE_R...	true
FIRST_O...	false

Key	Value
MD	0G4T12...
RG	S4
NM	7
AS	42
XS	0

REF	Read-Pos	Ref-Pos	OP	Len	Read-B...
rotavirus	0	2	M	1	T
rotavirus	1	3	M	1	C
rotavirus	2	4	M	1	T
rotavirus	3	5	M	1	T
rotavirus	4	6	M	1	T
rotavirus	5	7	M	1	A
rotavirus	6	8	M	1	A

Figure 3: A Screen of jfxngs with a VCF and a BAM window. The 'JS' tabs provide a text area to write a javascript code to filter the data.

JfxNgs can read local files as long as they've been indexed with tabix or tribble. It can also access the remote files if the hosting server supports 'Byte-Range' requests. The main window provides tools for indexing BAM and VCF files. The VCF and the BAM windows have common functionalities: filtering the data with javascript, displaying the data in a very simple genome browser, viewing the selected item in a web browser for a common database like Exac (Song *et al.*, 2015), displaying simple statistics (Fig. 3), exporting the data with javascript, viewing the current selected genomic interval in *IGV*, viewing the components of the file header, saving the filtered data in a new file. The javascript areas contain a growing library of code snippets to help the user.

The BAM window shows the standard columns of the BAM specification (Read-Name, Sam-Flags, etc...), it provides some tables to view the details of the cigar string, the sam flags, the supplementary alignments. A pileup-like table displays the bases covering each position.

The VCF window displays the standard columns of a VCF file (CHROM, POS, etc...) , it provides some tables to clearly view the INFO, FILTER, ALTS data. If the annotations of a prediction algorithm like VEP (McLaren *et al.*, 2010) of SNPEff (Cingolani *et al.*, 2012) are detected, the predictions for each transcripts are displayed in a new table. A table of each genotypes displays the calls for each variant with the ability to filter out the Hom-Ref and No-Call genotypes. A pedigree file can be associated to a VCF and is then used to detect the Mendelian incompatibilities.

2 Availability

The software is available in the jvarkit package (Lindenbaum, 2015). It has been tested under Linux, Windows and MacOS and is freely available at https://github.com/lindenb/jvarkit/blob/ma
At the time of writing, our tool is also available, packaged as a java webstart application, at http://redonlab.univ-nantes.fr/public_html/jnlp/jfxngs, meaning that it doesn't require any installation but java and an always up-to-date application is downloaded each time the user invokes it.

Acknowledgements

We want to thank the bioinformatics core facility of Nantes (Biogenouest) for technical support.

Funding

This work was supported by the Institut National de la Santé et de la Recherche Médicale (INSERM).

References

Cingolani, P., Platts, A., Wang, l. e. L., Coon, M., Nguyen, T., Wang, L., Land, S. J., Lu, X., and Ruden, D. M. (2012). A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3. *Fly (Austin)*, **6**(2), 80–92.

htsjdk (2016). A java api for high-throughput sequencing data. https://github.com/samtools/htsjdk. [Online; accessed 17-Mar-2017].

Lindenbaum, P. (2015). JVarkit: java-based utilities for Bioinformatics.

Lindenbaum, P., Le Scouarnec, S., Portero, V., and Redon, R. (2011). Knime4Bio: a set of custom nodes for the interpretation of next-generation sequencing data with KNIME. *Bioinformatics*, **27**(22), 3200–3201.

McLaren, W., Pritchard, B., Rios, D., Chen, Y., Flicek, P., and Cunningham, F. (2010). Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor. *Bioinformatics*, **26**(16), 2069–2070.

Picard (2017). Picard: A set of command line tools (in Java) for manipulating high-throughput sequencing (HTS) data and formats. [Online; accessed 16-Mar-2017].

Song, W., Gardner, S. A., Hovhannisyan, H., Natalizio, A., Weymouth, K. S., Chen, W., Thibodeau, I., Bogdanova, E., Letovsky, S., Willis, A., and Nagan, N. (2015). Exploring the landscape of pathogenic genetic variation in the ExAC population database: insights of relevance to variant classification. *Genet. Med.*

Thorvalds, H., Robinson, J. T., and Mesirov, J. P. (2013). Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Brief. Bioinformatics*, **14**(2), 178–192.