Comparison of VCF files comparing tools

Author: Sebastian Dziura

Introduction

Development of science always goes along with the development of technology. It especially applies to genomics.

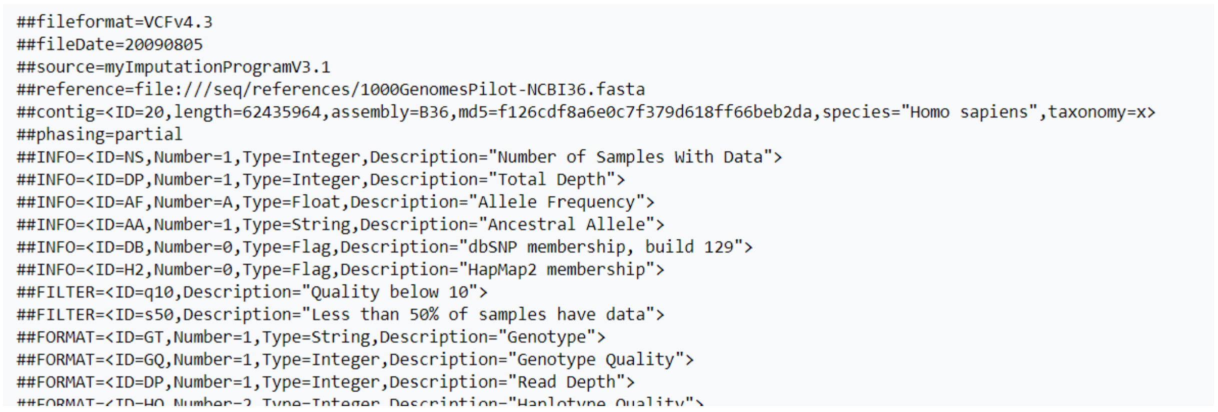
For better understanding of our own bodies, it is necessary to understand human DNA. Double helical structure was deduced in 1952-1953 and it is still bringing us new discoveries all the time. However researching DNA is not an easy task. In human DNA there are around 3 billion bases. It requires sophisticated machines and techniques to read sequences of DNA bases. In typical whole genome sequencing experiment there are 200-500 million of 50-150 bases long. These sequences must be later reconstructed, which requires tools that will perform algorithms for assembly or alignment.

Since the sequence is already reconstructed, it is the time for variant calling. More than 99% of DNA is the same for all humans, so the most interesting part is the one that is different. Variant calling is the process of identifying that part. This is important, because in variants there are hidden sources of many diseases and health issues.

Since there is only significant part left, more precise analysis can be performed. The next thing that can be done is comparing variants form different samples. It can be same sequence from different callers to check the quality their quality. It can be also from two different people, to find out how each variant can influence our lives. Comparing trio father-mother-son DNA sequences can bring many information about inheritance. For this task there are many tools that can be used and two of them are tested and compared in this work.

Comparing VCF files

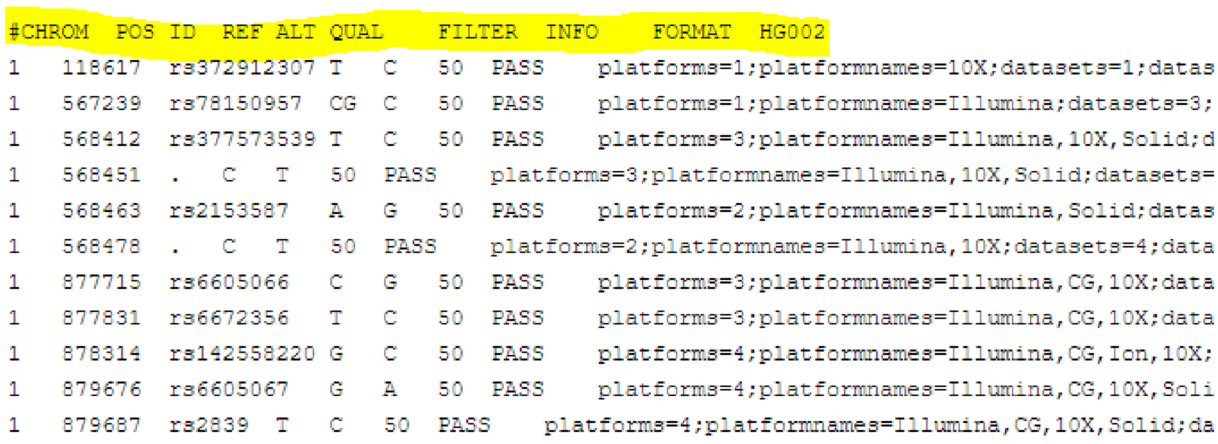
VCF (Variant Call Format) is an output file from the variant calling process. It consists lines starting with “##”. It provides metadata describing the body of the file. On picture 1 there is example of such meta-information lines.



```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
```

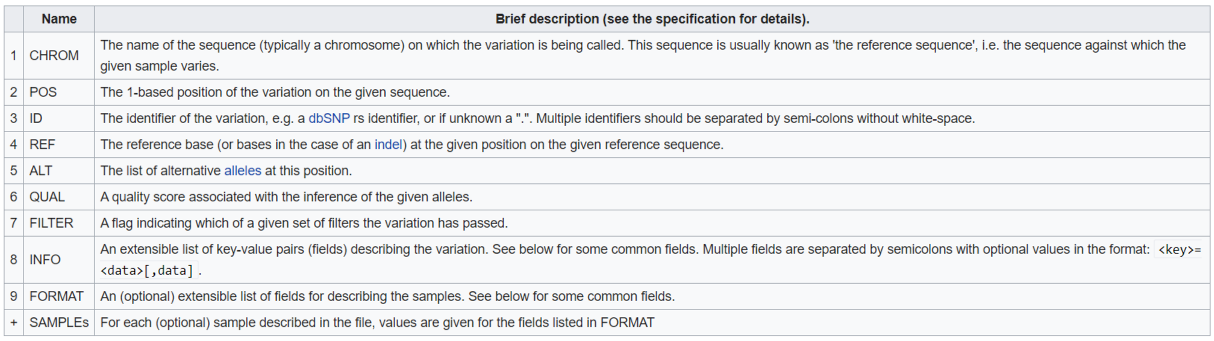
Picture 1- Example of VCF file header [2]

Above the proper data, there is also header line denoted with single “#”. There are labeled names of all the columns. Piece of data with labels above are presented at picture 2 and description of the fields is at picture 3.



```
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO    FORMAT  HG002
1   118617  rs372912307 T   C   50  PASS    platforms=1;platformnames=10X;datasets=1;datas
1   567239  rs78150957  CG  C   50  PASS    platforms=1;platformnames=Illumina;datasets=3;
1   568412  rs377573539 T   C   50  PASS    platforms=3;platformnames=Illumina,10X,Solid;d
1   568451  .   C   T   50  PASS    platforms=3;platformnames=Illumina,10X,Solid;datasets=
1   568463  rs2153587   A   G   50  PASS    platforms=2;platformnames=Illumina,Solid;datas
1   568478  .   C   T   50  PASS    platforms=2;platformnames=Illumina,10X;datasets=4;data
1   877715  rs6605066   C   G   50  PASS    platforms=3;platformnames=Illumina,CG,10X;data
1   877831  rs6672356   T   C   50  PASS    platforms=3;platformnames=Illumina,CG,10X;data
1   878314  rs142558220 G   C   50  PASS    platforms=4;platformnames=Illumina,CG,Ion,10X;
1   879676  rs6605067   G   A   50  PASS    platforms=4;platformnames=Illumina,CG,10X,Soli
1   879687  rs2839  T   C   50  PASS    platforms=4;platformnames=Illumina,CG,10X,Solid;da
```

Picture 2 – Columns of data with labels marked with yellow marker



	Name	Brief description (see the specification for details).
1	CHROM	The name of the sequence (typically a chromosome) on which the variation is being called. This sequence is usually known as 'the reference sequence', i.e. the sequence against which the given sample varies.
2	POS	The 1-based position of the variation on the given sequence.
3	ID	The identifier of the variation, e.g. a dbSNP rs identifier, or if unknown a ".". Multiple identifiers should be separated by semi-colons without white-space.
4	REF	The reference base (or bases in the case of an indel) at the given position on the given reference sequence.
5	ALT	The list of alternative alleles at this position.
6	QUAL	A quality score associated with the inference of the given alleles.
7	FILTER	A flag indicating which of a given set of filters the variation has passed.
8	INFO	An extensible list of key-value pairs (fields) describing the variation. See below for some common fields. Multiple fields are separated by semicolons with optional values in the format: `<key>=<data>[,data]` .
9	FORMAT	An (optional) extensible list of fields for describing the samples. See below for some common fields.
+	SAMPLEs	For each (optional) sample described in the file, values are given for the fields listed in FORMAT

Picture 3 – Description of columns [2]

When comparing VCF files, the point is to find where they are different. The baseline and called sequences are compared, so as the result we are informed which variants match (they are marked as TP – true positives), which are private for baseline (FN- false negatives) and which are private for called sequence (FP- false positive). They are usually presented in separate VCF files as an output.

Tools

RTG tools

First of tools kit is *RTG tools*. The particular command that will be discussed is *vcfeval.* As its said at their github page :*" RTG vcfeval performs variant comparison at the haplotype level, that is, it determines whether the genotypes asserted in the VCFs under comparison result in the same genomic sequence when applied to the reference genome. This in itself is a non-trivial problem and naive approaches face a combinatorial explosion to determine the most accurate analysis. To date, no other tool is capable of performing this analysis as accurately and as fast as RTG vcfeval. RTG developed vcfeval for in-house use in 2010, and through our collaborations we found this tool to be highly useful outside of RTG. RTG vcfeval outputs VCF files containing the results of comparison, summary metrics, and ROC curve data files."* [6]

This gives an advantage over conventional tools which can't deal with more complex variants, as they are only directly comparing variant positions, alleles, and genotypes.

BCFtools

Second tool is *BCFtools isec*. It is complex than the previous one. In the documentation it is described : "*Creates intersections, unions and complements of VCF files. Depending on the options, the program can output records from one (or more) files which have (or do not have) corresponding records with the same position in the other files.*" [5]

Data

To compare tools, two tests have been performed. For both tests *HG38* has been used as reference sequence.*” GRCh38/hg38 is the assembly of the human genome released December of 2013, that uses alternate or ALT contigs to represent common complex variation, including HLA loci. Alternate contigs were also present in past assemblies but not to the extent we see with GRCh38. Much of the improvements in GRCh38 are the result of other genome sequencing and analysis projects, including the 1000 Genomes Project.”* [8]

Test 1

As a input data, two of the *AshkenazimTrio* (Ashkenazim Jewish ancestry father, mother and son) were used. Both samples were taken from GIAB.

Baseline:	*HG003_NA24149_father*
Called:	*HG002_NA24385_son*

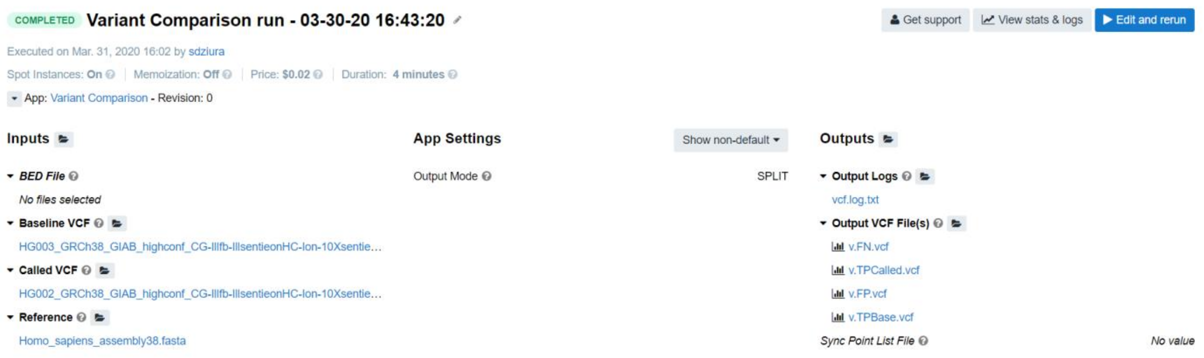
Test 2

In this case samples from the same genome, but two different callers were compared.

Baseline:	*NA12878_HG001 from GIAB*
Called:	*NA12878_HG001 from Platinum Genomes*

Analysis

Tests were performed at *CGC Platform* from *Seven Bridges Genomics.* Platform provides the tools and simplify the process of using them. At picture 4 and 5 there are shown screens from the tasks ordered to tools at platform. The only thing to do is to choose input files and parameters and the coding is done automatically.


COMPLETED **Variant Comparison run - 03-30-20 16:43:20**

Get support | View stats & logs | Edit and rerun

Executed on Mar. 31, 2020 16:02 by sdziura

Spot Instances: On | Memoization: Off | Price: $0.02 | Duration: 4 minutes

App: Variant Comparison - Revision: 0

Inputs

BED File
No files selected

Baseline VCF
HG003_GRCh38_GIAB_highconf_CG-Illfb-IllsentieonHC-Ion-10Xsentie…

Called VCF
HG002_GRCh38_GIAB_highconf_CG-Illfb-IllsentieonHC-Ion-10Xsentie…

Reference
Homo_sapiens_assembly38.fasta

App Settings Show non-default

Output Mode	SPLIT

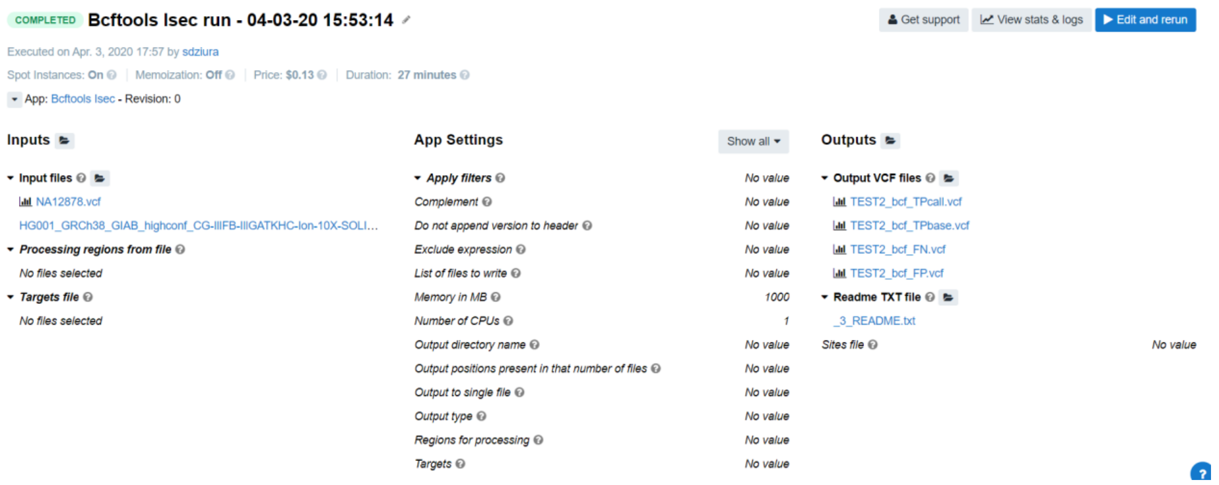
Outputs

Output Logs
vcf.log.txt

Output VCF File(s)
v.FN.vcf
v.TPCalled.vcf
v.FP.vcf
v.TPBase.vcf

Sync Point List File — *No value*

Picture 4 – Task view of vcfeval command from RTG tools at CGC platform


COMPLETED **Bcftools Isec run - 04-03-20 15:53:14**

Get support | View stats & logs | Edit and rerun

Executed on Apr. 3, 2020 17:57 by sdziura

Spot Instances: On | Memoization: Off | Price: $0.13 | Duration: 27 minutes

App: Bcftools Isec - Revision: 0

Inputs

Input files
NA12878.vcf
HG001_GRCh38_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-10X-SOLI…

Processing regions from file
No files selected

Targets file
No files selected

App Settings Show all

Apply filters	*No value*
Complement	*No value*
Do not append version to header	*No value*
Exclude expression	*No value*
List of files to write	*No value*
Memory in MB	*1000*
Number of CPUs	*1*
Output directory name	*No value*
Output positions present in that number of files	*No value*
Output to single file	*No value*
Output type	*No value*
Regions for processing	*No value*
Targets	*No value*

Outputs

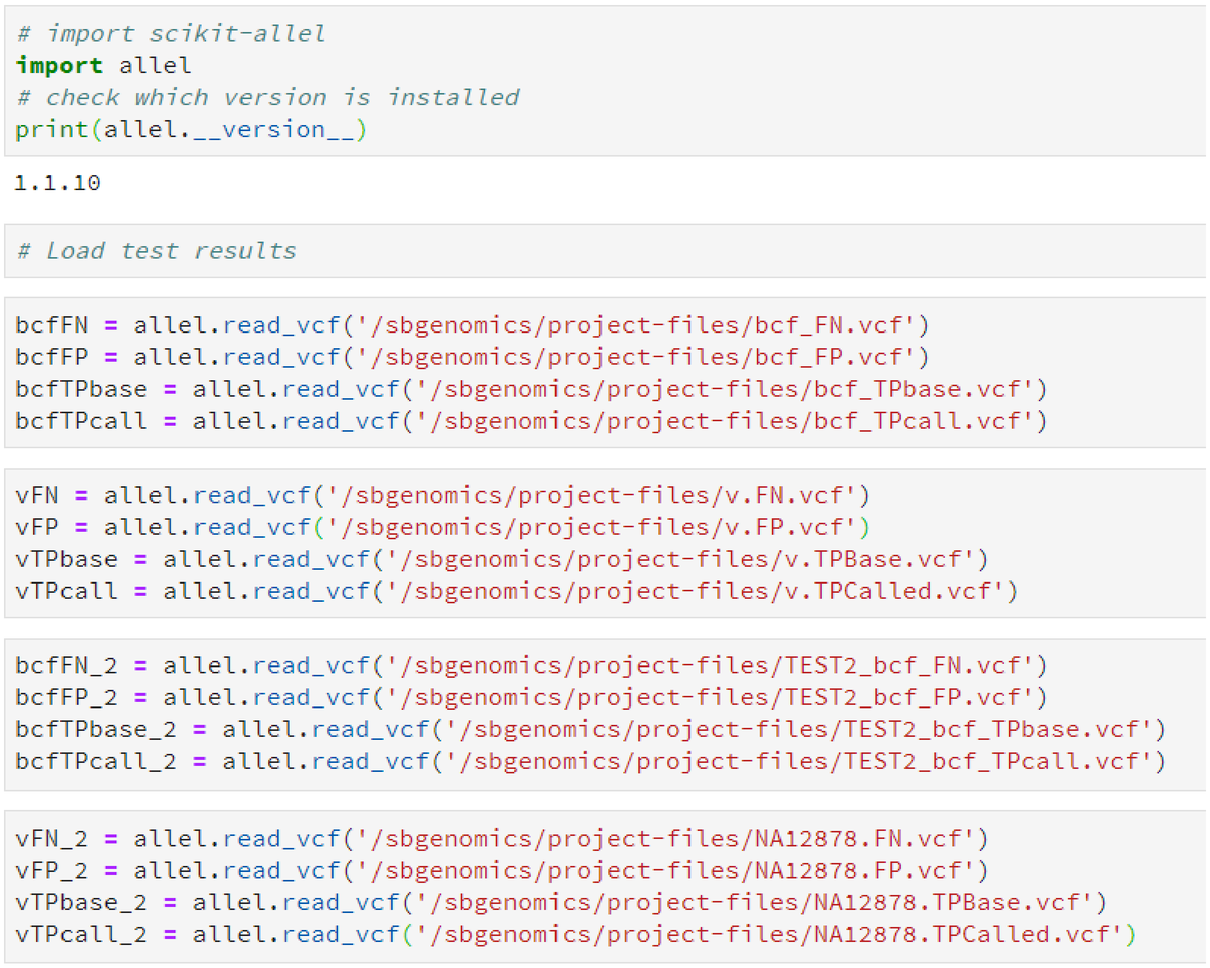
Output VCF files
TEST2_bcf_TPcall.vcf
TEST2_bcf_TPbase.vcf
TEST2_bcf_FN.vcf
TEST2_bcf_FP.vcf

Readme TXT file
_3_README.txt

Sites file — *No value*

Picture 5 – Task view of BCFtools isec at CGC platform

At CGC Platform there is also opportunity to analyze the results. *Data cruncher* provides us with *JupyterLab* and *RStudio* (Beta version). For these tests, analyze was done using python *JupyterLab*. At picture 6 there is shown *Jupyter Notebook*, where using *allel* library the tests output VCF files were load.



```
# import scikit-allel
import allel
# check which version is installed
print(allel.__version__)
```

```
1.1.10
```

```
# Load test results
```

```
bcfFN = allel.read_vcf('/sbgenomics/project-files/bcf_FN.vcf')
bcfFP = allel.read_vcf('/sbgenomics/project-files/bcf_FP.vcf')
bcfTPbase = allel.read_vcf('/sbgenomics/project-files/bcf_TPbase.vcf')
bcfTPcall = allel.read_vcf('/sbgenomics/project-files/bcf_TPcall.vcf')
```

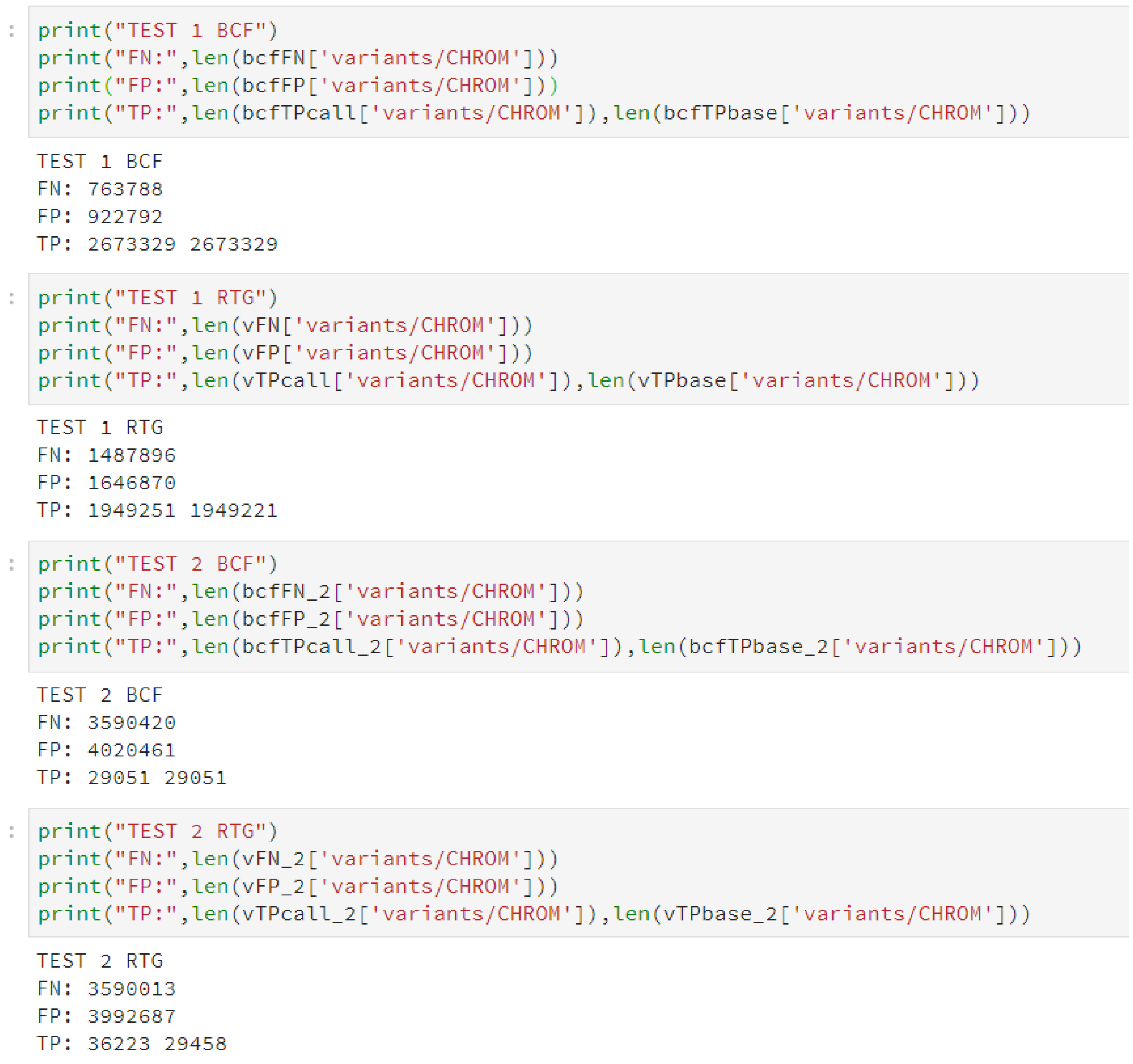
```
vFN = allel.read_vcf('/sbgenomics/project-files/v.FN.vcf')
vFP = allel.read_vcf('/sbgenomics/project-files/v.FP.vcf')
vTPbase = allel.read_vcf('/sbgenomics/project-files/v.TPBase.vcf')
vTPcall = allel.read_vcf('/sbgenomics/project-files/v.TPCalled.vcf')
```

```
bcfFN_2 = allel.read_vcf('/sbgenomics/project-files/TEST2_bcf_FN.vcf')
bcfFP_2 = allel.read_vcf('/sbgenomics/project-files/TEST2_bcf_FP.vcf')
bcfTPbase_2 = allel.read_vcf('/sbgenomics/project-files/TEST2_bcf_TPbase.vcf')
bcfTPcall_2 = allel.read_vcf('/sbgenomics/project-files/TEST2_bcf_TPcall.vcf')
```

```
vFN_2 = allel.read_vcf('/sbgenomics/project-files/NA12878.FN.vcf')
vFP_2 = allel.read_vcf('/sbgenomics/project-files/NA12878.FP.vcf')
vTPbase_2 = allel.read_vcf('/sbgenomics/project-files/NA12878.TPBase.vcf')
vTPcall_2 = allel.read_vcf('/sbgenomics/project-files/NA12878.TPCalled.vcf')
```

Picture 6 – Jupyter Notebook – output VCF files loading

Allel.read_vcf returns dictionary of arrays with all data. Access to each array is through the *keys* taken from the header line. To count number of samples in each file, it is enough to check the length of one of the arrays that consists data from each sample. At picture 7 it is shown how it was done for *test 1* and *test 2* by checking the length of *variants/CHROM* array of each file.



```
: print("TEST 1 BCF")
  print("FN:",len(bcfFN['variants/CHROM']))
  print("FP:",len(bcfFP['variants/CHROM']))
  print("TP:",len(bcfTPcall['variants/CHROM']),len(bcfTPbase['variants/CHROM']))

  TEST 1 BCF
  FN: 763788
  FP: 922792
  TP: 2673329 2673329

: print("TEST 1 RTG")
  print("FN:",len(vFN['variants/CHROM']))
  print("FP:",len(vFP['variants/CHROM']))
  print("TP:",len(vTPcall['variants/CHROM']),len(vTPbase['variants/CHROM']))

  TEST 1 RTG
  FN: 1487896
  FP: 1646870
  TP: 1949251 1949221

: print("TEST 2 BCF")
  print("FN:",len(bcfFN_2['variants/CHROM']))
  print("FP:",len(bcfFP_2['variants/CHROM']))
  print("TP:",len(bcfTPcall_2['variants/CHROM']),len(bcfTPbase_2['variants/CHROM']))

  TEST 2 BCF
  FN: 3590420
  FP: 4020461
  TP: 29051 29051

: print("TEST 2 RTG")
  print("FN:",len(vFN_2['variants/CHROM']))
  print("FP:",len(vFP_2['variants/CHROM']))
  print("TP:",len(vTPcall_2['variants/CHROM']),len(vTPbase_2['variants/CHROM']))

  TEST 2 RTG
  FN: 3590013
  FP: 3992687
  TP: 36223 29458
```

Picture 7 – Jupyter Notebook – counting number of TPs, FNs and FPs from each test

Results

Results of the tests are presented in *table 1* and *table 2*. Green cells mean that amount samples of this type was bigger for that tool and red that it was smaller.

Table 1 – results for test 1

	TEST I			
	TP		FP	FN
BCFtools isec	2673329		922793	763788
RTG vcfeval	1949251	1949221	1646870	1487896

In *test 1*, samples from father and son were compared. Therefore intersect of 50% is expected. Further samples were taken from the same source, so there should not be any difference in representation, but comparing different samples is more complex case. In *table 1* results show that *vcfeval* was closer to expected result.

Table 2 – results for test 2

	TEST II			
	TP		FP	FN
BCFtools isec	29051		4020461	3590420
RTG vcfeval	36223	29458	3992687	3590013

In *test 2* where both were HG001, but called from different callers, high amount of TPs is desirable. Here thanks to more complex algorithm, *RTG vcfeval* found more TPs than *BCFtools isec*.

Conclusions

VCF files are used for holding data about variants. They are divided into *header* part with all meta-information about file and *data* part with information about each variant. There are also many tools for comparing variants. One of the simple tool is *BCFtools isec* which directly compare variant position. More complex tool is *RTG vcfeval* which use much more complicated algorithm that use also reference sequence to compare variants at haplotype level.

From tests results it can be concluded, that *vcfeval* should works better in a cases of comparing variants from different callers and also when comparing variants in parent-child pair.

Sources

[1] Slides from lectures
[2] https://en.wikipedia.org/wiki/Variant_Call_Format
[3] https://www.ebi.ac.uk/training/online/course/human-genetic-variation-i-introduction-2019/variant-identification-and-analysis
[4] http://alimanfoo.github.io/2017/06/14/read-vcf.html
[5] https://samtools.github.io/bcftools/bcftools.html#isec
[6] https://github.com/RealTimeGenomics/rtg-tools
[7] https://github.com/genome-in-a-bottle/giab_latest_release
[8] https://gatk.broadinstitute.org/hc/en-us/articles/360035890951-Human-genome-reference-builds-GRCh38-or-hg38-b37-hg19
[9] https://www.nature.com/articles/sdata201625
[10] https://scikit-allel.readthedocs.io/en/stable/
[11] https://academic.oup.com/bioinformatics/article/34/24/4241/5026653
[12] Sequence Analysis,
Comparing Variant Call Files for Performance Benchmarking of Next-Generation Sequencing Variant Calling Pipelines
John G. Cleary, Ross Braithwaite, Kurt Gaastra, Brian S. Hilbush, Stuart Inglis, Sean A. Irvine, Alan Jackson, Richard Littin, Mehul Rathod, David Ware, Justin M. Zook, Len Trigg and Francisco M. De La Vega
1 Real Time Genomics, Hamilton 3240, New Zealand, and 4 San Bruno, CA 94066, USA. 2 NetValue Ltd, Hamilton 3240, New Zealand. 3 National Institute of Standards and Technology, Gaithersburg, MD 20899, USA. 5 Department of Genetics, Stanford University, Stanford, CA 94305, USA.