VCF2JSON: A desktop application for managing VCF files

## Abstract

The Variant Call Format (VCF) is widely used since the Next-generation sequencing (NGS) has become available. As the amount of VCF data continues to increase, it is more difficult for non-expert researcher and physicians to effectively manage VCF data. Since the management of VCF files requires professional bioinformatics, resulting in very few management tools of VCF files are developed. Here, we present VCF2JSON, a standalone, cross-platform and freely available desktop software, which is developed for researcher without bioinformatics skills. VCF2JSON transform VCF files into JSON files, which achieve zero loss. After that, researcher can use JSON tools to process data in VCF files much easier. Users can download the executable program and double-click it to run directly due to the fact that it is easy to implement and has a user-friendly graphical user interface (GUI). Beside, VCF2JSON can be used on a personal computer or servers and is freely available for download from <https://github.com/lyotvincent/vcf2json>.

## Introduction

The next-generation sequencing (NGS) has been widely used in different fields for research, such as hygienic genetics, clinical medicine, microbiology etc. The advent of NGS platform led undoubtedly to massive amounts of data. In order to process the data, a variety of file formats were born and the Variant Call Format (VCF) is one of the standard formats which has been developed with [1000 Genomes Project](https://en.wikipedia.org/wiki/1000_Genomes_Project)ii. VCF is very expressive, accommodates multiple samples, and is widely used in the community. However VCF is actually a text file formatiii, therefore, Processing VCF files require programming and bioinformatics skill. At present, one of biggest challenges is how to managing big VCF data.

There are some bioinformatics tools developed base VCF file, however, limited by the need for bioinformatics skill, fewer management tools are developed, especially based on big data mechanisms. Therefore, we are considering transform a VCF file to a more generic file that can be managed without bioinformatics skills, so that a large number of existing file management tools can be used to manage VCF data. In this case, the JSON (JavaScript Object Notation) format file is introduced. JSON is a lightweight data-interchange format which is easy for humans to read and write, and is also easy for machines to parse and generatei. Furthermore, JSON is a completely language-independent text format. These properties make the JSON format widely used in the computer science, which yield many JSON-based management tools. In summary, JSON is an ideal data transformation model

Since there is currently a gap in tools for transform VCF to JSON and in order to take advantage of JSON files to process VCF files, we present VCF2JSON, a standalone, cross-platform and freely available desktop software designed to transform VCF files to JSON files. Researchers input a VCF file and get a JSON file with zero data loss, and then they can import JSON file to MongoDB or other JSON-based database. That is to say, they can easily search variants with annotation and samples in MongoDB. In addition, they can also do complex statistics based on these data.

## MATERIALS AND METHODS

#### Framework of VCF2JSON

The base framework of VCF 2JSON is shown in figure 1. As described in figure 1, users can load a VCF file, may be uncompressed or gzip-compatible compressed file(\*.gz). VCF2JSON get parameters set by users from GUI, and then call python program. Finally, VCF2JSON generates a JSON file meet user requirements.

Figure 1

VCF Files(.vcf or .vcf.gz)

Python programing

Json Files

Electron GUI

#### Loading and processing a VCF

The VCF has become the standard format to store variation, which widely used in genomics projects. A VCF file divided into three sections, called a meta-information lines, a header line and data lines. Each meta-information line starts with “##” and must be key=value pairs. The header line starts with “#” and names the 8 fixed, mandatory columns. If genotype data is present, these are followed by a FORMAT column header, then an arbitrary number of sample IDs. The header line is tab-delimited. In data section, there are 8 fixed fields per record. All data lines are tab-delimited. In all cases, missing values are specified with a dot (‘.’). The first eight columns describe the chromosome(CHROM), position(POS), identifier tags in dbSNP varinant(ID), reference base of the variation(REF), alternate base of the variation (ALT), variation quality(QUAL), filter status(FILTER), variant annotation(INFO) and the last column is genotype fields which contain all samples.

VCF2JSON accept uncompressed or gzip-compatible compressed file(\*.gz) VCF files. Since each line of data in the VCF file is independent, when loaded a VCF file, VCF2JSON uses multiple processes to process the data, and the number of processes is equal to the number of CPUs. Each time 25k rows of data are read and sent to all child processes for processing. Consequently, when loading 25k rows of data and writing to the file, the memory will release the data so that VCF2JSON will not excessively occupy the system memory. In theory, VCF2JSON can process any size of data on any machine with any system.

When parsed a VCF file, three types of logic are applied. The first logic is processing header lines. VCF2JSON read the header lines and save it into JSON as value mapping to key named “header”. The reason for this is to provide conditions for the inverse transformation which is from JSON to VCF. It is easy for user to restore VCF file by extracting header information and the data lines from JSON file. The second logic is mapping multi-value. When a key mapping multiple values, VCF2JSON turn the value into array, in all cases, missing values are specified with a ("”) when values are string and specified with a (-1) when values are int or float. For example, the value of alternative allele field(ALT) is ("G,T”), then mapping the JSON value is ["G”,”T”,””] and the value of allele count in genotypes(AC) is 2, then mapping the JSON value is [2, -1, -1] in annotation field. The third logic is filter field modification. There are several types in filter field, VCF2JSON will get filter type and give it a “Filter\_” as prefix. As the result, each filter field is stored as a separate key in JSON. For example, If there is a row of data which filter field value is “s50”, then after transform to JSON, the value is “filter: {‘FILTER\_s50’: true, ‘FILTER\_pass’:false}”.

There is an example shown in Figure 2. As described in figure 2, the first line in JSON file saves the header information in the original VCF file. The next line is an array, each element represents a row of data in VCF.

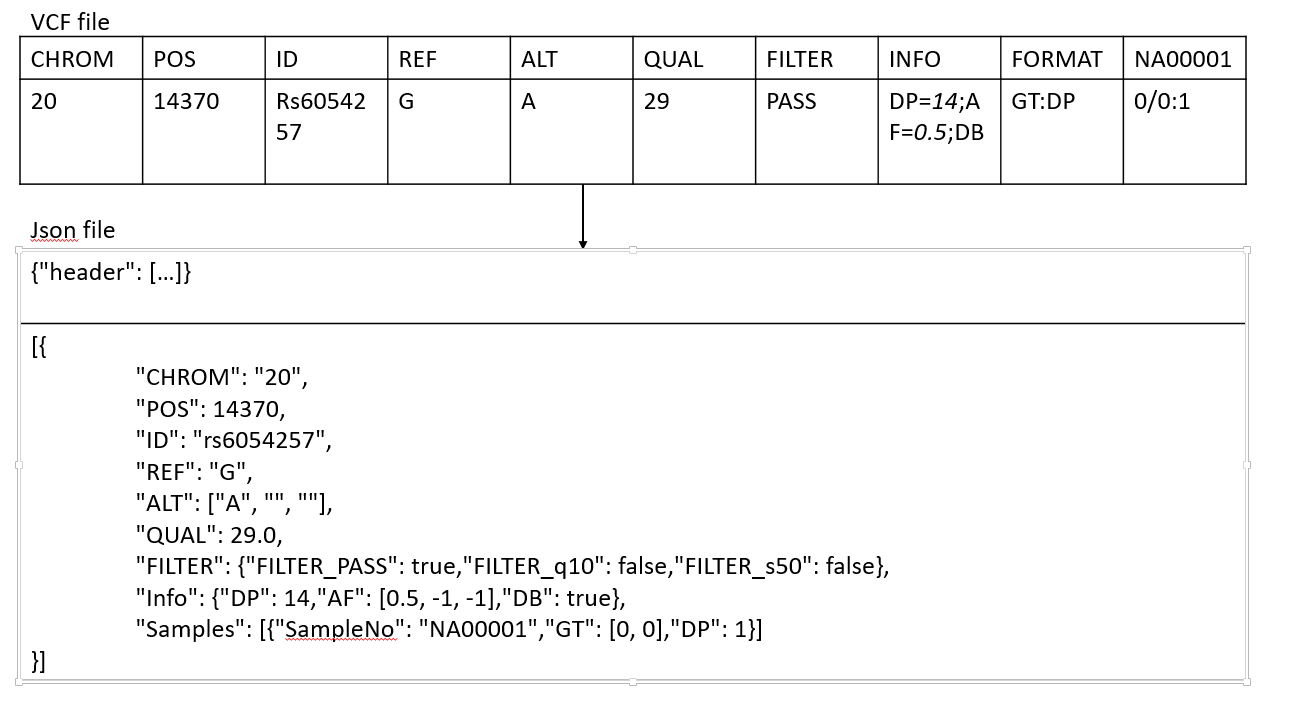


Figure 2

#### Implementation

The core of VCF2JSON was built with python program. The framework of the desktop application were designed and implemented using the electron and user interface was developed using html, css , jQuery and Bootstrap libraries. The table in preview page were produced by Datatable 1.10.18. The communication between python core and GUI was used zerorpc which is a light-weight, reliable and language-agnostic library for distributed communication between server-side processes.[[1]](#endnote-1)

## Results

#### Performance of VCF2JSON

To evaluate the efficiency and performance of VCF2JSON, we collect eight different files from the 1000 Genome Project ([www.internationalgenome.org](http://www.internationalgenome.org)) and test on it. VCF2JSON will create multiple processes based on the number of CPU cores when transforming. The transform results of eight different files are presented in Table 1. As the table 1 shown, the JSON file is larger (1-10x times) than the VCF file when transformed. The number of variants, annotations, samples in a VCF file is the principle factors that affect the transformation time. To evaluate whether the transform has achieved zero data loss, we count three groups of files, and got the number of variants, annotations, samples from VCF file and JSON file. As table 2 shown, the number of elements between VCF and JSON are all the same, therefore, the transform is achieved without loss data.

Table 1. Transform results for eight different VCF files

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| File | VCF Size(MB) | Variants | Samples | Transform time(s) on PC | JSON size(MB) |
| ALL.chr11.phase3\_shapeit2\_mvncall\_integrated\_v5a.20130502.genotypes.VCF | 42086.4 | 4045628 | 2504 | 14579 | 407449.6 |
| ALL.chr22\_GRCh38.genotypes.20170504.anno.VCF.hg38\_multianno.VCF | 11776 | 1100291 | 2504 | 4560 | 111513.6 |
| HG01920\_blood\_f.VCF | 1025.5 | 12344293 | 1 | 554.4 | 16486.4 |
| CEU.low\_coverage.2010\_10.indel.genotypes.VCF | 727.2 | 728075 | 60 | 139.5 | 3276.8 |
| clinvar\_20180603.VCF | 165 | 392746 | 0 | 9.6 | 277.6 |
| YRI.low\_coverage.2010\_10.indel.sites.VCF | 81.2 | 942693 | 0 | 34 | 307.2 |
| ALL.wex.union\_illumina\_wcmc\_bcm\_bc\_bi.20110521.snps.exome.sites.VCF | 33.3 | 657888 | 0 | 15.9 | 126.6 |
| ALL.wex.union\_solid\_bcm\_bc\_um.20110521.snps.exome.sites.VCF | 20.4 | 464700 | 0 | 10.8 | 88.6 |

Note. PC is a desktop running Ubuntu Linux 1.04 LTS, with Intel core i5 -8500 CPU 3.00GHz \* 6 and 32GB RAM

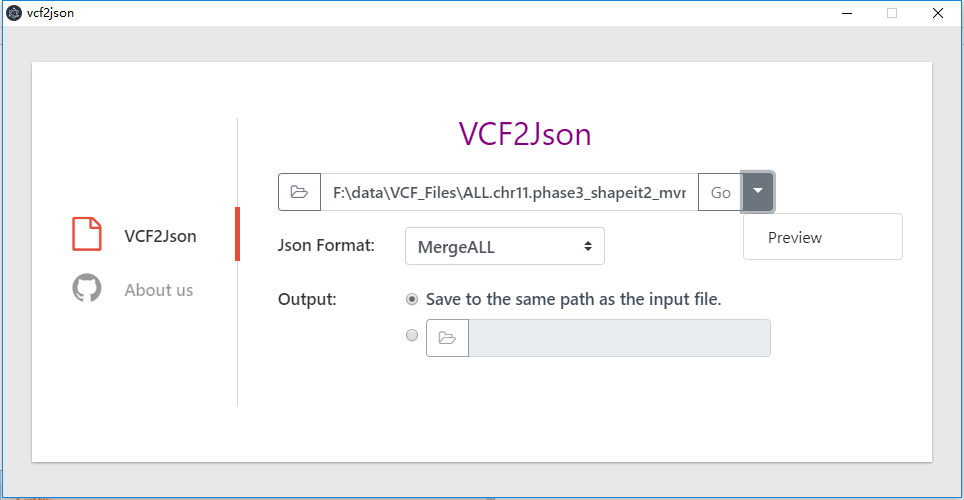
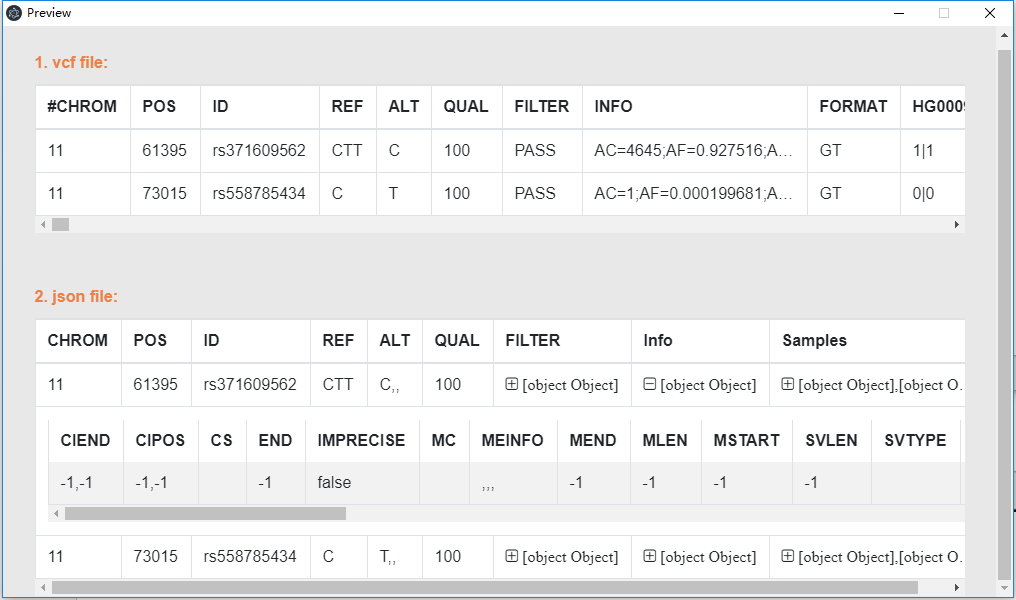
Table 2. Extracted the number of elements from three group of files

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Files** | **type** | **variants** | **annotations** | **samples** |
| ALL.chr11.phase3\_shapeit2\_mvncall\_integrated\_v5a.20130502.genotypes | .VCF | 4045628 | 109231956 | 2504 |
| .JSON | 4045628 | 109231956 | 2504 |
| HG01920\_blood\_f | .VCF | 12344293 | 259230153 | 1 |
| .JSON | 12344293 | 259230153 | 1 |
| CEU.low\_coverage.2010\_10.indel.genotypes | .VCF | 728075 | 7280750 | 60 |
| .JSON | 728075 | 7280750 | 60 |

#### Application of VCF2JSON: a utility case

First, the transform is initiated from the main page (Figure 3A), user open file dialog and select a VCF file, and then choose output directory. Furthermore, two JSON formats are available for the user to choose. The default is to merge annotation together. Of course, you can also choose another format which is no merging annotations. When user click “Go” button and wait a moment, they can get a JSON file at output directory. Second, by clicking the preview button, the user can preview the results of the transform, however only the first two lines can be previewed (Figure 3B).

Figure 3. (A) Main interface for VCF2Json (B) Preview of transform, only show first two lines in VCF file



Open file dialog and select a file

Select JSON format

Transform preview of first two rows in VCF

A

B

## Discussion and further direction

VCF2JSON is a standalone, cross-platform and freely available desktop application, which is developed for researcher without strong bioinformatics skills. In transform of eight different file, VCF2JSON demonstrated excellent performance due to using multiple processes to process data.

VCF2JSON is the first tool to transform VCF files. The main advantage is to convert the bioinformatics professional problem processing VCF files into a general computer problem processing JSON files. For example, after transform via VCF2JSON, the user can import the JSON file into MongoDB for quick query.

In the future, we will update VCF2JSON as a web application. When users uploaded VCF files through the browser, the server will get JSON file via VCF2JSON and then import the JSON file into MongoDB. In this way, it is more convenient for users to query VCF data.

[[2]](#endnote-2) https://www.JSON.org/[[3]](#endnote-3)[[4]](#endnote-4)

1. http://www.zerorpc.io/ [↑](#endnote-ref-1)
2. https://www.json.org/ [↑](#endnote-ref-2)
3. 10.Abecasis GR, Auton A, Brooks LD, et al. An integrated map

   of genetic variation from 1,092 human genomes. Nature

   2012;491:56–65. [↑](#endnote-ref-3)
4. VCF4.2.pdf [↑](#endnote-ref-4)