User manual

A quick manual for

123VCF 1.0

Highlights:

- Graphical user interface, easy to use, easy to understand
- Filtration orders can be defined by a simple text file
- Apply filters to VCF formatted annotated or non-annotated files (Single and multi-samples VCF files)
- Operable in Linux, macOS, and Windows platforms using just one .jar file
- Can be used in research and clinical settings
- Lightweight Java-based tool which can handle even the huge VCF files on an ordinary desktop computer

Contents

Introduction

123NGS has great aspirations for fulfillment. The ease of next-generation sequencing (NGS) data analysis for every specialist is its main goal. Now, we introduce 123VCF, a tool that tries to meet all the user's needs in the filtration process of VCF files.

How to download and run 123VCF?

The latest version of 123VCF is available on the 123NGS page at SourceForge:

https://project123vcf.sourceforge.io

123VCF is written in java and can launch in Linux, Windows, or macOS using only one .jar file. In other words, 123VCF is cross-platform. The only requirement to run the software is to install java runtime environment (JRE) version 1.8 on your operating system.

To run 123VCF, at the command line of your operating system, change the working directory to the extracted 123VCF folder, and then type the below command and press Enter simply.

```
#    Java –jar 123VCF.jar
```

We prepared a runner bash script to make the executing software even easier in the windows operating system.

Input

123VCF accepts annotated or not-annotated VCF files in compressed (vcf.gz, vcf.bz2, and vcf.zip) or non-compressed formats. It also needs a filtration order file which explained below.

Preparing filtration order file

123VCF needs a filtration order file that is a plain text file with such rules. Each filter should be defined in a line like the sample below:

Filter's name, Filter's scenario, Filter's operator == Filter's command1, Filter's command2 …

The filter's name is the exact name of target annotation in the variant annotated file on which the filter should take action. For the Format field annotations like GT, DP, and GQ, which all of them are sample-based, the filter's name should be like FormatGT, FormatDP, and FormatGQ, respectively (Take a look at the filter definition examples section). For the Info field annotations, the name of annotation is sufficient.

Filter's scenario tells 123VCF the type of filter, whether it can be **numeric**, **text_contain,** or **text_exact**.

- ✓ **Numeric** should be used for numerical annotations. Remember that the BED-based filtration should be defined in this type of scenario.
- ✓ **Text_contain** should be used for the text-based annotations, and it finds the filter's command(s) when the filter's name variant annotation just contains (not whole word match) the specified filter command. This filter type isn't case sensitive.

- ✓ **Text_exact** should be used for the text-based annotations. It finds the filter's command(s) when the filter's name variant annotation completely matches the specified filter's command (whole word match). Remember that the FormatGT(genotype) should be defined in this type of scenario. This filter type isn't case sensitive.

The filter's operator tells 123VCF to include or exclude the variants compatible with the filter's command(s). Possible values for this part of the filter definition are **include** or **exclude**.

Filter definition examples

For example, if you want to define several numeric filters like filters for gnomad population database, CADD, and DANN prediction tools, it should be defined in the filtration order file like the examples below: (pay attention to the separators like commas, angle brackets, and double equals position)

gnomad211_genome, **numeric, include == <= 0.05**

CADD16, **numeric, include == > 14, = 10**

DANN, **numeric, exclude == <= 0.9**

If you want to define a filter for a text-based annotation, write your filters like the example below:

CLNSIG, **text_contain, exclude == benign**

In addition, to define a genotype filter for a multi-sample VCF file the filter definition should be like this:

FormatGT, text_exact, include == sample1_Name = Het, sample2_Name = Het, sample3_Name = Hom

If the user wants to apply Format field filters to all the samples which are in the VCF file, the filter should be defined like this:

FormatGT, text_exact, include == ALL = Hom

Also let's take a look at two numeric Format field filters:

FormatDP, numeric, include == ALL > 8

FormatGQ, numeric, include == ALL = 99

To learn more about how to define the filters for 123VCF, take a look at "**Filters"** folder which contains a couple of filter order files.

Output

Remained variants after the execution of defined filters based on the output's specified name file by the user can be compressed (vcf.gz and vcf.bz2) or non-compressed. 123VCF also make tab separated value (TSV) text files from the remained variants which could be very helpful for the users to visualize and do the downstream surveys. There will be two TSV files, one is normally TSV file of variants and the another one which contains same feature variants next to each other, will be generated if the user provides a BED-based filter in the filter order file.

Conclusion

Up to now, 123NGS has been developed some of the easy to use and reliable tools in different parts of NGS data analysis pipelines, and this way continues. To help and encourage us, give us your feedback about 123VCF here:

https://sourceforge.net/p/project123vcf/discussion/

Sincerely,

123NGS team

Changes

- Version 1.0
  Initial release, October, 2022