# 1. About

## 1.1. VICTOR

[Back to top]

VICTOR (Variant Interpretation for Clinical Testing Or Research) is a pipeline for the analysis of next-generation sequencing data starting from a multi-sample raw VCF file without decomposing, normalization, or annotation. It can be used for disease gene discovery research or clinical genetic testing. It is designed to be scalable to whole genome sequencing (WGS) of a large sample of individuals that is typical of a research on a complex disease. The downloadable package includes PERCH, VANNER, utility programs, script templates, and data files. It is mostly a self-contained package, whereby the requirement for third-party applications is minimal (see the "USE NOW" page for details), and all necessary databases are already included. It provides data updates on a monthly basis.

This pipeline runs on Linux or Mac OS X. It supports parallel computing. The primary interface of this pipeline is a SLURM script template named slurm.all_steps. SLURM is a job scheduler for Linux systems that is used by many of the world's supercomputers and computer clusters. This script template can also be converted to work for other job schedulers such as PBS or MOAB. Alternatively, it can be used as a Bash script without a job scheduler. Using this template, users can modify analysis parameters and input files specific to their research project and submit a job to a computer cluster. This script supports re-analysis from the middle of the pipeline if previous steps are successful. Users can modify parameters in the script, select the steps to be executed, and then submit a job to the cluster again. The script will automatically write log files sequentially.

This webpage describes the features common to all included programs and the usage of the utility tools. For the details of PERCH and VANNER, please be referred to the corresponding user Manual.

**What VICTOR does:**

1. Conducts genotype-, variant-, and sample-wise quality control of data. Please click here for a list of methods.
2. Performs principle component analysis and automatically adjust for population structure in association tests.
3. Calculates relatedness, remove correlated individuals, and detect pedigree structure errors.
4. Annotates allele frequency, deleteriousness, functional consequence, regulatory regions, microRNA-binding sites, protein domains, and clinical significance of variants.
5. For gene discovery research on a Mendelian disease, complex disease, or the disease that is hypothesized to be caused by *de novo* mutations, it performs variant prioritization, gene prioritization, and gene set analysis. It works for different study designs including case-control, case-only, extended pedigrees, trios, or mixed. Statistics include linkage analysis, linear or logistic regression, sum squared U, Fisher's exact, rank sum, etc.
6. For clinical testing, it integrates deleteriousness, co-segregation and association test to calculate a posterior probability of pathogenicity for a variant. It implements the components in the IARC guidelines that are not available in other software. You can also convert the VICTOR outputs to a strength of evidence (supporting, moderate, strong, very strong) for integration with the ACMG guidelines.
7. It reports incidental findings for the return of results to study participants.

**What are included in the package:**

1. A variant/gene/gene set prioritization and variant classification software bundle named PERCH
2. A variant annotation software bundle named VANNER
3. A set of utility programs
4. A set of scripts and SLURM templates for submitting jobs to a computer cluster
5. The reference sequence GRCh37, GRCh38, and hg19

6. dbSNP 150
7. Ensembl transcript database release 92 for GRCh38 and 87 for GRCh37
8. RefSeq transcript database 2018-05-21
9. A light-weighted Ensembl database containing principle transcripts from APPRIS
10. A light-weighted RefSeq database containing principle transcripts from APPRIS
11. Database of discrepancies between RefSeq sequence and reference sequence
12. Micro-RNA binding sites predicted by Target Scan 7.1
13. Splice-altering (intronic or exonic) SNVs predicted by dbscSNV
14. Ensembl regulatory build motif features
15. A converted GeneMania gene network ready to be used by PERCH
16. PROVEAN scores for InDels observed in public databases
17. The maximum allele frequencies obtained from UK10K, gnomAD, GoNL, and 1k Japanese
18. BayesDel scores for all possible SNVs in the entire human genome
19. Maximum BayesDel scores among all possible SNVs for each gene
20. ClinVar variants, excluding those with conflicting reports
21. Databases of mappability (DukeExcludable and DacExcludable)
22. InterPro protein domain database
23. Several gene panels for reporting incidental finding
24. Several pathway databases for gene set analysis
25. ExAC non-TCGA v1

**Features in addition to those inherited from PERCH or VANNER:**
1. Supports parallel computing.
2. Supports re-analysis from the middle of the pipeline. It writes logs sequentially.
3. Supports multiple genomes including hg19, GRCh37, and GRCh38.
4. Automatically determines the VQSLOD cutoffs for SNVs and InDels separately.
5. Automatically adjusts for population structure detected from a principle component analysis.
6. Automatically removes correlated individuals that have higher missing rates.
7. Provides monthly database updates.

**Version checking**
The version string of this software package is a version number and a build date. If it is a beta version, the version number is followed by the word "beta". All programs have the same version. Program version and data version are separate; sometimes you only need to upgrade the programs but not the data. The (--version) option of any program will check for new versions for both programs and data through the Internet. This option will not send out any information.

## 1.2. Options

[Back to top]

In general, the usage of program options follows these rules:
1) Options with 2+ arguments must be used like this: "--option arg1 arg2";
2) Options with only 1 augment can be used like this: "--option=arg" or "--option arg";
3) You can omit a boolean argument that is positive, ie: "--option=yes" is equal to "--option";
4) It is better to always use "--option=arg" whenever possible, which is more readable;
5) If the argument is an array, you can replace the default by "--array-option=v1,v2";
6) You can clear the default by assignment to an empty array "--array-option=";
7) You can keep the default and insert additional items by "--array-option+=v1,v2";
8) The order of different options does not matter. For example, "-a -b" is equivalent to "-b -a";
9) The order of the same option matters. The last one overrides the others. E.g., "-a=yes -a=no" is "-a=no";
10) You cannot merge multiple single-letter options into one: "-a -b" cannot be written as "-ab".

Be careful about array-type arguments: the correct way to assign two values is "--array-option=v1,v2" but not "--array-option=v1 --array-option=v2". The latter will first assign v1 to the array, then assign v2 and get rid of v1, leaving only one element (v2) in the array.

All escape sequences in program options and arguments will be replaced. The following sequences are recognized: \\ (backslash), \a (alert), \b (backspace), \f (form feed), \n (new line), \r (carriage return), \t (horizontal tab), \v (vertical tab), and \xHH (byte with 2-digits hexadecimal value).

The (-h) or (--help) option will print a short help text for the program. In the help text, the description of each program option follows the convention of "ProgramOption ArgumentDataType Function {CurrentValue}".

Anything within a pair of square brackets is optional. Things within curly brackets are the current value after setting by factory defaults, configure files, and program options. To make the help text more readable, please set the terminal window size to at least 160 columns.

ArgumentDataType is represented by a letter or a string as shown below:
  D or DBL -- a floating point number
  I or INT -- an integer without commas for thousands (1234 but not 1,234)
  B or BOOL -- a case-insensitive boolean value (0/1/no/yes/n/y/false/true/f/t)
  C or CHAR -- a character
  S or STR -- a string, escape sequence allowed (e.g., "hi\n" = hi\\n = $'hi\n')
  F or FILE -- a filename, may include relative or absolute path, environment variables allowed
  P or PATH -- a path, relative or absolute, environment variables allowed, better ends with /
  FLD or FIELD -- a field number. The first field is 1, and so on. The last field is -1, second last is -2, and so on.
  FLDs or FIELDs -- an array of field numbers divided by a comma
  Ss or STRs -- an array of strings separated by a comma
  Fs or FILEs -- an array of filenames separated by a colon

If specified, there may be a strict requirement for the format and the number of arguments. For example, "D,D,D" refers to exactly 3 floating point numbers separated by a comma, such as the --penetrance arguments. ArgumentDataType could be followed by an integer to help the writing of the descriptions, such as S1,S2,S3 for the --xct-pfx option, where S1 is ExAC file prefix, S2 is cases' qc.log file prefix, S3 is cases' coverage file prefix.

### 1.3. Configure file

[Back to top]

A Configure File is another way to pass parameters to programs besides program options. Although being optional, using a Configure File is more convenient than program options, especially for the parameters that need to be the same for all VICTOR programs. The programs read two Configure Files if they exist. The first is /path/to/VICTOR/etc/par.txt, which can be used to set up parameters that are universal to all users. The second is ./par.txt, which should be used to specify parameters in the current analysis. The former overrides default values; the latter overrides the former; and command line options override all of them.

In this file, one row is one option. The format is "--option=argument". Space is not allowed in the arguments. Anything after the argument will be omitted. Multiple consecutive spaces or tabs are treated as one delimiter. Lines starting with a # are comments. Comments and blank lines will be omitted. The first row of the file has a special content (see the examples below), so that the programs will not accidentally read a wrong file.

Not all program options can be set in a Configure File. Below is a list of supported options:

```
--col-one Ss       Header of the first column {"#CHROM","#Chrom","CHROM","Chrom","#CHR","#Chr","#chr","Chr","CHR","chr"}
--col-func S       Header of the column for functional consequence {Func_Type}
--col-gene S       Header of the column for gene symbol {Func_Gene}
--col-fdet S       Header of the column for functional details {Func_Detail}
--info-func S      The name of the INFO sub-field for functional consequence annotation {vAnnGene}
--gnuplot S        The command for gnuplot 4.6.0 or later {gnuplot}
--gdb S            The gene database (refGeneLite/refGeneFull/ensGeneLite/ensGeneFull) {refGeneLite}
--filt-vqs-nan B   Exclude variants if VQSLOD is missing {no}
--filt-vqs-snv D   Exclude variants if VQSLOD<D (-inf = no filtering) for SNVs {-5.368}
--filt-vqs-indel D Exclude variants if VQSLOD<D (-inf = no filtering) for InDels {-4.208}
--filt-miss-rate D Exclude variants if missing rate in cases or controls is > D (1 = no filtering) {0.01}
--filt-filter Ss   Exclude variants if FILTER is not one of the Ss {.,PASS}
--filt-QD D        Exclude variants if QD (Qual. By Depth.) < D (0 = no filtering, GATK default = 2)  {0}
--filt-MQ D        Exclude variants if MQ (Mapping Quality) < D (0 = no filtering, GATK default = 40) {0}
--filt-FS D        Exclude variants if FS (Fisher Strand)   > D (0 = no filtering, GATK default = 60) {0}
--filt-HS D        Exclude variants if HaplotypeScore       > D (0 = no filtering, GATK default = 13) {0}
--filt-MR D        Exclude variants if MQRankSum            < D (0 = no filtering, GATK default = -12.5) {0}
--filt-RP D        Exclude variants if ReadPosRankSum       < D (0 = no filtering, GATK default = -8) {0}
--hard-filter B    Exclude variants by hard filters. Will set thresholds with GATK defaults if not set. {No}
--HardFiltIfNoVQ B Exclude variants by hard filters if there's no VQSLOD. Will set thresholds with GATK defaults. {yes}
--filt-DP I        Exclude genotypes if DP<I (0 = no filter) {10}
--filt-GQ I        Exclude genotypes if GQ<I (0 = no filter) {40}
--filt-MaxAF D     Exclude variants if MaxAF > D (0 = no filter) {0.01}
--filt-SplAF D     Exclude variants if SplAF > D (0 = no filter) {0}
--filt-FdrAF D     Exclude variants if FdrAF > D (0 = no filter) {0.05}
--filt-del D       Exclude variants if BayesDel<D (-inf = no filtering) {-0.0592577}
--prevalence D     Prevalence {0.025}
--penetrance D,D,D Penetrance {0.02,0.1,0.5}
--include Fs       Restrict analysis to regions     in FILEs. Use 2+ files to define intersection. {}
--exclude Fs       Restrict analysis to regions not in FILEs. Use 2+ files to define union. {}
--lof-only B       Restrict analysis to loss-of-function variants only, BayesDel still applies {no}
--lof-no-del B     Restrict analysis to loss-of-function variants only, BayesDel not considered {no}
```

```
--rm-ind Fs        Remove individuals listed in file(s) Fs {}
--vc B             Run mode is Variant Classification {no}
--no-web B         Do not check version from web {false}
```

Below is an example of the /path/to/VICTOR/etc/par.txt:

```
VICTOR_parameters_1.0                    << Do not delete or modify. This line prevents reading a wrong file.
--gnuplot=/opt/gnuplot/4.6.0/bin/gnuplot   << This is necessary if gnuplot 4.6.0 or above is not included in $PATH.
```

Below is an example of the ./par.txt:

```
VICTOR_parameters_1.0                    << Do not delete or modify. This line prevents reading a wrong file.
--prevalence=0.025                       << prevalence
--penetrance=0.02,0.1,0.5                << penetrance
```

### 1.4. Pipe [Back to top]

Most VICTOR programs read a data file from the standard input, do analyses, then write a modified data file to the standard output. Therefore, you can use the Unix pipe ("|") to run multiple programs sequentially. This feature can help you reduce CPU usage time and save disk spaces.

### 1.5. Standard error [Back to top]

Analysis logs, messages, and runtime errors are normally written to the standard error (StdErr) output. However, the usage of Unix pipe and multi-processing parallelism (e.g., using GNU parallel to run multiple instances simultaneously, usually separated by chromosomes) will make the StdErr outputs between programs intermingle with each other. To solve this problem, please set an environment variable STDERR_MUTEX to a non-empty string. The provided SLURM script templates already implemented this feature. If you use the script template to run an analysis, you don't have to set the variable separately.

Messages written to the standard error output by VICTOR programs are one-liners, i.e., one message per line. Each line starts with a number, which is the process ID of the program. By this number, you can see to which program a message belongs. You can easily check for errors and warnings using the Linux command "grep". To check for errors: grep -i 'error\|exception\|slurmstepd\|NODE_FAIL\|segmentation' <your StdErr file>
To check for warnings: grep -i 'warning' <your StdErr file>

### 1.6. Input files [Back to top]

Unless otherwise stated, input files for this software have columns divided by a space or a tab. Multiple successive delimiters are not treated as one. Lines starting with a '#' are comments and will be ignored. The reading of input files is robust to Linux/Mac/Windows/mixed line breaks. It is also robust to the missing of a line break at the end of the last line. Programs will stop reading a file at the first blank line; therefore, you can write something useful to you in the file after a blank line. The input file does not need to be uncompressed if it is a .gz or .bz2 because the programs can read them directly. Both bgzip and gzip compressions are supported. You don't even need to type ".gz" or ".bz2" in the command, as the programs will first look for the uncompressed file, then file.gz, followed by file.bz2.

#### 1.6.1 Sample File

A Sample File contains ID, sex, outcome and covariates for all samples, including unrelated cases and controls for association tests, duplicated samples for quality control, individuals from pedigrees for linkage analysis or de novo mutation detection, and any other samples that have been sequenced and could be used for quality control. This file has a header row. The first column should be named "SeqID", which contains the Sequence ID for each sequenced sample in a Genotype File, i.e., the ID in a VCF header row. The second column is "Sex" with contents of female or male; other strings represent unknown sex. If you don't know the sex of some samples, set them to unknown; don't set unknown sex as males or females because this will lead to wrong allele frequency calculations for chromosome X or Y. The third column is the outcome variable, which can be an affection status (unaffected or affected; other strings are unknown affection status) or a quantitative trait (if there is any numbers not equal to 0/1/2). Sex and Affection Status are case-insensitive. Samples not to be used in a case-control association test should have an unknown affection status. They are in this file just to provide the sex information. Columns starting from the fourth are optional. They are covariates for association tests and quality control. Missing values can be . or unknown. Covariates can be string-type or numeric-type but not mixed, i.e., a covariate cannot contain both numbers and non-missing strings. The name of a covariate cannot start with "_PC_" or "_iPop", which is reserved by VICTOR to store the principle component analysis

results. If you want to provide the population origin information instead of performing a PCA, you can add a string-type covariate with the header "pop". A Sample File may contain more or fewer people than the Genotype File. Only the overlapping samples will be analyzed. In an association test, samples with a missing value for the outcome or any covariate will be omitted. Lines in this file need not be sorted nor in the same order as the Genotype File. Below is an example, where the columns are separated by multiple spaces for web display purpose only. In a real file they should be divided by one tab.

```
SeqID    Sex      Disease     Pop
ped1_i1 male     unknAff     Eur
ped1_i2 female   unknAff     Eur
ped1_i3 male     affected    Eur
ped1_i4 female   unknAff     Eur
random  unknSex  unknAff     SAS
HG00096 male     unaffected  Eur
HG00097 female   unaffected  Eur
HG00099 female   unaffected  Eur
HG00100 female   unaffected  Eur
```

### 1.6.2 Pedigree File

A Pedigree File contains all pedigrees for linkage analysis by vSEG or de novo mutation analysis by vAAA. This file is similar to PLINK .fam, but is more robust, flexible, and informative. If you already have a PLINK .fam file you can directly use it as a Pedigree File for PERCH. It has 6 columns corresponding to PedigreeID, IndividualID, FatherID, MotherID, Sex, and AffectionStatus. It has a header line as shown in the example below. Optionally, it can also have the 7th column for LiabilityClass. Pedigrees should not have any consanguinity loop or marriage loop. If your pedigree has loops, you can manually break the loops by founderizing a minimum number of individuals while maintaining the relationships among affected individuals as much as possible. My program "pedpro" has the function to do so automatically. You can use that program to break the loops before analysis by vSEG. There is no other restriction on pedigree structure. For variant classification, Pedigree File should indicate who is the proband in each pedigree, which can be done by adding the phrase "(proband)" at the end of the IndividualID. Beware that the word "proband" may means different concepts to different people -- to a genetic counselor it may be the first person requested for a genetic test, who may or may not be a carrier of the mutation; while to a statistician it is the first person in the pedigree who tested positive for the mutation. If you don't know how to set a proband, then just leave it to the program; vSEG will automatically choose a proband in a conservative way. You can use the traditional coding of Sex (1 for male, 2 female, 0 for unknown) and AffectionStatus (1 for unaffected, 2 for affected, 0 for unknown) or use the same coding as in a Sample File. Liability Class is an integer starting from 1. If an individual was sequenced, the IndividualID should match with the header of the VCF file. This makes the PLINK .fam file a bit more difficult to prepare. However, a program named vConvertPed in the VICTOR package can help you create this file from a PERCH-format Pedigree File, which is more convenient in some situations. Below is an example. Here columns are separated by multiple spaces for web display purpose. In a real file they should be divided by exactly one tab.

```
PedID IndID    Father   Mother   Sex     Aff
ped1  ped1_i1  0        0        1       2
ped1  ped1_i2  0        0        2       1
ped1  ped1_i3  ped1_i1  ped1_i2  1       2
ped1  ped1_i4  ped1_i1  ped1_i2  2       2
```

### 1.6.3 Seed File

A Seed File lists the seed genes for guilt-by-association analysis by gnGBA. The first column of this file is a disease name and the second column gene symbols. Space is not allowed in the disease name or the gene symbol. Make sure that gene symbols are NCBI Official Symbols, not full names or synonyms or GeneIDs. This file does not have a header row. Below is an example:

```
Disease Gene1
Disease Gene2
```

### 1.6.4 Annotation File

An Annotation File is a database file for variant annotation by vAnnDel. The information for annotation could be deleteriousness scores, dbSNP IDs, allele frequency, etc. Normally you don't need to create this file because these information are already included in the VICTOR package. If you have allele frequency data specific to the study population, you can make an Annotation File with allele frequency for annotation. The format of this file: 1) Lines starting with ## are comments;

2) The first row is the header row. Contents of this row will be used as headers in the output;
3) The first 4 columns are #CHROM, POS, REF, and ALT;
4) There is no limit to the number of columns;
5) Variants with multiple alternative alleles are split into separate lines;
6) Lines are sorted by #CHROM, POS, REF and ALT;
7) The file is compressed by bgzip and indexed by tabix;
8) Column header cannot be MaxAF, which is already reserved for internal usage by VICTOR.

#### 1.6.5 Cohort File

If your samples were target-enriched by different reagents or sequenced by different platforms or with different depths, then you need to generate a Cohort File (cohort.txt) and set --cohort=cohort.txt for QC1 in slurm.all_steps. Here a cohort is a sample set that is target-enriched by the same reagent and sequenced by the same platform with the same depth. A Cohort File has 2 columns, SeqID and CohortID, delimited by a tab. The CohortID should not contain white spaces. It can contains any number of alphanumeric or other characters. Samples with an unknown CohortID (empty or "." or "unknown") will be excluded from missing rate calculation by vQC. Below is an example. Here columns are separated by multiple spaces for web display purpose. In a real file they should be divided by one tab.

```
SeqID   Cohort
ped1_i1 MyStudy
ped1_i2 MyStudy
ped1_i3 MyStudy
ped1_i4 MyStudy
HG00096 1000Genomes
HG00097 1000Genomes
HG00099 1000Genomes
HG00100 1000Genomes
```

### 1.7. Genome [Back to top]

VICTOR provides data files for hg19, GRCh37, and GRCh38. Because there may be multiple genome databases stored in VICTOR's data folder, users need to tell the programs which one to use by either one of the two methods. 1) Use the --genome option either in the command line or a par.txt. 2) Place the current directory within a full path that has the name of the genome, for example, /path/to/GRCh37/PI/project_A/analysis_1. The genome name does not have to be the whole folder name; it can be something like "assembly_GRCh37". But if the full path contains multiple genome names, such as "GRCh38_liftover_from_GRCh37", then the programs cannot infer the genome. In this case, you have to use the --genome option.

## 2. Scripts

### 2.1 slurm.all_steps [Back to top]

This is a SLURM script template for you to submit jobs to a computer cluster. SLURM scripts are basically Bash scripts with additional SLURM parameters, which are the lines starting with "#SBATCH". You can easily convert a SLURM script to a PBS/MOAB script or a plain Bash script.

This script will submit one job and perform parallel analyses on one computer node. It provides a mechanism to save log files sequentially when you execute this script multiple times. For this purpose, you need to use --array in submitting the job (e.g.: sbatch --array=1 slurm.all_steps). Below takes --array=2 as an example. When a job starts running, a file named "slurm.all_steps.run_2.start" will appear in the folder. When the job finishes successfully, another file named "slurm.all_steps.run_2.stop" will appear. Standard outputs and errors will be written to "slurm.all_steps.run_2.stdout" and "slurm.all_steps.run_2.stderr", respectively. The script itself will be saved to slurm.all_steps.run_2.script. The local parameter file par.txt will be saved to slurm.all_steps.run_2.par. The version of VICTOR will be written to slurm.all_steps.run_2.version.

This script
1. Conducts genotype-wise, variant-wise, and sample-wise quality control of data (see 4.1 below).
2. Determines VQSLOD cutoffs for SNVs and InDels, separately.
3. Performs principle component analysis and adjust for population structure in analyses.
4. Calculate relatedness. Selects one sample from each related group by minimizing missing rate.
5. Detect pedigree structure errors.
6. Annotates functional consequence, allele frequencies, and the deleteriousness score BayesDel.
7. Tests whether cases and controls are comparable by counting the number of rare variants per sample.

8. Performs PERCH analysis, which may be a burden test, variance-based association test, linkage analysis, gene prioritization, variant prioritization, gene set analysis, or reporting incidental findings.

### 2.2 slurm.step2 [Back to top]

This script conducts the second step of slurm.all_steps. It is different from slurm.all_steps in that it will submit multiple jobs to a computer cluster. Each job performs an analysis on one chromosome. This is useful when you need to do phasing, which is time consuming.

### 2.3 victor.sbatch [Back to top]

This is a script to run the "sbatch" command to submit a job to a computer cluster. Instead of using sbatch, this script has the advantage of working in harmony with slurm.all_steps. Since slurm.all_steps hijacks the array number to make sequential logs if you run slurm.all_steps for multiple times, victor.sbatch makes sure that you do use the --array option and the array number has not been used before. It also has the advantage of saving the job ID to a file named slurm.all_steps.account, so that you can easily keep track of the job or cancel the job. victor.sbatch can also work for other slurm scripts.

## 3. Utility tools

### 3.1. vBED and vBED2 [Back to top]

It reads coverage data output from samtools or GATK, calculates some statistics, and then write a BED file for well-covered regions. Here the statistics is the proportion of a region above a certain read depth (--dp) in a minimum percentage of the samples (--pc). The regions could be targeted regions (--target) or genes (--gene). After calculating these statistics, it combines the data and write a BED file of well-covered regions, which have read depths above a threshold (--cutoff). If the (--log) option is set, it will also write to a file the number of samples well-covered for each locus.

The input files could also be BED files (--format=bed). In this case, the coverage statistics cannot be calculated, but it still integrates the data and write a merged BED file for downstream analysis. The merged regions are those covered by a minimum percentage of the input BED files (--pc). If you have only one input BED file, the output should have the same regions as input, but you can use it to see how many basepairs are included in the BED file by this command: "vBED --format=bed input.bed >/dev/null".

vBED2 is a faster version of vBED, but some options are not implemented.

### 3.2. vConvertVCF [Back to top]

vConvertVCF modifies VCF files to be read by KING or PLINK 1.9. If you provide multiple VCF files, they will be concatenated. You need to provide a Pedigree File (--ped) and/or a Sample File (--spl). Sequenced individuals not in these files will be dropped; individuals in these files but not in the VCF can be added (--add); VCF headers will be changed to <FID><delimiter><IID> for samples in the Pedigree File, or <SeqID><delimiter><SeqID> for those in the Sample File. The delimiter can be customized (--id-delim). If a sample is in both Pedigree File and Sample File, the former has the priority. It will also write a PLINK .fam file to replace the one created by PLINK (--fam). Please note that the order of individuals in this .fam file is not the same as that in the output VCF file. So to use this .fam file, you need to enable --indiv-sort for PLINK:

```
plink_1.9 --vcf vConvertVCF.output.vcf --id-delim : --indiv-sort file vConvertVCF.output.fam --make-bed
mv vConvertVCF.output.fam plink.fam
```

### 3.3. vConvertPed [Back to top]

VICTOR and some other programs support a Pedigree File in PLINK or LINKAGE format, where the Individual IDs should match with the headers of a VCF file. However, sometimes this format is not ideal. You may already have an old pedigree file with different Individual IDs. You may also want to separate Individual IDs from Sequence IDs, since they are different concepts. Sometimes you may have sequenced some individuals more than once. These would make the creation of a Pedigree File in PLINK format a bit more difficult.

Previously, PERCH supports another Pedigree File format (hereafter PERCH format), which is basically a PLINK format Pedigree File with an extra column of Sequence ID that matches to the header row of a VCF file. For the

individuals that are not sequenced the Sequence ID is 0. This file has a header row, where the Sequence ID column should have the header name SeqID. Because of this extra column, the Individual ID does not need to match with the VCF header, nor does it need to be unique across pedigrees. Preparing a pedigree file in this format is more convenient than the PLINK format. Now this file format is no longer supported by most of my programs, but you can still use this format and then use the vConvertPed tool to convert the file to PLINK format. This tool reads two files, a Pedigree File in PERCH format (--ped) and a Genotype File in VCF (--geno). The reason to read a VCF is to find the sequenced individuals, so that it can translate the Individual IDs for the sequenced individuals only.

### 3.4. vConvertTest [Back to top]

This program read a variant (--chr --pos --ref --alt) and a pedigree file (--ped), then write a VCF file and a pedigree file (--prefix) for variant classification by PERCH.

The input pedigree file should have 8 columns: Pedigree_ID, Individual_ID, Father, Mother, Sex, Affection Status, Age, and Genotype of the variant. Age should be a positive number less than 150, otherwise it is deemed missing. Genotype is het, hom, neg, +/+, +/-, -/+, -/-, 2/2, 2/1, 1/2, 1/1. Other strings are deemed missing value.

The program will calculate sex- and age-specific liability classes (--age). There are options to specify what to do with a missing age (--unk-age), sex (--unk-sex), or affection status (--unk-aff).

### 3.5. vSPLIT [Back to top]

vSPLIT is a program to split multi-allelic variants into multiple lines. This is helpful for the downstream analysis, as each alternative allele may have a different deleteriousness score, population frequency, and functional consequence.

In splitting alternative alleles, the program splits the genotypes too if the file contains any genotype columns. For example, if a sample's genotype is 1/2 (i.e., alternative allele 1 and alternative allele 2), the person's genotype will become 1/a and a/1, respectively. Here 'a' denotes "another alternative allele", which can be changed by one of the options (-a). The program assumes that starting from the 10th column are genotypes, which conforms to the VCF format. If the last few columns are not genotypes, use the option (-l) to change the column number of the last sequenced individual.

This program also reads the INFO field and splits certain variables. There are several modes of splitting: a) Direct Splitting (--ds), when the values correspond to each alternative allele in the same order as the "ALT" field; b) Plus Reference (--pr), when there is an extra value for the reference allele at the end; and c) Genotype Counts (--gc), which are the number of observations for all possible genotypes. Please use the (-h) option to see the names of the default variables that will be split in each mode. These default variables are designed to work for VCF files from the 1000 Genome Project, the NHLBI GO Exome Sequencing Project, the Exome Aggregation Consortium, and VCF files created by the Genome Analysis Toolkit. You can specify more variables than those actually exist in a VCF file. So you don't have to change the variable names unless you have a variable that is not set by default.

This program does not split the AD field in each genotype column. This is because the AD field is not part of the standard VCF, and the usage of AD is not recommended by GATK: "Because the AD includes reads and bases that were filtered by the caller (and in case of indels, is based on a statistical computation), it should not be used to make assumptions about the genotype that it is associated with. Ultimately, the phred-scaled genotype likelihoods (PLs) are what determines the genotype calls."

### 3.6. vQC [Back to top]

**Input and Output**

vQC performs genotype-, variant- and sample-wise quality control (QC). The input is a VCF file with or without genotype fields. If it has no genotypes (such as those in ExAC), it is required to specify the total number of samples in the sequencing (--tot-spl) and the INFO sub-field name for AC (--info-ac) and AN (--info-an). The output is a modified VCF file after removing low-quality genotypes and variants. The reason for the removal of each variant can be logged to a file (--log). Some samples will be excluded from the output if specified (--rm-

ind). Uncommon intergenic or intronic variants may be excluded (--rm-ii-MaxAF-lt) to reduce the computation time of downstream analysis, such as phasing. To identify intergenic and intronic variants, the VCF input needs to have functional annotations in an INFO sub-field (--info-func, supports annotation by VANNER). vQC can also filter variants by variation type (--snv-only, --indel-only), chromosomal region (--include, --exclude), chromosome type (--auto-chr). De *novo mu*tations may be lost during a subsequent analysis, such as phasing by BEAGLE. vQC can solve this issue by exporting de *novo mu*tations to a file before the analysis (--out-dn), then insert them back to the VCF afterward (--insert-dn).

The formats of Pedigree File (--ped) and Sample File (--spl) are described in the Manual page for PERCH. If the Sample File contains population information (column 4, header "pop", case-insensitive strings, missing value is an empty string or . or unknown), Hardy-Weinberg equilibrium (HWE) tests (--filt-hwe-pv) will be performed in unaffected samples within each population. Non-founder individuals from the Pedigree File will be excluded automatically (--hwe-founder). If population is not defined, you have the option to choose whether to do HWE test in all independent controls (--hwe-controls).

**Filters**

Most options of this program are parameters for classical QC procedures, such as hard filtering (--hard-filter, --HardFiltIfNoVQ) or VQSLOD filtering (--filt-vqs-snv, --filt-vqs-indel), missing VQSLOD score (--filt-vqs-nan), Hardy-Weinberg equilibrium test (--filt-hwe-pv, --filt-hwe-info), FILTER field (--filt-filter), minor allele count (--filt-mac), total allele count (--filt-an), missing rate in cases or controls (--filt-miss-rate), missing rate discrepancy between cases and controls (--filt-miss-pv), genotype discordance among duplicated samples (--filt-discord), Mendelian errors excluding *de novo* mutations (--filt-Mendelian), the same *de novo* mutation in multiple individuals (--filt-de-novo), unoriginal *de novo* mutations (--filt-uo-dn), the proportion of heterozygous haploidy among non-missing haploidy individuals (--filt-hh), the proportion of samples with sufficient coverage (--filt-cov-DP, --filt-cov-pc), and the mean depth if smaller or greater than a certain threshold (--filt-min-DP, --filt-max-DP). By default, homozygous *de novo* mutations are treated as Mendelian errors but not a *de novo* mutation. Please see the program help text for the options for each of the above filters.

In addition, vQC implements a unique QC method using allele frequency. If you are searching for low-frequency variants, and you expect a low founder effect and high locus and/or allelic heterogeneity, you may want to exclude common variants from subsequent analysis. This filtering may be an efficient QC too because some sequencing artifacts may be "observed" in almost every sequenced samples. Similar to this idea, vQC removes variants that are supposed to be rare but are too common in the data. It takes a minor allele frequency cutoff (--filt-obs-maf). For variants whose MaxAF is smaller than the cutoff it calculates a one-sided p-value for observing the data among the samples (cases and controls altogether) given an allele frequency equal to the cutoff, and then removes the variants if the p-value is smaller than a user-defined threshold (--filt-obs-pv). As you may see, this method is rather conservative because variants were filtered based on p-value instead of the observed frequency. So this method is robust to founder mutations. This QC method may be valid even if the samples are enriched for a certain phenotype that is distinct from the disease of interest. A variant that is too common in those samples may be: 1) an artifact; 2) a common variant specific to the studied population; 3) a causal variant for the enriched phenotype; or 4) a protective variant for the disease of interest. For most studies the last situation is very unlikely and hence it is safe to remove these variants. Nevertheless, use it with caution.

The missing rate filtering can be done in cases and controls separately (--filt-miss-ea=yes) or jointly (--filt-miss-ea=no). However, sometimes it makes more sense to do it by cohort. Here a cohort means a sample set that is exome-captured by the same reagent and sequenced by the same platform with the same depth. Also, if you have sequenced pedigrees, some pedigree members may be indicated with an unknown affection status in the sample file (because they are not index cases), who will not be included in the missing rate calculation if (--filt-miss-ea=yes). In this case, you can create a Cohort File with 2 columns, Sample ID and Cohort ID, delimited by a tab. Use the (--cohort) option to specify this file, then the program will do the missing rate filtering by cohort only, disregarding case/control status or pedigrees. The cohort ID should not contain white spaces. It can contains any number of alphanumeric or special characters. Samples with an unknown cohort ID (empty or "." or "unknown") will be omitted.

**Duplications and Replications**

To perform quality control by genotype concordance among duplicated samples, vQC reads a **Duplication File** (--dup). This file does not have a header row. Each row are the different SeqIDs for one biological sample. The number of columns in each row may vary, but there should be at least two columns as these are duplications.

The vQC will compare column 2 with column 1, column 3 with column 1, and so on. Therefore, it is better to put the "gold standard" in column 1, and each experiment in a specific column consistently for all lines. Missing SeqID should be "0" or "." or empty. The first column should not be missing. At least one SeqID per line in this file should also be in the Sample File too, because vQC needs to obtain sex information from the Sample File. Below is an example of this file. Suppose you have performed four experiments: Sanger sequencing, exome array, and next-generation sequencing by two platforms, then the Duplication File may looks like this:

```
#Sanger Array NGS1  NGS2
S1S     S1A   S1N1  S1N2
S2S     0     S2N1  S2N2
S3S     S3A   0     S3N2
...
```

Besides duplications (a sample went through the same experiment twice), vQC can also perform quality control by genotype concordance among replications (a sample went through another experiment to validate the sequencing results, such as high-throughput SNP array genotyping or targeted sequencing). Variants in an array most likely have a common or low allele frequency, so it is not very helpful for the QC of rare variants discovered from sequencing. But it can still be used to estimate the concordance rate and find out a good threshold for GQ and DP. To do this, vQC reads two files (--rep). The first one is a **Replication ID File**, which translates IDs between the two experiments. The second one is a **Replication Genotype File**, a tabix-indexed cleaned VCF file that contains genotypes from the replication experiment. The number and order of samples in the second file need not be the same as in the primary VCF input file. The option (--out-dup) will write the GQ an DP values of all genotypes to a file. In that file, the first three columns are the comparison result (C for concordance, D for discordance, followed by the primary genotype in sequencing), GQ, and DP, respectively. If the option (--out-dup) is not set, vQC will write a genotype comparison summary to the standard error.

**Sample-wise QC**

vQC can also output several statistics for sample-wise quality control (--sample-qc), which include missing rate, heterozygous to non-reference homozygous ratio, Ti/Tv ratio of coding SNVs, genetic sex, mean GQ, mean DP, and the genotype concordance rate among replicated variants (if you have array data). Since it is most likely that the proportion of bad samples is small, while the proportion of bad variants could be large, these statistics are calculated after genotype-wise and variant-wise quality control. Ti/Tv was calculated for coding regions only so that it is comparable even between a whole exome and a whole genome sequencing. To identify coding variants, the Genotype File should have been annotated for functional consequences, where the annotation is in the INFO field (--info-func). vQC looks for the phrase Missense, Synonymous, StopLoss, StopGain and Translation within this sub-field for coding variants.

vQC infer sex from X and Y separately. It doesn't join the two chromosomes to make one inference because they may contradict to each other, which may be due to sample contamination or rare conditions. For sex inference from X, vQC requires an estimated genotyping error rate (--x-err-rate). Unless you use very gentle QC filters, the default value should be fine for most studies.

If you have array data and use it in the quality control (--rep), vQC reports the genotype call rate and the concordance rate among the overlapping variants.

**--join-sample-qc**

If you perform QC for one chromosome at a time, which is recommended for a large study, you can aggregate the results and calculate the genome-wide overall statistics (--join-sample-qc). If you have calculated genetic relatedness between independent samples by KING, vQC can read the KING output (--king) and select the individuals to be removed in subsequent analyses. It first finds groups of people that are correlated by kinship. The number of people in each group could be more than two. From each group, vQC selects the person who has the lowest missing rate. The remaining people will be written to a file named <prefix>.rel_problem. For this option to work, you also need to set the (--spl) and (--prefix) option. In a subsequent analysis that requires independent individuals, you can remove the samples listed in this file (--rm-ind+=<prefix>.rel_problem). The default of the (--king) option removes second degree relatives or closer (--kin-cutoff), because the kinship calculation is not very reliable beyond this relatedness level. If you'd like to include related individuals in an association test, using individual weights to control for genetic relatedness, then you can choose to remove duplicates and monozygotic twins only (--kin-cutoff=0.35355).

Table: contents of --sample-qc or --join-sample-qc output.

| Column | Header | Content |
|---|---|---|
| 1 | SeqID | Sequence ID |
| 2 | Sex | Input sex from Sample File |
| 3 | ms | Missing genotypes on autosomal chromosomes |
| 4 | non-ms | Non-missing genotypes on autosomal chromosomes |
| 5 | Het | Heterozygous genotypes on autosomal chromosomes |
| 6 | AltHom | Homozygous ALT genotypes on autosomal chromosomes |
| 7 | HetRare | Heterozygous rare variants within the minimum exome on autosomal chromosomes |
| 8 | HomRare | Homozygous rare variants within the minimum exome on autosomal chromosomes |
| 9 | HetPers | Heterozygous personal variant within the minimum exome on autosomal chromosomes |
| 10 | HomPers | Homozygous personal variant within the minimum exome on autosomal chromosomes |
| 11 | HetSing | Heterozygous Singleton within the minimum exome on autosomal chromosomes |
| 12 | HomSing | Homozygous Singleton within the minimum exome on autosomal chromosomes |
| 13 | Ti | Ti coding SNVs on autosomal chromosomes |
| 14 | Tv | Tv coding SNVs on autosomal chromosomes |
| 15 | MsRate | Missing rate on autosomal chromosomes |
| 16 | Het/Hom | Het/Hom ratio on autosomal chromosomes |
| 17 | HetRate | Heterozygous rate on autosomal chromosomes |
| 18 | Ti/Tv | Ti/Tv ratio on autosomal chromosome coding regions |
| 19 | Y_ms | Missing on Y |
| 20 | Y_call | Genotype calls on Y |
| 21 | Y_call% | Genotype call rate on Y |
| 22 | Y_sex | Genetic sex inferred from Y |
| 23 | X_call | Genotype calls on X |
| 24 | X_het | Heterozygous genotypes on X |
| 25 | X_LOD | LOD score for sex inference from X |
| 26 | X_sex | Genetic sex inferred from X |
| 27 | RepGeno | Number of variants that replication experiment has a genotype call |
| 28 | RepBthG | Number of variants that both replication and this VCF have a genotype call |
| 29 | RepConc | Number of concordant genotype calls |
| 30 | Call% | Call rate in this VCF among the variants that replication has a genotype call |
| 31 | Conc% | Concordance rate among the variants that both have a genotype call |
| 32 | numGQ | Number of non-missing GQ score |
| 33 | numDP | Number of non-missing DP score |
| 34 | meanGQ | mean GQ score |
| 35 | meanDP | mean DP score |

**Overall quality**

vQC assesses the overall quality by the number of rare variants per sample. Rare variant means MaxAF<0.05 (--rv). This measure is more interpretable than singletons as the latter relies on the number of samples sequenced. To make this measure comparable between studies where the targeted regions may be different, rare variant are counted only within the minimum overlap regions across multiple exome capturing reagents including Agilent V4 V5 V6, NimbleGen V2 V3, VCRome V2, the well-covered (15x in 95% of the samples) regions in ExAC, and the gene regions in refSeq and Ensembl. vQC reports these numbers break down by case-control status, so that you can evaluate whether there's a batch effect between cases and controls and whether the cases and controls are comparable. These numbers are also reported for each individual so that it can be used for sample-wise QC too (--sample-qc).

### 3.7. vQS [Back to top]

This program reads the VQSLOD scores output from “vQC --out-vqs” and determines the VQSLOD cutoff for SNV and InDel separately. The cutoff value is the lower fence for outliers from known variants by Kimber’s method. If this cutoff is beyond the lower bound (--lb) or upper bound (--ub) of the presumed range, the program will write an error message to the standard output, otherwise it will write a program option with the cutoff value for subsequent analyses.

### 3.8. vINFO [Back to top]

This program modifies a VCF file. It can remove specific INFO fields (--remove) or remove all INFO fields except the specific ones (--keep). A unique utility of this program is to restore the original POS,REF,ALT data after vAnnGene run (--restore). In functional consequence annotation by vAnnGene, these fields are modified to reflect the biological consequence of the variant. Therefore, some variants may be left-aligned, while others right-aligned. This is helpful for a correct deleteriousness annotations. Fortunately, vAnnGene also stores the original POS,REF,ALT data in the INFO field, so you can restore them by this program if you want.

### 3.9. vPROV [Back to top]

This program annotate PROVEAN for in-frame insertions or deletions by calling the PROVEAN program. It applies multithreading (--nt) for faster computation.

### 3.10. HGVS_to_genomic [Back to top]

This program reads HGVS nomenclatures and translate them into genomic coordinates and sequence changes. Input is a file with HGVS in the first column; other columns will be ignored. Output to stdout is another file with 7 columns: HGVS, CHR, POS, HGVS, REF, ALT, INFO. Note that the first and fourth columns are the same for historical reasons. You can easily convert this output into a VCF file by the following commands:

```
echo '##fileformat=VCFv4.1' > HGVS_to_genomic.output.vcf
echo $'#CHROM\tPOS\tID\tREF\tALT\tQUAL\tFILTER\tINFO' >> HGVS_to_genomic.output.vcf
grep -v ^# HGVS_to_genomic.output | cut -f 2- | sed 's/\t/\t.\t.\t/5' >> HGVS_to_genomic.output.vcf
```

Features of this program include:
1) Automatically check for reference sequence errors
2) If only a gene symbol is given, translate it to the principal transcript defined in APPRIS;
3) If only a gene synonym is given, translate it to a gene symbol and then the principal transcript;
4) If multiple HGVS nomenclatures are given, select the unambiguous one (choose c.xxx over p.xxx);
5) Some non-standard or deprecated HGVS are supported. For example, c.IVS1-5T>C, p.(Arg3381Ser).
6) If a protein-level HGVS have multiple genomic interpretations, select the most probable one that has a higher allele frequency, has a matching in dbSNP or ClinVar, or a lower deleteriousness score.

Please use the option (--singular) to turn on the feature of choosing the most probable SNV for a protein-level HGVS. For this to work, the program requires a MaxAF database (--MaxAF), a dbSNP database, and the BayesDel score database (--del). VICTOR already provides all these databases, so you don't have to set these options.

## 4. Topics

### 4.1. Implemented quality Control [Back to top]

**Genotype-wise QC:**

1) Remove bad genotypes. Genotypes are deemed missing if the DP (read depth) or GQ (genotype quality conditional on the variant being true) score is less than a threshold, which can be changed by a program option, (--filt-dp) and (--filt-gq), respectively.

2) Correct genotypes based on the expected ploidy of chrX, chrY, chrM for each individual (sex-dependent). It takes the pseodo-autosomal region (PAR) into account.

**Variant-wise QC:**

1) Remove bad variants. vQC filter variants by VQSLOD, the FILTER field, missing rate, missing rate discrepancy between cases and controls, Mendelian errors in pedigrees excluding de novo mutations, the same de novo mutation in multiple individuals, the probability of observing data give a low allele frequency, Hardy-Weinberg disequilibrium in independent controls and/or external samples, genotype discordances among duplicated samples, the percentage of samples covered by sequencing with a minimum depth, the mean depth smaller or greater than a certain threshold, and the proportion of heterozygous genotypes among non-missing haploidy individuals. If VQSLOD is not available, vQC automatically applies hard filtering by QD, MQ, FS, HaplotypeScore, MQRankSum, and ReadPosRankSum.

2) Quantitatively integrate quality of variant calls. Quality control by variant filtering may not be sufficient because variants at the borderline of a filtering threshold may not be clean enough. To alleviate this problem, PERCH integrates VQSLOD into a variant weight together with a deleteriousness score. Therefore, this method takes a balance between variant call quality and biological relevance.

3) Select regions or variant types. vQC can filter variants by chromosomal region and variant type. A common usage of this feature is to select variants within the intersection of captured regions between cohorts, and/or exclude the regions that are too difficult for next-generation sequencing. You can also restrict the analysis to SNV only.

4) Wrong REF. The annotation program vAnnGene checks whether the REF sequence matches with the reference genomic sequence. Variants that do not match will be reported.

**Sample-wise QC:**

1) Sex

2) Het/non-ref-hom ratio

3) Ti/Tv ratio

4) Missing rate

5) Mean GQ and DP across all variants

6) pedigree structure errors (by the program KING)

7) cryptic relatedness among unrelated individuals (by the program KING)

8) population stratification (by the program PLINK)

The script slurm.all_steps doesn't conduct QC based on .bam files, such as contamination rates and sequencing yields. But I have provided a script to do so in the Tutorial. Other QC methods are not yet implemented but may be considered in our future work, such as multiple recombination events on one chromosome and homozygous or compound heterozygous mutations in critical genes.

### 4.2. Compare to ExAC

[Back to top]

You need to provide the sequencing coverage on each basepair of the genome in your cases. The coverage files (one chromosome each file) can be generated by the following script from .bam files:

```
export DP_CUT=20
export PC_CUT=80
export BAM_PFX=prefix.of.bam.files
export BAM_SFX=suffix.of.bam.files
export SPL=samples.txt
export OUT=VICTOR
export GENOME=GRCh37

# first, make the list of cases that have a bam file.
cut -f 1 $SPL | sed "s/\(.*\)/\1$BAM_SFX ]; then echo \1; fi/" | sed "s;^;if [ -s $BAM_PFX;" | bash > samples.bam_available

# calculate read depth on each position for all samples
mkdir -p ${DP_CUT}x/perInd
my_func() { samtools depth -q 10 -Q 20 ${BAM_PFX}$1${BAM_SFX} | awk '($3>='${DP_CUT}')' | gzip -c > ${DP_CUT}x/perInd/$1.depth.gz; }
export -f my_func
module load parallel
parallel -a samples.bam_available my_func

# join the results.
cat samples.bam_available | sed "s;^;${DP_CUT}x/perInd\/;" | sed "s/$/.depth.gz/" > samples.to_join
vBED2 --genome=$GENOME --filelist samples.to_join --format=samtools --dp=${DP_CUT} --pc=0 --log=${DP_CUT}x/files.to_join --nt=auto >/dev/null
rm samples.to_join*
# to save time, do it by a batch of 300 to 500 samples, and set the options --filelist and --log accordingly. Below is an example:
# cat samples.bam_available | sed "s;^;${DP_CUT}x/perInd\/;" | sed "s/$/.depth.gz/" | head -300 > samples.to_join.1
# vBED2 --genome=$GENOME --filelist samples.to_join.1 --format=samtools --dp=${DP_CUT} --pc=0 --log=${DP_CUT}x/files.to_join.1 --nt=auto \
# >/dev/null

# now create the coverage files
mkdir -p ${DP_CUT}x/${PC_CUT}pc
TotSpl=`cat samples.bam_available | wc -l`
RequiredSpl=`echo "$TotSpl * 0.$PC_CUT" | bc`
RequiredSpl=`echo "($RequiredSpl+0.5)/1" | bc`
vBED2 ${DP_CUT}x/files.to_join* --genome=$GENOME --format=vBED --min-spl=$RequiredSpl --tot-spl=$TotSpl \
      --log=${DP_CUT}x/${PC_CUT}pc/p_samples.txt --nt=auto > ${DP_CUT}x/${PC_CUT}pc/Covered.bed
bgzip ${DP_CUT}x/${PC_CUT}pc/p_samples.txt
tabix -p vcf ${DP_CUT}x/${PC_CUT}pc/p_samples.txt.gz
CHRS=( 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X )
for CHR in ${CHRS[@]}; do
  echo "#DP_CUT=$DP_CUT"  > ${DP_CUT}x/${PC_CUT}pc/$OUT.chr$CHR.cov
  echo "#PC_CUT=0.$PC_CUT" >> ${DP_CUT}x/${PC_CUT}pc/$OUT.chr$CHR.cov
  tabix ${DP_CUT}x/${PC_CUT}pc/p_samples.txt.gz $CHR:1-300,000,000 >> ${DP_CUT}x/${PC_CUT}pc/$OUT.chr$CHR.cov
  gzip -f ${DP_CUT}x/${PC_CUT}pc/$OUT.chr$CHR.cov
done
```

Then, in slurm.all_stpes, add the following parameters.

```
# What to do.
ExACnT=yes

# Main Parameters.
export COV=Prefix.of.the.coverage.files.for.your.cases # if you use the above script, COV=${DP_CUT}x/${PC_CUT}pc/$OUT
export EFP=Prefix.of.the.ExAC.files                    # the default is ExAC1/noTCGA, it uses the provided nonTCGA data
export AWX=additional.options.for.vAAA                 # use this parameter to specify which ExAC population to use
```

If you want to compare to a specific population rather than the entire ExAC, use the (--xct-pop) option for the AWX parameter in the slurm.all_steps script. An example is AWX="--xct-pop=NFE". Now you can do the analysis to compare your cases to ExAC nonTCGA samples.

Note that the annotation parameters for the ExAC data must match with those for the cases. The ExAC1 folder provided by VICTOR is pre-computed from ExAC nonTCGA samples, annotated with refGeneLite and default parameters. If you like to annotate with another gene database, then you have to generate the ExAC files. Also, if you like to use another ExAC subset (nonPsych) or the whole ExAC, you need to generate the ExAC files too. Below is the script to make these files.

```
# set parameters according to ExAC subset (nonTCGA, nonPsych, all)
export ExAC_FILE=ExAC_nonTCGA.r1.sites.vep.vcf.gz # downloaded file
export DP_CUT=20    # depth cutoff
export PC_CUT=60    # percent of samples with read depth >= $DP_CUT
export TotSpl=53105 # 53105 for nonTCGA, 60702 for all
export OUT=ExAC1    # ExAC1 for ExAC r1
export PFX=noTCGA   # noTCGA or noPsych or all

# 2) Prepare ExAC's coverage file $PFX.chr$CHR.cov.gz.
mkdir coverage && cd coverage
wget -r -np -nd -nH  ftp://ftp.broadinstitute.org/pub/ExAC_release/release0.1/coverage/
if   [ "$DP_CUT" == 30 ]; then COLUMN=11
elif [ "$DP_CUT" == 25 ]; then COLUMN=10
elif [ "$DP_CUT" == 20 ]; then COLUMN=9
elif [ "$DP_CUT" == 15 ]; then COLUMN=8
elif [ "$DP_CUT" == 10 ]; then COLUMN=7
else echo >&2 ERROR: DP_CUT is wrong.
fi
CHRS=( 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X )
for CHR in ${CHRS[@]}; do
  echo "#DP_CUT=$DP_CUT"  > $PFX.chr$CHR.cov
  echo "#PC_CUT=0.$PC_CUT" >> $PFX.chr$CHR.cov
  zcat Panel.chr$CHR.coverage.txt.gz | sed 1d | awk '($'$COLUMN'>=0.'$PC_CUT')' | awk '{print $1,$2,$'$COLUMN'}' >> $PFX.chr$CHR.cov
  gzip -f $PFX.chr$CHR.cov
done
mkdir -p ${PC_CUT}pc${DP_CUT}x && mv *.cov.gz ${PC_CUT}pc${DP_CUT}x
cd ..

# 3) prepare ExAC's annotation file $PFX.chr$CHR.ann.del.gz and quality control log file $PFX.chr$CHR.qc.log
export MaxAFDB=MaxAF_20170801.gz
export VAG=
export DEL=
HWE_FIELDS=AN_NFE,Het_NFE,Hom_NFE,AN_AFR,Het_AFR,Hom_AFR,AN_SAS,Het_SAS,Hom_SAS,AN_EAS,Het_EAS,Hom_EAS,AN_AMR,Het_AMR,Hom_AMR
rm -rf ${OUT} ${OUT}_log
mkdir -p ${OUT}
mkdir -p ${OUT}_log
my_func() {
  tabix $ExAC_FILE $1:0-300,000,000 -h | vSPLIT |\
  vAnnGene --norm-only | vAnnDel --ann=dbSNP --ms=index |\
  vAnnGene --no-filter --no-split --add-info --info-func=vAnnGeneAll |\
  vAnnDel  --add-info --ann=$MaxAFDB --add-ms |\
  vAnnBase --add-info --ann=BayesDel_nsfp33a_noAF -x=3 --min=-1.5 --step=0.01 --indel=max --padding=1 |\
  vAnnDel  --add-info --ann=InDels.provean.tsv.gz -f 5 --wr=PROVEAN --preload --indel-only |\
  vDEL --no-sort --check-ms=${OUT}_log/${PFX}.chr$1 --info-func=vAnnGeneAll |\
  vQC --info-func=vAnnGeneAll --tot-spl=$TotSpl --info-ac=AC_Adj --info-an=AN_Adj --filt-miss-rate=1 \
      --filt-hwe-info=$HWE_FIELDS --filt-vqs-snv=-inf --filt-vqs-indel=-inf --log=${OUT}/${PFX}.chr$1.qc.log.gz --nt=auto-12 |\
  vAnnGene --all-ann=vAnnGeneAll $VAG | gzip -c > ${OUT}_log/${PFX}.chr$1.ann.gz; }
export -f my_func
module load parallel
parallel -a ~/local/VICTOR/script_template/chr_noMnoY my_func
cat ${OUT}_log/${PFX}.chr*.for_PROV > ${OUT}_log/${PFX}.provean
### submit ${OUT}_log/${PFX}.provean to provean.jcvi.org, calculate, and save results to ${OUT}_log/${PFX}.provean.result.one.tsv
sed 1d ${OUT}_log/${PFX}.provean.result.one.tsv | cut -f 2,11 | grep -v NA | grep -v $'\t$' | tr , '\t' |\
sort -k 1,1 -k 2,2n -k 3,3 -k 4,4 | sed '1s/^/#CHROM\tPOS\tREF\tALT\tPROVEAN\n/' > ${OUT}_log/${PFX}.provean.tsv
if [ -s ${OUT}_log/${PFX}.provean.tsv ]; then
  bgzip -f ${OUT}_log/${PFX}.provean.tsv
  tabix -f -p vcf ${OUT}_log/${PFX}.provean.tsv.gz
  my_func() {
    vAnnDel ${OUT}_log/${PFX}.chr$1.ann.gz --indel-only --ann=${OUT}_log/${PFX}.provean.tsv.gz --preload -f 5 --wr=PROVEAN --add-info |\
    vDEL --add-af $DEL |\
    vINFO --remove CSQ,DP_HIST,GQ_HIST,AGE_HISTOGRAM_HET,AGE_HISTOGRAM_HOM |\
    gzip -c > ${OUT}/${PFX}.chr$1.ann.del.gz;
    cp coverage/${PC_CUT}pc${DP_CUT}x/$PFX.chr$1.cov.gz ${OUT}/${PFX}.chr$1.cov.gz;
  }
  export -f my_func
  module load parallel
  parallel -a ~/local/VICTOR/script_template/chr_noMnoY my_func
else
  my_func() {
    vDEL ${OUT}_log/${PFX}.chr$1.ann.gz --add-af $DEL |\
    vINFO --remove CSQ,DP_HIST,GQ_HIST,AGE_HISTOGRAM_HET,AGE_HISTOGRAM_HOM |\
    gzip -c > ${OUT}/${PFX}.chr$1.ann.del.gz;
    cp coverage/${PC_CUT}pc${DP_CUT}x/$PFX.chr$1.cov.gz ${OUT}/${PFX}.chr$1.cov.gz;
  }
  export -f my_func
  module load parallel
  parallel -a ~/local/VICTOR/script_template/chr_noMnoY my_func
fi
```

## 4.2. BayesDel score

[Back to top]

BayesDel is a deleteriousness meta-score. You can calculate BayesDel for a VCF file using the provided script template. Please see the Tutorial for detailed procedures. If you want to annotate just a few number of

variants, you can also use the online calculator (see the USE NOW page for the URL). The range of the score is from -1.29334 to 0.75731. The higher the score, the more likely the variant is pathogenic. There is a universal cutoff value (0.0692655 with MaxAF) that was obtained by maximizing sensitivity and specificity at the same time. There are also gene-specific cutoff values. Please see the BayesDel_GST file within the data folder for the cutoff. In that file the first column is gene symbol, the second is cutoff for BayesDel without MaxAF, the third is cutoff for BayesDel with MaxAF. It's important to note that these cutoff values were designed for gene discovery research, not for clinical operations, where you may want to have a double-cutoff system, i.e., BayesDel>cutoff1 is likely pathogenic, BayesDel<cutoff2 is likely benign, while others are variants of uncertain significance.