pVAAST User Guide

written by
Hao Hu
and
Chad Huff

Huff Lab
Department of Epidemiology
The University of Texas MD Anderson Cancer Center
Houston, Texas

Table of Contents

1. Introduction

The **Pedigree Variant Annotation, Analysis and Search Tool** (pedigree VAAST, or pVAAST) is a statistical package for identifying genetic variants causing disease or influencing disease risk using DNA sequencing data. It is built upon The Variant Annotation, Analysis and Search Tool (VAAST), a software package that uses DNA sequencing data from unrelated cases and controls to identify genetic variants causing disease or influencing disease risk. pVAAST retains all the functionality of VAAST and is backward compatible with all VAAST commands. In addition, it supports DNA sequencing data from outbred families. Therefore, in addition to this guide, the VAAST quick-start guide, VAAST User-guide, and VAAST protocol paper are also excellent references. Much of the material from these documents are included in this guide.

So what does pVAAST do exactly?

pVAAST is a software package that prioritizes genes and genetic variants according to their evidence of association with a disease phenotype, using germline DNA sequencing data from affected individuals, their affected/unaffected relatives and healthy controls. Broadly speaking, the core test statistic in pVAAST falls into a large class of genetics tests called *rare-variant association tests*, which detect associations between disease phenotypes and frequencies of rare variants in a gene. Notable examples of these types of tests include *Sequence Kernel Association Test (SKAT), Weighted Sum Statistics (WSS),* and *Variable Threshold (VT) test*. pVAAST differs from other rare variant association tests in a few key aspects:

- **Incorporation of linkage information**
  pVAAST calculates a genetic linkage score, or *logarithm of odds (LOD)* score, for each family that contains affected individuals. The pVAAST LOD score is similar to the traditional LOD score generated by a two-point linkage analysis but is designed for sequencing data and is calculated across an entire gene rather than for an individual variant. Therefore, it is more powerful compared to classic linkage analysis on sequencing datasets (*Hu et al, Nature Biotechnology 2014)*. pVAAST integrates the LOD scores, variant frequency information and the functional predictions into a unified framework, and calculates a single p-value to evaluate the evidence of association.
- **Sensitive to both common and rare risk variants**
  Most rare-variant association tests were designed for low-frequency risk variants. In comparison, pVAAST maintains high statistical power when either common or rare variants are causal (or both).
- **Providing a comprehensive report to help pinpoint exact causal variant**

As compared to most other tools that output only gene-level p-values, pVAAST provides a detailed variant-level report to help locate causal variants. The report includes 1) a LOD score to show the linkage signal; 2) an aggregated variant score that summarizes the evidence for a single variant and 3) an easy-to-read genotype summary for all sequenced individuals.

- **Controlling for Type I error**
  pVAAST uses a combination of re-sampling and gene-drop techniques to calculate exact rather than asymptotic p-values. Therefore, pVAAST has well calibrated Type I error rates regardless of sample size or the number of families.

2. Getting ready to run pVAAST

2.1. System requirements

pVAAST runs on any Linux or Mac OS system that have Perl installed. The requirement for memory and CPU resources depends on the size of the data. On smaller cases and controls (e.g. less than 5 case individuals), a 2-core MacBook Pro laptop with 8GB memory is sufficient to run pVAAST. On large samples, typically a server with at least 10 CPUs and >=80 GB memory is recommended.

2.2. Installation

Installing pVAAST is essentially the same process as installing VAAST. The text below is copied from VAAST Quick-Start guide, and provides a concise description of this process.

1. After you have downloaded a copy from the website, you will have a file called: VAAST_Code_2.X.X.tar.gz in your downloads directory. Simply move this file to the directory wherein you want VAAST to reside. Then type the following command:

   ```
   tar -xzvf VAAST_Code.tar.gz
   ```

2. Make sure you have a C compiler then type the following commands (you can answer 'no' when asked to install the optional dependencies):

   ```
   perl Build.PL
   sudo ./Build installdeps
   ./Build test
   ./Build install
   ```

See the VAAST User's guide for more details on the installation process.

You may also want to add VAAST/bin and VAAST/bin/vaast_tools to your PATH environment variable, but this isn't a requirement. It should be ready to run after installation.

2.3 Need help?

If you have a question and cannot find the answers in this user guide, please post it to our mailing list (vaast-user@yandell-lab.org), which is actively maintained by our VAAST developers. *Please let us know your issues, comments and feature suggestions. We would love to help!*

3. Basic workflow

3.1 Overview

pVAAST utilizes three types of information for disease gene finding:

1. Variant frequency data in cases and controls,
2. AAS (Amino Acid Substitution) information of individual variants, and
3. Linkage information in pedigrees

VAAST uses the composite likelihood ratio test (CLRT) framework described in *Yandell, et al. 2011 Genome Research;* each of three components comprises an independent likelihood ratio score in the final gene statistic. pVAAST then uses a combination of re-sampling and gene-drop procedures to compute a gene-level p-value.

The set of genomes being analyzed for disease causing features (cases) are referred to below as the *target* genomes or *target* pedigrees. The set of healthy genomes (controls) that the target genomes are being compared to are referred to as the *background* genome (**target = cases. Background = controls).**

The basic inputs to pVAAST consist of:

1. A set of target variant files (including the family members of targets) in either .vcf or .gvf format
2. A set of background variant files (.vcf or .gvf format)
3. A set of features to be scored, usually genes (.gff3 format)
4. A multi-fasta file of the reference genome
5. A 6-column plink .ped file describing the pedigree structure and phenotypes of pedigree members
6. A parameter file specifying pVAAST options (.ctl format)

Inputs (3) and (4) can be directly downloaded from the VAAST FTP site. The instructions for creating (1), (2), (5), and (6) are given in the next section.

3.2 Creating the target and background variant files

Whenever possible, case and control genotypes should be generated with the same sequencing platform and variant calling pipeline. As with most association tests, pVAAST assumes that under the null hypothesis, cases and controls are sampled from a homogenous population. When case and control genotypes are generated using different sequencing platforms, the null hypothesis is violated, which will inflate the p-value Type I error and reduce prioritization accuracy.

In many situations, it is not possible to obtain matched cases and controls. For these scenarios, we recommend jointly calling cases and controls genotypes using the GATK UnifiedGenotyper or HaplotypeCaller, which greatly alleviates issues related to cases and controls that are poorly matched on sequencing platform or variant calling protocols. The VAAST FTP site also provides background CDR files created from publically available genomes. However, the users should be cautious about possible population stratification and platform bias issues when using these background CDR files.

This requires running the following steps:

1. **VCF file conversion**
   If you start from VCF genotype files, it is necessary to convert it to the GVF file format for the downstream analysis. This is done using the vaast_converter program.

2. **Variant Annotation**
   The variants for the individual genomes are annotated for the effects that they cause on the genomic features. Common SNV effects on genes are synonymous and non-synonymous codon changes, stop-codon loss and gain and splice-site variants. This is done using the VAT program.

3. **Variant Selection**
   Some set of (or commonly all) the variants in target genomes are combined together for comparison against the set of all background variants. Individual research designs may call for selection of a subset of variants, for example all variants shared by affected family members, but not present in unaffected family members. This is done using the VST program.

The final outputs of these three steps are in condenser format (with .cdr extension), which provides a condensed representation of annotated genotypes. Detailed descriptions on executing these procedures are available in VAAST User Guide. However, if you start from multi-sample VCF genotype files, pVAAST has a streamlined process for creating CDR files, using the vcf2cdr.pl program.

vcf2cdr.pl starts from multi-sample VCF file(s) and creates CDR files for subsequent pVAAST runs. It calls vaast_converter, VAT and VST internally and can parallelize the VAT and VST steps by chromosomes. The basic vcf2cdr.pl syntax is:

```
perl vcf2cdr.pl -vcf <vcf filename(s)> --output <output suffix>  --build <hg18/hg19> --fasta
<human genome fasta file>  --gff3 <human genome annotation file in gff3 format> --info <info
file> --cpus <number of CPUs to use>
```

You can download the .fasta and .gff3 file from the VAAST FTP sites. Otherwise, you can customize your own feature files (see the **VAAST User Guide** for more details).

The info file (--info) contains three tab-delimited columns for each individual: individual ID, filename prefix for the individual's destination CDR file and sex. The individual ID must match the ID provided in the VCF file. The sex can be either female or male. All non-familial, unaffected individuals will typically go to a single control CDR file. Individuals from the same pedigree, including affected individuals in a single pedigree and their sequenced pedigree members, should go to the same destination CDR file. Sporadic cases, or affected individuals with no sequenced family members, should all go into a single CDR file.

The final output CDR files can be found in <$prefix-step4> folder.

A test case of vcf2cdr.pl script can be found at: examples/vcf2cdr_example/vcf2cdr.sh.

3.3. Creating the pedigree file

pVAAST expects one pedigree file per family in standard 6-columns plink .ped format. Detailed descriptions can be found at pngu.mgh.harvard.edu/~purcell/plink/data.shtml#ped. Briefly, each line corresponds to one family member with the following 6 columns:

- **Family ID:** pVAAST ignores the content of this column.
- **Individual ID:** If this is a sequenced individual, this ID should match the FILE-INDEX ID of the same individual in the corresponding genotype (.cdr) file. This column cannot be 0.
- **Paternal ID**
- **Maternal ID:** If the individual is a family founder, then both the paternal ID and the maternal ID should be set to 0 (meaning missing). Otherwise, neither of them can be 0.
- **Sex:** 1 represents male, 2 represents female and 0 represents unknown sex.
- **Phenotype:** 1 represents unaffected and 2 represents affected. If the phenotype is either unavailable or ambiguous, then set to 0.

pVAAST also has a few special requirements regarding the pedigree structure:

1. The family cannot be an inbred family.
2. Any nuclear family cannot have more than one extended pedigree. That is, for any given individual, either the individual's paternal extended family is included or the individual's maternal extended family is included. It is ok to have neither extended family included, but not both.
3. Under the recessive model, pVAAST can currently only analyze two-generation nuclear families.

If the family that you are analyzing does not meet the above requirement, you will need to modify the pedigree and genotype files accordingly.

3.4. Creating your pVAAST parameter file

pVAAST is a highly flexible software package applicable to many different disease scenarios, including dominant/recessive inheritance modes, common, rare and/or Mendelian diseases, *de novo* causal variants etc. There are many tunable options in pVAAST. Generally, VAAST parameters are provided in the command line, while pVAAST options are given in a parameter file that is fed to pVAAST via the –pv_control option. You can typically find a suitable template parameter file in 'data/pvaast' folder . The filenames of these templates should be informative. However, if you cannot decide which file to choose, "generic.ctl" can be used as a general-purpose template file to start with.

A detailed description on each of the parameters is below:

Basic options

input_ped_cdr_files

Whitespace separated list of .ped and .cdr files for pedigrees containing at least one affected individuals. Each .ped and .cdr file pair contains the genotype and phenotype information for a single pedigree. For example, if you have two families, you should provide something like "fam1.ped fam1.cdr fam2.ped fam2.cdr".

pedigree_representative

Designating one affected and sequenced representative from each pedigree. Representatives are designated by their sample ID which must be identical to the FILE-INDEX for that individual in the CDR file. Pedigree representatives are an important concept in pVAAST that serve two major functions. First, only the pedigree representatives will contribute to the calculation of the case-control component of the composite likelihood. Second, LOD scores will only be calculated for variants carried by the pedigree representatives. The optimal pedigree representatives are affected individuals in the center of a pedigree that would be obligate

carriers assuming Mendelian rules of inheritance. When the pedigree_representative option is left blank, pVAAST will choose a pedigree_representative for the analysis based on the specified unknown_representative option (see below).

unknown_representative [yes|no]

When the pedigree_representative option above is left blank, pVAAST will choose a pedigree_representative for the analysis based on the setting of the unknown_representative option. If the unknown_representative option is set to no, pVAAST will randomly choose one pedigree representative for the entire analysis. If the unknown_representative option is set to yes, pVAAST will examine each variant in every individual in the pedigree and attempt to choose the pedigree representative that is most likely to carry the causal variant for each feature (i.e. selects the individual that maximizes the statistic specified by the informative_site_selection parameter). This option maximizes the chance of pVAAST correctly identifying a disease-causing variant in a pedigree. However, because pVAAST must evaluate every variant in every individual in the pedigree, the uknown_representative = "yes" option can result in a considerable loss of statistical power (i.e. higher p-values) due to the number of additional tests that are performed. Note that if a representative is set and the causal variant is not present in that individual, the variant will not be scored.

informative_site_selection [1|2|3]

The selection scheme for the causal variant. In each gene, pVAAST designates one variant as the causal mutation within each pedigree. 1 stands for selecting sites based on CLRT score; 2 for LOD score; 3 for LOD+CLRT score. The choice depends on the relative amount of signal coming from linkage and association. If you have only a few large pedigrees and expect high LOD scores, then 2 should be chosen; if you have many small pedigrees and expect high locus/allelic heterogeneities, then 1 should probably be used. For intermediate situations, choose 3.

additional_cases

A CDR file containing all additional unrelated affected individuals (ie., sporadic cases) to be included in the pVAAST analysis.

inheritance_model [dominant|recessive]

When the inheritance_model is set to dominant, pVAAST selects the one highest-scoring variant in the feature and use its LOD score as the gene LOD score. When the option is set to recessive, pVAAST examines the LOD scores for all possible pairs of variant alleles (occurring in any individual) and sets the gene LOD score to the LOD score of the highest scoring allele pair.

penetrance_lower_bound

A floating point number from 0 to 1. See below.

penetrance_upper_bound

A floating point number from 0 to 1. The penetrance_lower_bound and penetrance_upper_bound options describe the permitted range of penetrance for the causal variant. Values from 0 to 1 are allowed. A value of 0 represents complete lack of penetrance of the causal allele and a value of 1 represents complete penetrance. Increasing the lower bound above 0 penalizes any variant which segregates with penetrance below the value set for penetrance_lower_bound. Likewise, decreasing the upper bound below 1 will penalize any variant which segregates with a penetrance above the value set for penetrance_upper_bound. Note that the --penetrance option in VAAST should be independently set and does not have an effect on pVAAST.

Performance tuning options

simulate_genotyping_error [yes|no]

Setting the simulate_genotyping_error option to yes allows pVAAST to simulate genotyping error during the gene drop simulation. Allow this genotyping error simulation improves the accuracy of the p-value estimate. This option should typically be set to yes unless inheritance errors have previously been removed from the samples. Setting simulate_genotyping_error to yes is required for accurately scoring of *de novo* variants.

genotyping_error_rate

A floating point number between 0 and 1. The genotyping_error_rate option sets the rate at which genotyping errors are introduced during the gene-drop permutation test. Setting this option accurately is critical for accurately prioritizing and assessing the statistical significance of *de novo* mutations. The default value is 1e-4, but the value can be estimated from the data (when at least one complete trio is available) using the estimate_genotype_error_rate.pl script in the bin/vaast_tools directory.

Gene and Variant Filtering options

max_prevalence_filter

A floating point number between 0 and 1. This is the maximally allowable prevalence of the disease. The max_prevalence_filter option allows the user to set the maximum rate at which the observed phenotype occurs in the population, therefore constraining the search space explored during LOD score calculation. Setting this option accurately will improve both the power to detect causal alleles and the performance of the algorithm. Note that this option deals with the incidence of the phenotype in the population whereas the --rate option in VAAST

sets the upper bound of the causative allele in the population and thus they can both be set to improve power when applicable.

lod_score_filter [yes|no]

A value of "yes" will instruct pVAAST to score only genes with positive LOD scores. Setting this option can help remove false positive signals by ensuring that the variant is supported by some evidence from the linkage model. Without this option, variants can be identified based solely on the case-control comparison with no supporting familial evidence.

clrt_score_filter [yes|no]

A value of "yes" will instruct pVAAST to score sites with a positive CLRT score. Setting this option can help remove false positive signals by ensuring that the variant is supported by some evidence from the case-control and variant prioritization model. Without this option, variants can be identified based solely on the familial information with no supporting case-control or variant prioritization evidence.

nocall_filter [yes|no]

A value of "yes" will remove sites with N or more nocalls. See next option.

nocall_filter_cutoff

A positive integer (default: 2). When the previous option is enabled, pVAAST will filter sites with this number of nocalls or more. This filter is applied separately to each pedigree.

inheritance_error_filter [yes|no]

A value of "yes" will instruct pVAAST to filter sites with an inheritance error. Do not set this option to yes if you are interested in *de novo* mutations.

3.5. Running pVAAST

An example pVAAST command is given below:

```
VAAST -m pvaast -o <output id> -pv_control < parameter file> -p <number of threads to use> -gw
<permutation number> <gff3 file> <background cdr file>
```

The speed of pVAAST depends on (1) the number of permutations being performed (given by –gw option); (2) the number of genomes being analyzed; and (3) number of CPUs being used. The permutation number also determines the minimal achievable p-values in a pVAAST run. For example, with 1e5 permutations, the minimal p-value will be 1/1e5 = 1e-5 for a given feature. If you want to achieve p-value as low as 1e-6 for a feature, you need to perform 10 times as many permutations,

which will require approximately 10 times the amount of computational time on that feature. Therefore, we recommend using –p option to parallelize most pVAAST runs.

Under the parameter set given in the example above, pVAAST will be executed twice. In the first run, pVAAST will examine every gene using a small number of permutations. It then collects a set of reasonably significant genes, and perform a second runs using a larger number of permutations to get more accurate p-values. In the end, pVAAST will generate three useful output files: <prefix>.vaast, <prefix>.simple and <prefix.reflist>, which will be discussed in the next section.

If you only need to run pVAAST on a few genes instead of genome-wide, we suggest the following pVAAST command line:

```
VAAST -m pvaast -o <output id> -pv_control < parameter file> -q <number of threads to use> -
features <feature list> -d <permutation number> <gff3 file> <background cdr file>
```

The only differences from previous command line are in –q, –d and –features options. Under this mode, pVAAST will be only executed once and will run the full number of permutations on all genes provided to the –features option (comma separated list of mRNA names, or a single-column file containing these names). –q differs from –p in that it is optimized for smaller number of genes. In this case, -d instead of –gw specifies the number of permutations. The output files will be in the same format as before. A key advantage of this option is that it can significantly reduce the RAM usage of pVAAST, and therefore maybe preferred on smaller RAM machines.

You can find an example at: examples/pvaast_example/pvaast.sh

4. pVAAST output

VAAST writes its outputs to a file with a '.vaast' extension. This file contains a ranked list of features and other pertinent information that helps users interpret feature ranking and scores. Below is a snippet from a VAAST output file:

```
## VAAST_VERSION        2.1.1
## COMMAND       VAAST -c 16 -m pvaast -o recessive_test -pv_control pv_recessive.ctl
refGene_hg19.gff3 pvaast_background.cdr -d 2 -p 20
## TOTAL_RUN_TIME       134 seconds
## CURRENT_FOLDER       /usr/local/epi/home/hhu/apps/trunk-vaast/examples/pvaast_example
>NM_032805      ZSCAN10
chr16   -       3139092;3140647;-;chr16 3141541;3141598;-;chr16 3141765;3141829;-;chr16
3142050;3142317;-;chr16 3142543;3142773;-;chr16
TU:     8.5337201(0|0.602059988709154;1|0)      3139654@chr16   G|A     N|2|A:G|V:A
TU:     8.79578015(0|0.602059988709154;1|0)     3140183@chr16   G|H     N|2|A:G|Y:H
```

```
BR:     3139149@chr16   G|S     0-193|^:^|S:S
BR:     3139187@chr16   T|R     0-193|^:^|R:R
BR:     3139190@chr16   C|A     75|A:C|S:A
BR:     3139246@chr16   C|G     0-193|^:^|G:G
BR:     3139310@chr16   G|L     192,196|C:G|V:L
RANK:2
SCORE:18.102
genome_permutation_p:0.0099009900990099
genome_permutation_0.95_ci:0,0.036889
Running_time:3
num_permutations:101
total_success:1
LOD_SCORE:0.6021,0.0099009900990099
```

Lines starting with '##' appear at the beginning of every pVAAST output file. These lines show the version of VAAST, the command line that generated the file, the total run time and the running folder.

Information on each ranked feature appears in the lines below that. The record for a single feature appears on multiple lines in the VAAST output. Each feature record begins with a line that begins with the character '>' and continues until the next record begins or until no more records are encountered. After the '>' character on the first line the ID and name of the feature is given. In the example above the details for the mRNA NM_032805 are shown.

The second line after the ID line in the feature record describes the structure of the gene model. Three or more tab-delimited columns occur on this line. Those columns are:

1. The first column records the chromosome/contig name (matches the seqid in GFF3 and GVF files).
2. The second column designates the strand that the feature is on.
3. The third and subsequent columns each specify the structural annotation for an exon. Four values are separated by semicolons:
    a. The start coordinate of the exon
    b. The end coordinate of the exon
    c. Strand of the exon
    d. Chromosome number of the exon

The third and subsequent lines contain information on individual variants located within the given feature. These lines are split into tab-delimited columns. The description of these columns differs depending on the value in the first column. If the first column has a value beginning with T (TU,TR,T) the data on that line describes variant details for the target genomes. If the first column begins with B (BR,B) the line describes variants found in the background genomes.

Target genome columns:

1. One of TU, TR or T:
    a. TU indicates that the variant is unique to the target genomes.
    b. TR indicates that the background genomes have at least one non-reference allele at this location.
    c. T is used for insertions and deletions because VAAST does not identify overlap by position with indels and thus doesn't report if this variant is shared with the background.
2. Likelihood ratio score at this locus. The higher the score, the more likely this variant is disease-causing. A score of 0 indicates that the feature is very unlikely to influence disease under the model. This column is present only in target genome loci and only meaningful for '--mode lrt' scoring. Inside the parenthesis are the LOD scores for each pedigree at this locus. The pedigree ID (starting from 0) and the corresponding LOD score are separated by the pipe sign (|). If more than one LOD score is present, the scores are separated by semicolons (;). For example, "0-1|0.6;2|0" means that family 0 and 1 have a LOD score of 0.6 while 2 has a score of 0.
3. The 1-based genomic coordinate of this variant and the seqid separated by '@'. For example, '56289513@16' indicates position 56289513 on chromosome 16.
4. All subsequent columns provide details of individual genotypes observed at this locus within the target genomes. The representation of the genotype at this locus as provided by the CDR file. For loci in the target genomes you'll see either 'B|' or 'N|' at the beginning of genotype. A 'B' indicates that this genotype is present in the background and 'N' indicates that it is not.

For variants in the background genomes, their format is identical to the target variants except that they do not have the second column.

After all of the lines describing variant loci within the feature have printed, there are several additional lines in each record, some of which are optional:

- **RANK: T**he 0-based rank of this feature relative to all other features in the GFF3 file.
- **SCORE:** The raw CLRT score of this feature, which is the sum of CLRT scores from VAAST CLRT test and the pVAAST linkage analysis.
- **genome_permutation_p:** The significance level obtained from permutation and gene-drop simulations. This value is the most indicative value for assessing evidence of disease association for the feature. However, keep in mind that the number of genomes in your target/background CDR and the number of informative meioses in the pedigrees constrain the lowest achievable significance level.
- **genome_permutation_0.95_ci:** The 95% confidence interval of the genome permutation p-value.

- **LOD_SCORE:** The LOD score of this gene, following by the LOD score p-value.

A second file is created by pVAAST, which has a .simple extension. This contains a limited subset of the information in the main .vaast output file with one line per feature.

```
RANK    Gene    p-value   p-value-ci  Score     LOD     Variants
1        DHODH   0.0099    0,0.0369      42.71     1.204    chr16:72050942;21.39;1.20;G->A;G-
>R;0,2        chr16:72055110;17.76;1.20;G->C;G->A;0,2
```

The headers in the .simple file describe the columns. The final column may represent many variants and for each one the values given correspond to the ones given in the file above as follows:

1. **chr16:** The chromosome/contig on which the feature is found
2. **72050942:** The start location of the variant on the given chromosome
3. **21.39:** The overall score (CLRT+LOD) of this variant
4. **1.20:** The LOD score of this variant
5. **G->A:** The reference and variant nucleotides
6. **G->R:** The reference and variant amino acids
7. **0,2:** The total number of chromosomes affected (NOT the number of individuals affected) in the background and target genomes respectively. In this example, there are two copies of the variant allele in the target (either two individuals are heterozygous or one individual is homozygous for the variant allele).

The .reflist file created by pVAAST provides a useful reference for you to relate individuals in the .vaast file with IDs in the PED and CDR files. The header describes the meaning of each column.

```
##VAAST-INDEX   PED_FILE        PED_ID  CDR_FILE        CDR_ID  AFFECTED_STATUS
0       pv_recessive1.ped       A01_1   pv_recessive1.cdr       0       Unaff
```

5. VAAST options useful in pVAAST

The VAAST User Guides and our protocol paper are both great references for VAAST options. We also provide a short list of VAAST options that are relevant in pVAAST runs:

chrom

The --chrom (-c) limits VAAST to scoring only features on specified chromosome(s). The argument to this option is a comma-separated list of chromosomes. In this case only the actual

chromosome number or letter is given. For example '--chrom 4,15' will score features on chr4 and chr15 only. Specifying chromosome X and Y would be done with '--chrom X,Y' (case insensitive).

regions

Provide a file of sequence regions in BED format. Only variants within the specified regions will be scored. This option is useful for limiting the regions of VAAST scoring to targeted capture (e.g. exome capture) regions or regions of the genome that have been identified by other methods as likely harboring causal alleles.

mask_regions

Provide a file of sequence regions in BED format. Variants within the specified regions will not be scored. This option is useful for excluding regions from VAAST likely to harbor false positives (e.g. repetitive regions).

protective

By default VAAST scores only variants that are over represented in the target genomes. Specifying this option will also score variants that are under-represented in the target genomes. This may allow identification of protective alleles.

rate

The -rate option takes a real number as its argument and allows you to specify an estimate of the maximum expected disease allele frequency within the background population. By setting this option, the CLRT will be constrained such that the allele frequency of healthy genomes in the alternative model cannot exceed the specified value. The effect of this is that a variants VAAST score will be heavily penalized as the MAF of the variant in the background file exceeds the value given by --rate. The rate option requires --mode lrt. It is recommended that you always use the --rate option when a disease allele frequency estimate is available.

all_variant

By default VAAST will only score missense coding variants. The --all_variant (-e) option causes VAAST to score synonymous coding variants and variants in non-coding regions.

indel

The --indel option allows VAAST to score nucleotide insertions and deletions. Insertions and deletions, which change the reading frame of the CDS, as well as insertions and deletions >= 25 nucleotides are scored the same as a missense variant. Short in-frame insertions and deletions are scored based on the frequency of insertions/deletions of that length in a disease dataset

(OMIM/HGMD) vs. the frequency of insertions/deletions of that length in the 1000 Genomes dataset.

significance

When a real number between 0 and 1 is given as an argument to this in combination with --gp, VAAST will stop permutations when the 95% confidence interval of the p-value is above or below the given level (e.g. the genome-wide significance level), thus avoiding unnecessary permutations even when the p-value has not converged.

ref

The --ref (-a) option allows VAAST to score loci where the minor allele in the background genomes is the same as the allele in the reference genome. By default VAAST will not score these variants.

cov_matrix

This option allows pVAAST control for covariates. It takes a covariate matrix file as input. Each row of the input file represent a sequenced individual. The first column is the ID of the individual as in the FILE-INDEX line of the cdr file; each of the remaining columns has a covariate value.

6. Frequently Asked Questions

1. What can I do to make my pVAAST run is successful?

Perhaps the most essential ingredient of a successful disease gene search is a matched case and control set, both in sequencing/variant calling platforms and ethnicities. The easiest way to ensure this is to sequence your own controls and perform joint calling with your cases; if this is not possible, section 3.2 offered some practice advice on how to proceed. After you finish your pVAAST runs, we recommend you to examine the QQ-plot of the genome_permutation_p values for all the genes and make sure it looks reasonable (i.e., no significant inflation of false positive rate). This can be achieved using the vaast_reporter script (under bin/vaast_tools folder) or the Haplin package in R. In case of inflated false positive rates, another option is to use only the pVAAST LOD score for ranking. Remember that the pVAAST score has two components: The VAAST CLRT score and the LOD score. While the CLRT score is quite sensitive to platform bias and population stratification issues, the LOD score is much less so (although it is still sensitive to misspecified allele frequencies). Therefore, if you are running large pedigrees and looking for mainly linkage signals, re-ranking genes by the LOD scores and their associated p-values may alleviate the unmatched cases and controls issue.

With matched genome sets and the right combination of parameters, pVAAST is a powerful tool for disease gene discovery. We provide several different flavors of template parameter files in the data/pvaast folder, which should cover most types of pVAAST runs.

2. Can pVAAST find deleterious *de novo* mutations?

Yes. Unlike traditional linkage analysis, our statistical models also calculate LOD scores and p-values for *de novo* mutations (see *Hu et al*. *Nature Biotechnology 2014* for details). You do not need to turn on any special options in order to use *de novo* mutations; instead, it is fully integrated into both dominant and recessive inheritance modes and therefore pVAAST will also detect Mendelian inheritance signals and possible *de novo* causal mutations in your data. You do, however, need to make sure that "simulate_genotyping_error" option is turned on and "inheritance_error_filter" option is turned off. Also, you need to have a good guess of the genotyping error rate in the pedigree (genotyping_error_rate option). pVAAST also includes a script for estimating this parameter from sequenced nuclear families (bin/vaast_tools/estimate_genotype_error_rate.pl). Some template parameter file in data/pvaast folder are also tailored for *de novo* causal mutation discovery.

3. There are too many significant genes in my pVAAST run. What to do now?

Besides the advice offered in previous question "What can I do to make my pVAAST run is successful?" setting filtering parameters can also be helpful.

1. A very powerful filtering option is the –rate option in VAAST, which sets the upper limit of causal variant frequencies in pVAAST. For example, if you are looking for rare causal mutations only, setting –r 0.01 will greatly penalize variants with allele frequency greater than 0.01, which eliminates a large number of false positive signals coming from population stratification and variant calling platform differences.
2. It is also usually helpful to apply no-call filters. No-calls are genomic variants where variant callers have no confidence in calling either a variant or a reference allele, and are often correlated with low variant qualities. We suggest removing variant sites with >= 10% nocalls prior to pVAAST runs. You can also set nocall_filter option inside pVAAST parameter file, but this parameter only applies to cases.
3. If you are only interested in sites with a high linkage signal, setting "lod_score_filter" will remove most false positive signals. This option is most effective when combined with "penetrance_lower_bound" and "penetrance_upper_bound" options in highly penetrance rare Mendelian diseases.

4. In my pVAAST result, every gene has a p-value of 1 and a score of 0. Why?

There are a few things that needs to be check in this situation. First, make sure you cdr file is annotated correctly. If you do:

```
grep amino_acid_substitution <your cdr file>
```

Do you see any output? If not, then we suggest double-checking your genome build version (ie. hg18 or hg19) to make sure it is consistent with your genome annotation file (GFF3). Re-annotating the cdr file using cdr_annotater.pl (in bin/vaast_tools folder) if necessary.

If your .cdr file is normal, the next thing to check is your pVAAST parameter file (.ctl). Try lowering the penetrance_lower_bound parameter if it is set too high. Alternatively, remove lod_score_filter and clrt_score_filter and see if the issue is fixed.

5. The output of the vcf2cdr.pl script is empty.

vcf2cdr.pl internally calls vaast_converter, VAT and VST scripts to produce the cdr file. If any of these steps failed, then vcf2cdr.pl will also fail. In this situation, we suggest you to run vaast_converter, VAT and VST scripts sequentially as documented in VAAST User Guide to trouble-shoot any potential problem.

Hao Hu (haohupku@gmail.com)