A Quick-Start Guide to using…

Welcome to the VAAST quick-start tutorial. The purpose of this tutorial is get you up and running with VAAST as *quickly* as possible. Toward this end, we will walk you through the VAAST family of tools, outline some basic analysis patterns using real world examples, describe a few of VAAST's key options, and show you how to use VAAST to rapidly identify damaged genes and their disease causing variants in personal genomes data. If you have ever performed a command-line BLAST search then we think you will find using VAAST a pretty straightforward task.

Obtain a version of the VAAST package

VAAST is free for academic research use with an academic license. There is also a commercial license available for commercial applications. Please see http://www.yandell-lab.org/software/vaast.html for information about how to download a copy of VAAST.

Please note: all the examples in this tutorial use VAAST version 1.0.1. If you are not using this version of VAAST, it is possible that you may encounter backwards/forwards compatibility issues.

Installing VAAST

After you have downloaded a copy from the website, you will have a file called: VAAST_Code_1.0.1.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

You may also want to add VAAST/bin and VAAST/bin/vaast_tools to your PATH environment variable, but this isn't a requirement. ***VAAST has no external dependencies***. It should be ready to run.

Now cd to the VAAST/examples directory and you are ready to proceed with this tutorial.

Please report any errors to the mailing list.

http://yandell-lab.org/mailman/listinfo/vaast-user_yandell-lab.org

Two Essential VAAST References

A VAAST User's guide is available here: http://www.yandell-lab.org/software/VAAST_Users_Guide.pdf
A copy of the User's guide and this Quick Start tutorial is also available in the VAAST/docs directory. The User's guide is an essential companion to this Quick-Start tutorial as it provides complete documentation of all VAAST command-line options as well as VAAST's input and output formats. If you have a question about something in this tutorial, have a look at the User's Guide.

For answers to algorithmic and scientific questions, please see the primary VAAST reference:

Mark Yandell, Chad Huff, Hao Hu, Marc Singleton, Barry Moore, Jinchuan Xing, Lynn Jorde and Martin Reese (2011), ***A probabilistic disease-gene finder for personal genomes***, *Genome Research* doi:10.1101/gr.123158.111

This paper is available as a PDF here: http://www.yandell-lab.org/publications/index.html

Basic workflow

VAAST combines variant frequency data with AAS (Amino Acid Substitution) information on a feature-by-feature basis. VAAST uses the likelihood ratio described in *Yandell et al 2011 Genome Research* to search for damaged genes by comparing the variants in a set of disease genomes (cases) to those in a set of healthy genomes (controls).

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

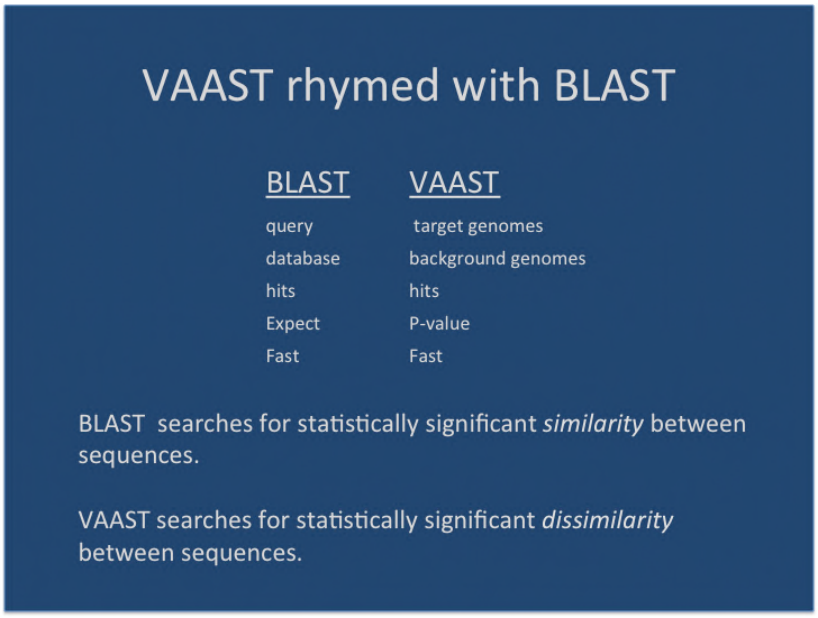


Figure 1. VAAST rhymed with BLAST. Think of a VAAST search as analogous to a BLAST search, where the target.cdr file is analogous to the query sequence in a BLAST search, and the Background.cdr file is analogous but the BLAST database. But whereas in a BLAST search you are looking for regions of statistically significant similarity between your query and the database sequences, in a VAAST search you are looking for regions of statistically significant dissimilarity (due to sequence variants) between your target and background genomes.

It's no accident that VAAST rhymes with BLAST. One of the major goals of the project has been to produce a tool that is very much like BLAST, one with which users can use personal genomes data to hunt for genes harboring disease-causing variants, in much the same way that BLAST is used to identify genes homologous to a query sequence.

The basic inputs to VAAST consist of:

1. A set of target (case) variant files in either VCF or GVF format.

2. A set of background (control) variant files in either VCF or GVF format.

3. A set features to be scored, usually genes; this file should be in gff3 format.

4. A multi-fasta file of the reference genome

Although VAAST can also employ INDELs and CNVs, the version we will be using today is restricted to SNVs. Thus your target and background variant files need not contain INDELs or CNVs.

Provided with these 4 inputs, most VAAST analyses share three basic steps in common:

1. **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.

2. **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 however call for selection of a subset of variants, for example all variants shared by affected family members, but not present in unaffected family members and VAAST has tools to do this.

3. **Variant Analysis** – All the variants found in the target genomes are then compared to all the variants in the background genomes and the features that contain those variants are scored and ranked by the likelihood that they are disease causing.

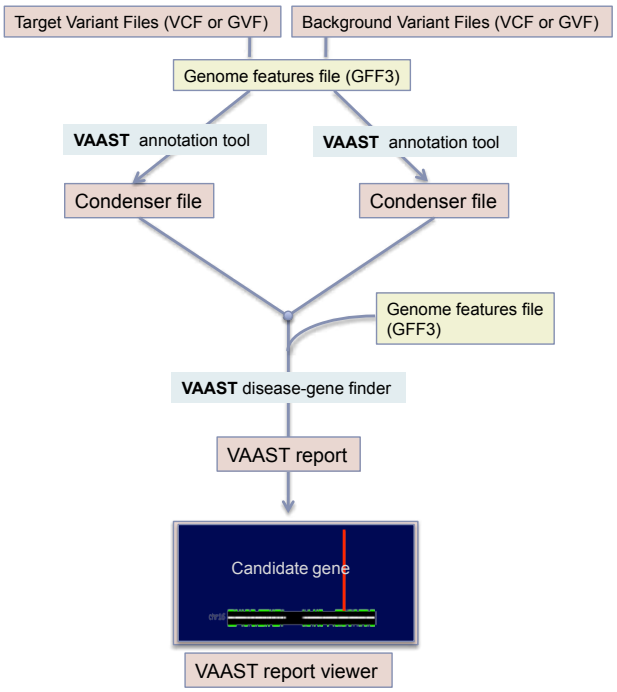


Figure 2. The VAAST search procedure. One or more variant files (in VCF or GVF format) are first annotated using the VAAST annotation tool (**VAT**) and a GFF3 file of genome annotations. Multiple target and background variant files are then combined by the VAAST selection tool (**VST**) into a single condenser file; these two files, one for the background and one for the target genomes, together with a GFF3 file containing the genomic features to be searched are then passed to VAAST. VAAST outputs a text file, which can also be viewed in the VAAST viewer.

The Exercises

The following exercises will acquaint you with each of the 3 basic steps common to almost all VAAST analyses; we will also cover some basic command-line options, and outline some basic strategies for getting the most out of a VAAST analysis.

Note: all of the exercises that we will carry out today will be relative to human genome build hg18. Remember: never mix files with coordinates relative to hg19 with hg18 or vice versa. Doing so will result in chaos!

Throughout this tutorial, command-lines will be numbered in red, *e.g.* **(1)** for easy reference. **Remember: don't include this number when pasting or typing the command.**

The Dataset. Have a look in the directory VAAST/examples/data

(1) cd VAAST/examples/data

Here you will find several files. Have a look at miller-1.gvf, miller-2.gvf, and miller-3.gvf. These files are derived from three different CEU exomes taken from the Danish Exomes dataset (PMID20890277). We have added different disease-causing variants in *DHODH* reported in individuals with Miller syndrome(Ng et al., 2010; Roach et al., 2010a) to these three variant files to produce 3 unrelated

Danish exomes, each carrying two different Miller syndrome causative alleles; non of the 3 exomes share any Miller syndrome alleles in common. These data are contained in miller-1.gvf, miller-2.gvf, and miller-3.gvf. Note that these files only contain variants on chromosome 16, so that the following examples will run as quickly as possible.

These variant files are in GVF format. GVF is the input format to VAAST for Variant Files. The formal specification of GVF can be found here:

http://www.sequenceontology.org/resources/gvf.html

Got VCF? No problem. These files can be easily converted to GVF using the vaast_converter script (VAAST/bin/vaast_tools/vaast_converter) included with this distribution; but for today we will be using GVF as our starting point.

Step 1. Variant Annotation. As we explained above, the first step in any VAAST analysis is to annotate your variant files. To do so you will also need two additional files, a multi-fasta file of the reference genome (in this case hg18), and a set of gene annotations in gff3 format. GFF3 is the most widely used format to distribute gene annotations. The formal specification can be found here: http://www.sequenceontology.org/gff3.shtml

examples/data/hg18-chr16.fasta This file contains the sequence of the hg18 reference human genome

examples/data/easy-hg18-chr16.gff3 This file contains all Chr16 hg18 gene models and transcripts downloaded from the UCSC genome browser FTP site site

NOTE: make sure you are in the ~/VAAST/examples directory before executing the command-line examples in this tutorial.

Now run the VAAST Variant Annotation Tool (**VAT**) to annotate your target GVF files.

(2) ../bin/VAT -f data/easy-hg18-chr16.gff3 -a data/hg18-chr16.fasta data/miller-1.gvf > data/miller-1.vat.gvf

(3) ../bin/VAT -f data/easy-hg18-chr16.gff3 -a data/hg18-chr16.fasta data/miller-2.gvf > data/miller-2.vat.gvf

(4) ../bin/VAT -f data/easy-hg18-chr16.gff3 -a data/hg18-chr16.fasta data/miller-3.gvf > data/miller-3.vat.gvf

Now have a look at data/miller-1.vat.gvf; compare its contents to data/miller-1.gvf.

Step 2. Variant selection. The next step is to decide which variants to include in your target dataset. The simplest approach is simply to combine (take the union of the contents) of each target (case) GVF file using the VAAST program **VST** to produce a target.cdr file for use with VAAST. The .cdr file extension is short for 'condenser format', and, as you might guess, it provides a more concise representation of your selected variants. If you have ever run a BLAST search and used the **formatdb** or **makeblastdb** programs to create a BLAST database, then this shouldn't prove too difficult. Think of **VST** as the equivalent of **formatdb** for VAAST.

Now Run the Variant Selection Tool (**VST**) to combine your target genomes and produce a formatted target cdr file (data/miller_output.cdr) for use in our VAAST searches; in keeping with our analogy to a BLAST search, you can think of this file as your query for the searches we will run today:

(5) ../bin/VST -o 'U(0..2)' -b hg18 data/miller-1.vat.gvf data/miller-2.vat.gvf data/miller-3.vat.gvf > data/miller_output.cdr

The options here are pretty straightforward. –b is the genome build, hg18 in this case; -o is a little less obvious; see the VAAST User's guide for complete complete documentation, but briefly, the command here says take the union of files 0,1, and 2. By contrast -o 'I(0..2)' would mean the intersection of the three files. See the VAAST user guide in the /docs directory for more details.

Now have a look at examples/data/189genomes-chr16.cdr. This .cdr file (already made) will be serving as the preformatted database of background genomes, which you will be searching your target genomes against. examples/data/189genomes-chr16.cdr is a combination of the first 179 genomes released from the 1000 Genome project and the 10Gen set of publically available personal genomes, which is described in:

Moore B, Hu H, Singleton M, De La Vega FM, Reese MG, Yandell M**.** *Global analysis of disease-related DNA sequence variation in 10 healthy individuals: Implications for whole genome-based clinical diagnostics* Genet Med. 2011 Mar;13(3):210-7.

Step 3. Variant Analysis. In today's helter-skelter world of genomics, personal genome and exome sequences differ dramatically as regards sequencing platform, depth of coverage, exome capture technique, completeness and variant calling procedures. The result is that systematic differences in your target and background genomes/exomes regarding any (or all) of these issues can result in false negatives and/or false positives. ***Well-matched target and background datasets are essential for carrying out a successful VAAST analyses***. The good news is that such datasets are getting easier to come by; the bad news is that the publically available datasets are still biased towards the low coverage 1000 genomes project dat. Its also still hard to assemble a large collection of high coverage genomes/exomes for specific platforms. You can measure the consistency of your background and target datasets using **quality-check.pl:**

(6)../bin/vaast_tools/quality-check.pl -bootstrap 1000 data/189genomes-chr16.cdr data/miller_output.cdr

p-value = 2.76E-1*
Mean of expected: 20.433 Standard Deviation of expected: 16.2195124335962
Observed: 26
Ratio of observed vs. expected: 1.27245142661381

*Note: this number my vary from run to run due to the random sampling algorithm employed by the program

The p-value (p-value = 2.76E-1) in the quality-check.pl output shown above is the probability that the background and target cdr files differ significantly as regards the number of rare variants. Note that in this case the background and targets are pretty will matched (there isn't a significant difference between the two). That's good. If you do see a significant difference between a background and target files, proceed with caution, as your VAAST searches may well be plagued with false positives and false negatives. For more information about these issues see the VAAST paper in *Genome Research*, especially that paper's figure 3 and its accompanying text.

Now that we have our background (controls) and targets (cases) properly annotated and formatted as cdr files, the next step is to search the targets for damaged genes and their disease-causing variants. To do so, run VAAST:

(7) ../bin/VAAST -iht r -lh n -fast_gp -d 1e4 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

That's a pretty hairy command-line, so before having a look at the results of our search, let's make sure we understand the search parameters. The options are summarized below; more complete information, however, can be found in the VAAST users guide and the supplemental file accompanying the VAAST paper published in *Genome Research*.

-iht. The --inheritance (-iht) option allows you to inform VAAST about the inheritance model. This option takes as its argument either d (dominant) or r (recessive). When dominant is specified only the best-scoring variant in each feature in each individual will be scored. When recessive is specified, only the best-scoring homozygous variant or the two best-scoring heterozygous variants in each feature in each individual will be scored. Miller Syndrome is a recessive disease, so we will set –iht r

-lh The --locus_heterogeneity (-lh y) option indicates that VAAST should allow locus heterogeneity, i.e. not every member of the target set is required to have the *same* damaged gene. Allowed values for arguments to this option are y (yes) and n (no). The default is "--locus_heterogeneity y". When "--locus_heterogeneity n" is specified under a dominant model, VAAST will require every target genome to have at least one variant with a score greater than 0 in any putative disease-causing gene (i.e. every individual must have at least one variant in the candidate gene with a score > 0). When "--locus_heterogeneity n" is specified under a recessive model, VAAST will require every target genome to have the *same* damaged gene, e.g. at least two variants with a variant score greater than 0 in a putative disease-causing gene. If these criteria are not met, the entire gene will receive a score of zero. Note that this is a stringent filter that will cause a true disease gene to be missed if just one individual in the target is heterogeneous (has the disease due to alleles in a different gene from the rest of the target individuals, or if any of the targets are incorrectly genotyped); this thus -lh is set to 'yes' by default. Since we know all three of our target targets harbor alleles in DHODH (the Miller Syndrome gene), we will set –lh to 'n' for now.

-fast_gp this option speeds things up a bit, but requires a bit more ram; see the user's guide in /docs for more information

-d specifies the number of permutations to perform. Setting –d to 1e4 instructs VAAST to perform 10,000 permutations when calculating the statistical significance of the results. We will discuss this in more detail in the examples that follow, but for now we have set it to 1e4; this is a small number of permutations, so VAAST will run quickly. For a complete explanation of the VAAST permutation procedure, see the *Genome Research* paper.

-r sets the maximum combined population frequency for the disease-causing alleles. For Miller Syndrome this value was calculated based upon the observed incidence of the disease worldwide; that's r^2 or $1/8.1X10^6$ births, about 900 people.

-o test tells VAAST to write the output to a file called test.vaast

-m lrt instructs VAAST to score genes using the CLRT method described in *Yandell et al 2011, Genome Research*. Other scoring methods are available; see the user's guide in /docs and the *Genome Research* paper for more details.

-k tells VAAST to use AAS (Amino Acid Substitution) information in addition to variant frequencies when scoring coding variants. Note that this option requires that you first have run **VAT** to annotate your GVF files, prior to making the target and background .cdr files using **VST**.
.

The basic output file test.vaast, provides a lot of basic information about the search, the variants and the genes scored; this format is analogous to a standard BLAST report, and like a standard BLAST report it's a bit complicated; ***the .vaast output format is fully described in the VAAST User's guide.*** VAAST also comes with a script that converts this output format to a simpler, Microsoft Excel compatible, tab-delimited file. Let's try it:

(8) ../bin/vaast_tools/simple_output.pl test.vaast | more

```
RANK Gene    p-value                Score         Variants
1    DHODH   8.61118758682451e-07   52.95337954   chr16:7061493610.86;C->T;R->W;0,1 …
2    LDHD    1                       0
3    CPNE2   1                       0
4    NUBP2   1                       0
5    CHP2    1                       0
…
736   AARS   1                       0            chr16:68852548;0.50;G->A;T->I;1,1
```

The first column gives the gene's, rank; just like in a BLAST report the best hit is first in the list.

The second column is the gene name.

The third column its the p-value, again just like a BLAST E-value, the lower the better.

The fourth column is the VAAST score for the gene; the higher the score the more likely that gene is to contain disease causing variants

The fifth column provides a list of the most likely disease-causing variants (as judged by VAAST) in the gene; each variant is separated by whitespace, and these are further delimited by semi-colons. The first semi-colon delimited field is the variant's location, the second the VAAST variant score (the greater the value, the more likely the variant is to be disease causing); the next field is the nucleotide change; the next the Amino-acid change (if non-synonymous); the last field give the number of counts for the variant in the background file, and the target file. For example, the variant located in DHODH at position chr16:7061493610 is seen 0 times among the background genomes, and 1 time among the target genomes.

Notice that there is only a single candidate, DHODH, out the 736 annotated protein-coding genes lying on Chromosome 16. That's perfect accuracy with only three unrelated individuals; no two of which share a DHODH disease causing allele in common. Try that with GWAS ;-)~

Let's see how we do, if we perform more naïve searches having fewer with fewer assumptions. First try running VAAST without the –r option

(9) ../bin/VAAST -iht r -lh n -fast_gp -d 1e6 -o test -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

Let's look at the results; first try the simple output script:

(10) ../bin/vaast_tools/simple_output.pl test.vaast | more

```
RANK    Gene      p-value                  Score           Variants
1       DHODH     8.61118758682451e-07     52.95337954     chr16…
2       ZCCHC14   0.00645246846637959       9.83531668     chr16…
3       DPEP2     0.0400008266740083       10.84866484     chr16…
4       LDHD      1                         0
5       CPNE2     1                         0
6       NUBP2     1                         0
7       CHP2      1                         0

…
736     AARS      1                         0    chr16:68852548;0.50;G->A;T->I;1,1
```

As you can see, even without making any prior assumptions about the population frequency of the disease causing allele(s), we still have only 3 candidates, and only DHODH comes anywhere near having a genome-wide significant P value ($P < 2.410^{-6}$). ***Which raises an important point: just what is a genome-wide significant P-value?*** To determine genome-wide significance we must correct for the number of features scored, the so-called *Bonferroni* correction. Assuming the human genome contains ~21,000 genes, $P < 0.05$ certainty would correspond to (alpha = 0.05/21,000) = $2.4x10^{-6}$, so the DHODH p-value (8.61118758682451e-07) is genome-wide significant and then some. The next best hit (ZCCHC14) with a p-value of 6.4e-03* misses the alpha by a long way. Thus even without setting a causal allele threshold (-r) we still get only one genome-wide significant VAAST hit, and it's the correct gene, DHODH.

*Note: as we explain below, ranks and p-values may vary slightly from run to run.

Now let's run VAAST without ***any*** prior assumptions about the genetic nature of Miller Syndrome. We will drop the –r option, forgo specifying an inheritance model, and allow locus heterogeneity. For this run we will increase the number of permutations to 1 million so that we will have better statistical confidence; also we will parallelize the search and set a significance threshold using the –significance, –mp1 and –fast_gp options, so that our 1 million permutations won't take too long (these options will be explained in a moment). Other than the options to speed it up, this is running VAAST in its most assumption free mode:

(11) ../bin/VAAST -o test -mp1 4 -d 1e6 --significance 2.4e-06 -fast_gp -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

Let's see what we got:

(12) ../bin/vaast_tools/simple_output.pl test.vaast | more

```
RANK    Gene      p-value                  Score           Variants
1       DHODH     8.61118758682451e-07     52.95337954     chr16…
2       CENPN     0.000159591140803324     23.63636624     chr16…
3       CLDN6     0.000159591140803324     23.40553616     chr16…
…
736     AARS      1                         0              chr16…
```

Not bad. DHODH, is still the top scoring hit and it's the only hit to obtain a genome-wide significant p-value. It's important to grasp that even without specifying an allele frequency, an inheritance model and allowing locus heterogeneity, there are still only 47 genes (out of 737 genes screened) with scores

greater than 0, and only DHODH has a genome-wide significant p-value. **This ability to find the damaged gene with great accuracy and using no prior assumptions demonstrates what we refer to as VAAST's *ab-initio* strength**.

Getting better performance out of VAAST.

VAAST is a multi-threaded application. If you have a multi-core computer, this feature of VAAST can be used to dramatically speed your searches. This feature of VAAST not only speeds things up, it also let's you look harder for disease-genes. Let me explain. First let's run a search again, and this time take a closer look at the test.vaast output file:

(13) ../bin/VAAST -iht r -lh n -fast_gp -d 1e4 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

Now have a look at test.vaast; specifically the following three fields for the top-scoring hit, DHODH:

(14) more test.vaast

>uc002fbp.1:mRNA DHODH
SCORE:52.95337954
genome_permutation_p:8.61118758682451e-07
genome_permutation_0.95_ci:8.61118758682451e-07,0.000291325435563329

Since we have only done a set number of permutations (in this case –d 1e4), VAAST reports the most likely P value given the results of these 10,000 permutations. In this case:

genome_permutation_p:8.61118758682451e-07

This is the VAAST p-value for this feature. Genome-wide significance requires that we correct for the number of features scored, the so-called *Bonferroni* correction. Assuming the human genome contains ~21,000 genes, $P < 0.05$ certainty would correspond to (alpha = 0.05/21,000) = 2.4×10^{-6}, so this result is genome-wide significant.

Although this is VAAST's best guess, the actual P-value (if you had done say $1e^{1000}$ permutations) is unknown. The 95% confidence interval for the true p-value is however is knowable; VAAST uses a Poisson-based approximation and reports these values too:

genome_permutation_0.95_ci:8.61118758682451e-07,0.000291325435563329

In other words, 8.6e-07 is the most likely p-value based upon the 1e4 permutations we did; but the true p-value is 95% likely to lie between 8.6e-07 and 2.9e-04. Note that the upper bound isn't anywhere near genome-wide significant, even though the most-likely p-value is. Confused? Hopefully the following examples will clarify matters.

First let's see what happens when we up the number of permutations to 1 million:

(15) ../bin/VAAST -iht r -lh n -fast_gp -d 1e6 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

Now have a look at the top-scoring feature in test.vaast:

>uc002fbp.1:mRNA DHODH
SCORE:52.95337954
genome_permutation_p:8.61118758682451e-07
genome_permutation_0.95_ci:8.61118758682451e-07,3.11278012926111e-06

Notice how we have now bracketed that p-value within a much smaller interval; P< 3.1e-06 with 95% confidence. This is because performing more permutations allows VAAST to better bracket the variance in its p-value calculation.

So 1 million permutations took a while (181 seconds on my quad core MacBook Pro); and we still didn't quite achieve P < 2.4e-06 with 95% confidence. One option would be to up –d to something like $1e^{10}$ permutations, but that would take *quite* a while. Fortunately VAAST has an option to help this situation: -—significance; by setting this to genome-wide significance (2.4 e-06), VAAST will permute to a max of -d permutations, but once the value of --significance has been obtained with 95% confidence it will stop; so there is no wasted time:

Let's do up to $1e^{10}$ permutations, but stop once we have genome wide significance or better for a feature:

(16) ../bin/VAAST -iht r -lh n -fast_gp -d 1e10 --significance 2.4e-6 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

>uc002fbp.1:mRNA DHODH
SCORE:52.95337954
genome_permutation_p:8.61118758682451e-07
genome_permutation_0.95_ci:8.61118758682451e-07,1.98691508529734e-06

That's got it. Best guess p-value is 8.6e-07; with 95% confidence it's below 1.98e-06. But notice it took a while to run; 186 seconds on my quad-core MacBook Pro.

Another approach is to first find promising candidate(s), and then do permutations only on those feature(s):

(17) ../bin/VAAST -iht r -lh n -fast_gp -d 1e5 --significance 1e-4 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

That was fast (10 seconds), and clearly DHODH (uc002fbp.1) looks promising; so let's bracket its p-value using the --features option. When invoked, this option tells VAAST to only permute the features contained in the comma-separated list following the option (see user's guide in /docs for more details):

(18) ../bin/VAAST -iht r -lh n -fast_gp -d 1e10 --significance 2.4e-6 --features uc002fbp.1:mRNA -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

So that took 171 seconds. We can do all of this in still less time, by taking advantage of VAAST's multi-threaded capabilities. Lets parallelize our permutations to speed things up. For our first search, we will tell VAAST to parallelize these individual gene permutations, using the –mp1 option set for 4 CPUs (the number of core's on a typical MAC laptop); this will speed up the search by a maximum of 4-fold:

(19) ../bin/VAAST -iht r -lh n -fast_gp -d 1e5 --significance 1e-4 -mp1 4 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

That took 12 seconds. In this case it wasn't any faster because only DHODH required anything close to 10,000 permutations before the lower-bound of its p-value was established. If we were searching the entire genome, and/or our datasets were noisier, we would have seen an approximately 4-fold speed up.

Now select some interesting candidate genes after inspecting the output file (in this case one, uc002fbp.1:mRNA, aka DHODH), and use 4 CPUs to in order to rapidly bracket the feature(s)' p-values:

(20) ../bin/VAAST -iht r -lh n -fast_gp -d 1e10 --significance 2.4e-6 -mp2 4 --features uc002fbp.1:mRNA -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/miller_output.cdr

>uc002fbp.1:mRNA DHODH
SCORE:52.95337954
genome_permutation_p:8.61118758682451e-07
genome_permutation_0.95_ci:8.61118758682451e-07,1.98691508529734e-06

That's got it nailed, and in only 108 seconds on my quad-core MacBook Pro, compared to 181 without using –mp2. That may seem a modest speed up, but if you were checking 100 features using the above command-line, you would definitely notice the difference: 3 hours with -mp2 4, versus nearly 3 *days* without.

How many permutations is enough? The above analyses naturally raise questions, such as how many permutations is enough? Likewise, is it possible to perform too many permutations? Fortunately there are good answers to both these questions. It is definitely possible to do too few permutations; in fact you should watch out for this. If you do fewer than 1e-05, you may notice that both the gene ranks and their p-values will jump around quite a bit between runs, *even when using the exact same command-line and datasets!* This is because from one run to the next—even though the data and options haven't changed—each gene's p-value will jump around within its genome_permutation_ci interval. If this interval is big—and as we have seen, it often is when –d is less than 1e-05—gene ranks and p-values will fluctuate from run to run. So for most applications –d 1e04 is fast, but also a bit dangerous. It is also possible to perform too many permutations, although this is less harmful (it just wastes time). The reason is that for a given number of background and target genomes, there are only a limited number of unique combinations to permute. This number is governed by the n-choose-k equation, *e.g.* the number of unique permutations = n!/(k!(n-k)!. With 189 genomes in the background and 3 genomes in the target, n = 192, and k = 3; thus there are 1,161,280 different unique combinations of three genomes to permute. You might be inclined to set –d equal to n!/(k!(n-k)!; but this isn't optimal either because in practice you will have to carry out more permutations than this in order to randomly sample *every possible permutation*. With these considerations in mind, I recommend setting –d to 10X the number of unique permutations (with an upper bound of 1e10), and using the --significance option set to the genome-wide alpha of 2.4e-06. This will give good accuracy for the gene ranks and their p-values, and waste a minimum of CPU cycles.

When to worry. Mismatched cases and controls are the Achilles heel of every sequence-based hunt for disease-causing variants. VAAST is no exception. With well-matched background and target datasets, VAAST is both accurate and fast, and even when the datasets are only moderately well matched, performance is still very good. We've seen this fact demonstrated in this tutorial, and the more extensive analyses presented in the *Genome Research* paper also show that VAAST is resistant to the complications introduced by moderately mismatched background and target datasets (see figure

3 in the paper). That said, not all personal genomes/exome datasets are created equal. Different sequencing platforms, depth of coverage and variant calling procedures, not to mention population stratification, can create situations where the target and background datasets are so mismatched as to preclude useful analyses. Badly mismatched targets and backgrounds create not only false negatives, they also produce false-positives. The first step in determining if your datasets are suitable for VAAST analyses is to use quality-check.pl to get a rough indication of whether or not your datasets are suitable for a VAAST analysis. Consider the results below:

(21) ../bin/vaast_tools/quality-check.pl -bootstrap 1000 data/189genomes-chr16.cdr data/miller_output.cdr

p-value = 2.76E-1
Mean of expected: 20.433 Standard Deviation of expected: 16.2195124335962
Observed: 26
Ratio of observed vs. expected: 1.27245142661381

The above output is what we would term a pretty good match— the p-value suggests that the background and targets were drawn from the same variant population distribution, and that the target genomes only contain a slight excess of variants relative to the background genomes (1.27-fold). ***As a rough rule of thumb, if the p-value is less than 0.0005 and the ratio of observed to expected is less than 0.2 you should be seriously concerned about false-negatives due to missing data. If the p-value is less than 0.0005 and the observed to expected ratio is greater than 5, you should be seriously concerned about false-positives.*** Keep in mind that these false-positives can be especially misleading, as the excess of rare variants in your targets will likely conspire to grant them very low VAAST p-values, often well below genome-wide significance. A large excess of variants in your targets relative to backgrounds can also lead to extremely lengthened compute times. The reason is that VAAST will be preforming a larger number of permutations on many more candidate-genes—this can take a while; with runs that should take minutes stretching in to hours or days. If this is the case you should consider recalling your variants using more conservative parameters, or try to obtain a better-matched background file.

In extreme cases, VAAST will actually terminate the search itself, sending the following error message:

********** Warning!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! **********
VAAST has detected a large excess of features (genes) with genome-wide significant scores. This is almost certainly a data quality issue. It might be due to significant population stratification and/or differences in sequencing platform/variant-calling procedures between your background and target genomes. We strongly recommend that you attempt to obtain background and target genome sets that are in better sync with one another with regards to genome-wide variant frequencies!
********** Warning!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! **********

If you see this message you should face the fact that you have a serious problem. The mismatch between your background and target datasets is simply *terrible*; in some cases, obtaining a new background dataset, one better matched with respect of ethnicity, sequencing platform, exon-capture technique, depth of coverage and variant calling procedures may solve the problem; in other cases, recalling your genomes' variants or even re-sequencing your target genomes will likely be the only path to real results.

Analysis of a Family's genomes

Family genomes datasets often take the form of trios (mom, dad and affected child), quartets (mom, dad, and two sibs, sometimes both affected, sometimes not), and so on. VAAST provides a variety of ways to explore these datasets. VAAST can also be used for analyses of more complex and extended pedigrees. For brevity's sake, however, in what follows we will explore only the most basic of family centered VAAST analyses, using the –t, or --trio option.

The **--trio** (or -t) option removes variants that are found in the affected childrens' genomes but not found in their parents, as these are likely to be sequencing errors. Of course *de novo* mutations in the affected genomes will be filtered out too, so this is not a good option to use if you suspect the disease-causing allele(s) may be *de novo* mutation(s). However, since the rate of *de novo* mutations is low relative to sequencing error rates (about 70 *de novo* mutations compared over 10,000 errors per genome) this option, when used can greatly improve the specificity of VAAST searches. The **--trio** option also works synergistically with the **--penetrance** option. When **'--penetrance c' (-pnt c)** is specified, VAAST will include the parental genomes among the background genomes for the purposes of evaluating penetrance criteria (see **-pnt** description provide above).

For our analyses today we will use chromosome 16 sequences of the Miller Utah quartet as reported in (Ng et al., 2010; Roach et al., 2010a). These four genomes consist of a mother, father and their two children. Both the children are affected with Miller Syndrome and are compound heterozygotes or, if you prefer 'trans-hets', to use the traditional genetics terminology to describe their genotypes. In other words, they both inherited one deleterious DHODH allele from their father and a second allele (at a different position in the gene from the first) from their mother (see **figure 3 below**). The /data directory contains 4 GVF files corresponding to the quartet:

data/Miller_Syndrome_B_chr16.gvf is the daughter
data/Miller_Syndrome_C_chr16.gvf is the father
data/Miller_Syndrome_D_chr16.gvf is the son
data/Miller_Syndrome_E_chr16.gvf is the mother

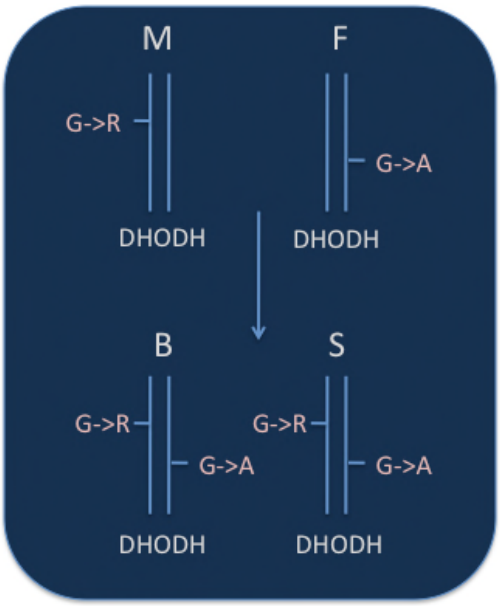


Figure 3. Genotypes of the Miller Syndrome Quartet at the *DHODH* locus on chromosome 16. M: mother, F: father, B: brother, S: sister. The facial and limb dismorphologies characteristic of Miller Syndrome arise from compromised copies of *DHODH*, a gene involved in pyrimidine metabolism (Ng *et al,* Nature Genetics 42, 30–35 (2010); Roach, *et al*, *Science* 328 636, 2011). Both affected siblings are compound heterozygotes (trans-hets) for damaging alleles in the *DHODH* locus. Thus function is not compromised by a single homozygous variant. Situations such as this are very hard to detect using traditional GWAS approaches.

Lets try a VAAST analysis of the Miller Utah quartet. The first step is to annotate each GVF file using **VAT**:

(22) ../bin/VAT -f data/easy-hg18-chr16.gff3 -a data/hg18-chr16.fasta data/Miller_Syndrome_B_chr16.gvf > data/Miller_Syndrome_B_chr16.vat.gvf

(23) ../bin/VAT -f data/easy-hg18-chr16.gff3 -a data/hg18-chr16.fasta data/Miller_Syndrome_C_chr16.gvf > data/Miller_Syndrome_C_chr16.vat.gvf

(24) ../bin/VAT -f data/easy-hg18-chr16.gff3 -a data/hg18-chr16.fasta data/Miller_Syndrome_D_chr16.gvf > data/Miller_Syndrome_D_chr16.vat.gvf

(25) ../bin/VAT -f data/easy-hg18-chr16.gff3 -a data/hg18-chr16.fasta data/Miller_Syndrome_E_chr16.gvf > data/Miller_Syndrome_E_chr16.vat.gvf

Note, you will see some warnings here—these are cases where the sequenced individual has an 'N' at a position in their genome corresponding to a codon; in some of these cases VAT is unable to determine the Amino acid, hence the warnings.

The second step is to use **VST** to make a target .cdr file consisting of the *intersection* of two affected children, *i.e*. all variants shared in common. Notice that previously we used VST to take the union of the target GVF files. But in this case it makes sense to take the intersection, as both children will almost surely have inherited the same disease causing alleles from their parents—one of the basic simplifying assumptions that family studies such as these make possible. The advantage of this assumption is that it can be used to restrict the number of variants to analyze, decreasing the possibility of false positives in our VAAST search:

(26) ../bin/VST -o 'l(0..1)' -b hg18 data/Miller_Syndrome_B_chr16.vat.gvf data/Miller_Syndrome_D_chr16.vat.gvf > data/Miller_Syndrome_BD.cdr

Now let's check and see of our target and background files are well matched; for these analyses we will use the same background file we used in our previous analyses above, namely data/189genomes-chr16.cdr:

(27) ../bin/vaast_tools/quality-check.pl -bootstrap 1000 data/189genomes-chr16.cdr data/Miller_Syndrome_BD.cdr

p-value = 1.00E-1
Mean of expected: 14.818 Standard Deviation of expected: 13.8177351908988
Observed: 36
Ratio of observed vs. expected: 2.42947766230261

In this case, (p = 0.1) there doesn't seem to be a significant excess of variants in our target file. In a sense this isn't surprising as we are looking only at the intersection of the two siblings—but that's what we wanted, as we have likely thrown out a lot of sequencing and systemic errors in these two genomes by taking the intersection. Even so, although it's not a statistically significant excess, the target file still contained over twice the number unique variants one would expect based on the variant frequencies in the background file!

Let's see what happens if we do the union of the two siblings:

(28) ../bin/VST -o 'U(0..1)' -b hg18 data/Miller_Syndrome_B_chr16.vat.gvf data/Miller_Syndrome_D_chr16.vat.gvf > data/Miller_Syndrome_BD_union.cdr

(29) ../bin/vaast_tools/quality-check.pl -bootstrap 1000 data/189genomes-chr16.cdr data/Miller_Syndrome_BD_union.cdr

p-value = 2.00E-3
Mean of expected: 14.46 Standard Deviation of expected: 13.7133678013994
Observed: 74
Ratio of observed vs. expected: 5.11756569847856

As you can see, the Utah Miller Siblings' files contain quite an excess of rare variants compared to our background genomes (~5X); and it's statistically significant as well (p = 0.002). This is because our background genomes are mostly taken from the original 1000 genomes project release. These are low coverage Illumina genomes, and hence lack a substantial fraction of rare variants. Our target genomes, on the other hand, are deeply sequenced Complete Genomics sequences, and it would appear are significantly enriched for rare variants compared to the background set. Also worrisome is the possibility that the different systematic error profiles characteristic of the two different sequencing platforms may also cause us problems. The overarching fear here is that this incompatibility—whatever the cause(s)—between targets (cases) and background (controls) will result in false positives. Let's see what happens if we run first run VAAST, without the –t option, for the moment disregarding the fact that we can also use the parents' genomes in the analyses:

(30) ../bin/VAAST -iht r -lh n –fast_gp -d 1e10 --significance 2.4e-6 -mp1 4 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/Miller_Syndrome_BD.cdr

First let's have a look at the simplified output:

(31) ../bin/vaast_tools/simple_output.pl test.vaast | more

RANK*	Gene	p-value	Score	Variants
1	DHODH	5.51116006856154e-05	37.00561330	chr16…
2	SULT1A2	5.51116006856154e-05	33.69885242	chr16…
3	ZFHX3	5.51116006856154e-05	23.04179631	chr16…
4	HS3ST4	5.51116006856154e-05	17.45342674	chr16…
5	BCAR1	5.51116006856154e-05	10.58231166	chr16…
6	PRR14	5.51116006856154e-05	7.38953997	chr16…
7	SULT1A1	5.51116006856154e-05	7.38953997	chr16…
8	WDR90	5.51116006856154e-05	7.38953997	chr16…
9	PIGQ	0.0200540093686719	7.38953997	chr16…
10	GCSH	0.030053458252665	7.38953997	chr16…
11	LDHD	1	0	
12	CPNE2	1	0	

*Note: you may see slightly different ranks and p-values, for reasons explained above.

Well, it's not too bad. As you can see there are 8 candidate genes, all with the same P-value, 5.51e-05. Why is this so? Why 8 genes, none with genome-wide significance, whereas in our previous searches using three unrelated individuals as targets gave us a single candidate-gene, and with genome-wide significance? There are four factors here, each combining with the others to produce false-positives.

First, the two target genomes are siblings; this means that about ¼ of even their rarest variants will be shared in common; whereas for two unrelated individuals the fraction of shared rarest variants is vanishingly small by comparison. Our VAAST search didn't take the relatedness of the two sibs into account; it thus mistakenly overestimated the significance of these shared variants—the result being false positives.

Second, as we discussed above, there is an excess of rare variants due to bias in these targets compared to our background file, which is missing many rare variants because its largely based on the low-coverage 1000 genomes data; this means that these rare variants will receive inflated scores, because they will appear to VAAST to be unique variants, absent from the background file, but shared in common between the targets; red-hot candidates for disease-causing variants. A better background file would solve this problem… if we had one.

Third, most of our background file was sequenced using the Illumina platform, our targets using the Complete Genomics platform. Every sequencing platform has certain intrinsic biases (so called 'systematic errors'), those common to the Complete Genomics platform, but not the Illumina platform will stand out as potential rare disease causing alleles—generating false positives in our VAAST search.

Finally, there is a fourth factor here that is being introduced by VAAST due to our limited number of genomes. Note that 8 of the candidate-genes have the identical p-value of 5.1e-05. This is the best statistical power VAAST can obtain using only two target genomes; the reason is that VAAST uses an n-choose-k approach when permuting the genomes, with 189 genomes in the background, and 2 genomes in the target in the target, n = 191, there are only n!/(k!(n-k)!, or 18,145 different combinations to permute, so the best possible P-value obtainable is 1/18,145, or 5.1e-05. In our previous examples using 3 target genomes, the number of possible permutations was 1,143,135, so the best possible p-value was 8.6e-07; which is what we got. In other words with 189 background genomes, and only two unrelated targets, our analysis is statistically underpowered--- the best p-value we can hope for is 5.1e-05, which isn't genome-wide significant; the fact that the two targets are siblings only makes matters worse; and the mismatched, Illumina-biased background and complete-genomics-biased target files, makes matters worse still.

So how do we overcome this problem? The answer is to use the parents' genomes and VAAST's –t option. Before going further however, I would like to introduce the **VAAST viewer**. When your results are complex, and contain many candidates, this viewer can prove very helpful by allowing you to get a quick overview of the data.

To access the VAAST viewer, paste the following URL into your browser; the username and password are also given below.

http://vaast.malachite.genetics.utah.edu/cgi-bin/vaast.cgi

Scroll down to the bottom of the page; there you will see an upload button; but *first* click the 'repeats checkbox'. Also, before uploading your file, please do two very important things

1. Prepend a 4-letter prefix of *your own choosing* to it, e.g. **test.vaast** becomes **wxyz_test.vaast**. The reason for this is that if multiple users, all logged on as 'cshl', try to upload files with the same name, they will overwrite one another's file:
(32) mv test.vaast wxyz_test.vaast

2. gzip it for faster service; to do so, use the following command:

(33) gzip wxyz_test.vaast.

Your file will now be called wxyz_test.vaast.gz Upload this file. Now scroll down to look at CHR16. You should see something like the figure below:

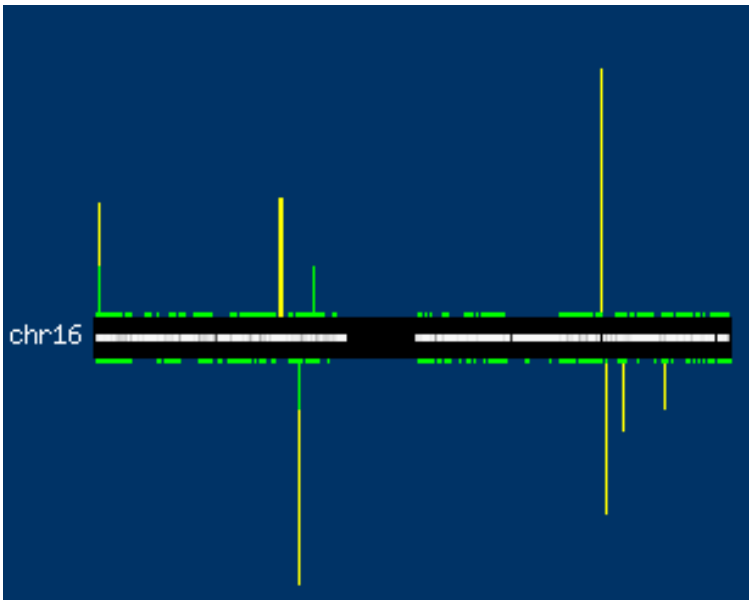


Figure 4. test.vaast visualized using the VAAST Viewer. Grey bar along the center of the chromosome shows the proportion of unique sequence along the chromosome arms, with white denoting completely unique sequence; black regions thus outline centromeric regions. Colored bars above and below the chromosome (mostly green) represent each annotated gene; plus strand genes are shown above and minus strand genes below; their width is proportional their length on the chromosome; height of bar, is proportional to their VAAST score. Genes with genome-wide significant p-values colored red (p < 2.4e-06), those with p-values less than 1e-03 but greater than 2.4e-06 are shown in yellow; those with p-values greater than 1e-03 in green.

Now lets see what happens if we take full advantage of the quartet of exomes. First we need to make a CDR file for parents. For this we will want the union of the two parents; use VST to make this file:

(34) ../bin/VST -o 'U(0..1)' -b hg18 data/Miller_Syndrome_C_chr16.vat.gvf data/Miller_Syndrome_E_chr16.vat.gvf > data/Miller_Syndrome_CE_union.cdr

Now run VAAST using the –t option with the .cdr file containing the union of the two parents' annotated variants as the argument; in this run we will also use the –pnt or penetrance option. By setting -pnt to 'c' for complete, together with –iht r we are assuming that the disease is recessive and 100% pentrant—both good bets as both parents are phenotypically normal and both siblings are clearly affected:

(35) ../bin/VAAST -t data/Miller_Syndrome_CE_union.cdr -pnt c -iht r -lh n -fast_gp -d 1e10 --significance 2.4e-6 -mp1 4 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/Miller_Syndrome_BD.cdr

(36) ../bin/vaast_tools/simple_output.pl test.vaast | more

```
RANK    Gene    p-value                 Score           Variants
1       DHODH   5.51116006856154e-05    37.00561330     chr16…
2       ZFHX3   5.51116006856154e-05    23.04179631     chr16…
3       LDHD    1       0
4       CPNE2   1       0
…
```

Let's also upload test.vaast to the VAAST viewer, remembering to gzip it first and to the click 'repeats check box button' on VAAST viewer as well:

Well, that's better, now there are only two gene candidates, and DHODH is one of them. Not bad, but we can do better. Lets try VAAST's 'masking' functionality.

As we explained above, the limited number of personal genomes available today often necessitates comparisons of genomes sequenced on different platforms, to different depths of coverage, and subjected to different variant-calling procedures. These factors can be a major source of false-positives in disease-gene searches. Based upon an analysis of these data, we have found variant-calling errors to be over-represented in low-complexity and repetitive regions of the genome, which is not unexpected. We therefore developed a VAAST runtime option for masking variants within these regions of the genome. VAAST users specify a read length and paired or un-paired reads. VAAST then identifies all variants in non-unique regions of the genome meeting these criteria and excludes them from its calculations. Let's try it.

The first step is to download a masking file from the VAAST website:

(37) wget http://www.yandell-lab.org/software/VAAST/data/hg18/Variant_Masking/35bp_se.bed.gz

In this case we chose the 35bp_se.bed.gz, as these short reads correspond to the read lengths for the target exomes in our current analysis. Place this file in your VAAST data directory. Now gunzip the file and use it as the argument to the –x option with VAAST:

(38) gunzip data/35bp_se.bed.gz

Now run VAAST:

(39) ../bin/VAAST -x data/35bp_se.bed -t data/Miller_Syndrome_CE_union.cdr -pnt c -iht r -lh n -fast_gp -d 1e10 --significance 2.4e-6 -mp1 4 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/189genomes-chr16.cdr data/Miller_Syndrome_BD.cdr

Humm… still two candidates; so in this case, masking didn't help, still it's usually worth a try. So what is that second candidate? Maybe it's a gene that was missed in the published analyses? This is a possibility, but in this case, its more likely a false positive, due to the underrepresentation of rare variants in our background file, that we mentioned previously. Actually, there are other pre-complied background files available for VAAST; let's try a slightly larger one that contains alleles from dbSNP

with a MAF < 0.05. It's a bit of a hack, as we don't know how many individuals the dbSNP data are derived from in every case, but it often proves a useful one.

The file is data/196-new-hg18_chr16.cdr. Let's use quality-check.pl to see if it's a better match for our targets:

(40) ../bin/vaast_tools/quality-check.pl -bootstrap 1000 data/196-new-hg18_chr16.cdr data/Miller_Syndrome_BD_union.cdr

p-value = 6.20E-2
Mean of expected: 40.854 Standard Deviation of expected: 111.503245054382
Observed: 63
Ratio of observed vs. expected: 1.54207666323983

For reference, lets run the data/189genomes-chr16.cdr comparison:

(41) ../bin/vaast_tools/quality-check.pl -bootstrap 1000 data/189genomes-chr16.cdr data/Miller_Syndrome_BD_union.cdr

p-value = 2.00E-3
Mean of expected: 15.035 Standard Deviation of expected: 13.8913190066222
Observed: 74
Ratio of observed vs. expected: 4.92184901895577

With p = 0.062 and O/E = 1.54, data/196-new-hg18_chr16.cdr seems a much better match for our target.cdr file than data/189genomes-chr16.cdr (p= 0.002 O/E = 4.92). Even though it's a bit of a hack, adding the dbSNP data really seems to have helped as regards rare variants not identified in the 1000 genomes project data. Now lets try our VAAST search again, this time using data/196-new-hg18_chr16.cdr rather than data/189genomes-chr16.cdr for our background file:

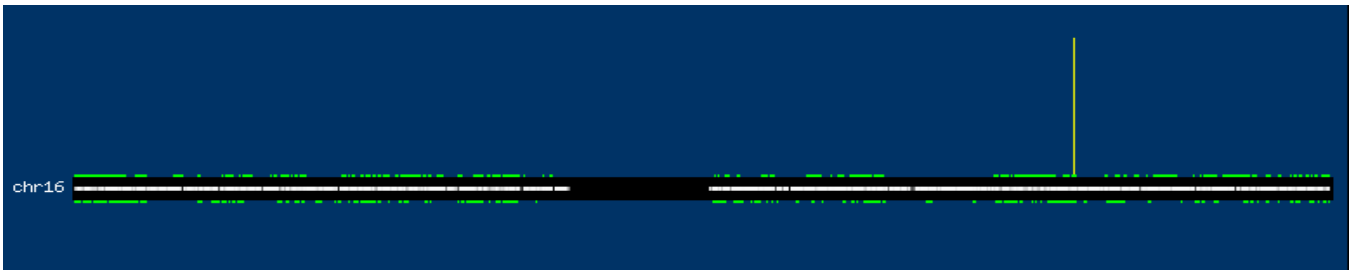
(42) ../bin/VAAST -x data/35bp_se.bed -t data/Miller_Syndrome_CE_union.cdr -pnt c -iht r -lh n -fast_gp -d 1e10 --significance 2.4e-6 -mp1 4 -o test -r 0.00035 -m lrt -k data/easy-hg18-chr16.gff3 data/196-new-hg18_chr16.cdr data/Miller_Syndrome_BD.cdr

Let's use look simple_output.pl to quickly eyeball the result:

(43) ../bin/vaast_tools/simple_output.pl test.vaast | more

```
RANK    Gene    p-value                 Score           Variants
1       DHODH   5.12741626621423e-05    37.15488694     chr16…
2       LDHD    1                       0
3       CPNE2   1                       0
…
```

Or try the VAAST viewer; DHODH is now the sole candidate-gene:



Note that now there is only a single candidate gene (DHODH), with a p-value of 5.1e-05. This is good, but still not genome-wide significant, at least not naively so. It turns out, however, that for quartet analyses, the genome-wide alpha for significance is considerable greater than the Bonferroni corrected one of 2.4e-06. The reasons are twofold. First, we can identify shared (IBD) regions in the two sibs prior to doing the VAAST analyses if we like. On average they will share 1/4 their genes in common, so the correction is 0.05/5250 or 9.5e-06, a little better than 2.4e-06. However VAAST can do better than this. As it turns out, when VAAST is run with the **–t, –ih r** and **–pnt c** options, the software pre-filters any gene whose genotype in an affected sibling is identical to either of the parents (such genes are denoted in the test.vaast file with the flag 'UPF:1'. See User's guide for details). VAAST only proceeds to score and permute those genes where every target has a genotype distinct from either of the parents and contains alleles that are inherited from both the parents (this last restriction helps to remove sequencing errors as well). For the case at hand, only a single gene on Chromosome 16 meets all of these criteria, DHODH (denoted by the 'UPF:0' flag in the test.vaast file) thus the Bonferroni corrected alpha in this case is 0.05/1, or 0.05, for our analysis of Chromosome 16. Genome-wide (data-not shown), there are around 1000 UPF:0 genes that meet these criteria for scoring and permutation, which would put the genome-wide alpha for statistical significance at around 0.05/1000, right at the observed VAAST p-value of 5.1e-05. Of course if you had more siblings to include, all of these analyses would have much greater power. As it is, two siblings and their parents is right at the threshold for statistical power.

Other approaches for analyses of pedigrees. For brevity's sake we have only covered the most basic of pedigree-based analyses with VAAST. Many others are possible. One particularly powerful one is to add the genomes of unaffected siblings and relatives to the background CDR file using the **cdr_manipulator.pl** script in the bin/vaast_tools directory. This helps to screen out false positives due to variants that are common to the family, but otherwise rare in the human populations. Careful though—this will only work under the assumption that penetrance is 100% and under a dominant inheritance scenario. This is only one example; many other variations of this sort are possible using the various tools in the VAAST package—*the point being that you can customize your VAAST analyses to leverage the unique features of your personal genomes/exomes datasets.* You just need to give it some thought.

Parting Advice. Hopefully this Quick–Start guide to VAAST has got you going. Although we have done our best to provide you with the basic knowledge needed to use VAAST successfully, VAAST has a many additional features and options beyond those we have mentioned here. For additional information we strongly suggest you have a close look at the VAAST user's guide in /data. It's also available at http://www.yandell-lab.org/software/vaast.html. There is also a VAAST list-serve group and bug tracker available; details are provided at the above URL. By signing up you will be able to get help and advice from the VAAST developers and from the rapidly growing user community— about 120 groups as of 11/2011. Finally don't forget to read the papers; links are available below:

VAAST User's Guide:

http://www.yandell-lab.org/software/VAAST_Users_Guide.pdf

Description of the VAAST algorithm and basic analyses approaches:

Mark Yandell, Chad Huff, Hao Hu, Marc Singleton, Barry Moore, Jinchuan Xing, Lynn Jorde and Martin Reese (2011). *A probabilistic disease-gene finder for personal genomes*, *Genome Research* doi:10.1101/gr.123158.111

This paper is available as a PDF here: http://www.yandell-lab.org/publications/index.html

A new rare Medelian disease discovered using VAAST:

Alan F. Rope, Kai Wang, Rune Evjenth, Jinchuan Xing, Jennifer J. Johnston, Jeffrey J. Swensen, W. Evan Johnson , Barry Moore , Chad D. Huff , Lynne M. Bird , John C. Carey , John M. Opitz, Cathy A. Stevens , Tao Jiang , Christa Schank , Heidi Deborah Fain , Reid Robison , Brian Dalley, Steven Chin , Sarah T. South , Theodore J. Pysher , Lynn B. Jorde , Hakon Hakonarson , Johan R. Lillehaug , Leslie G. Biesecker , Mark Yandell , Thomas Arnesen and Gholson J. Lyon (2011) *Using VAAST to Identify an X-Linked Disorder Resulting in Lethality in Male Infants Due to N-Terminal Acetyltransferase Deficiency* (2011) *AJHG,* 0.1016/j.ajhg.2011.05.017

This paper is available as a PDF here: http://www.yandell-lab.org/publications/index.html

A paper describing the GVF format and the 10Gen dataset:

Reese MG, Moore B, Batchelor C, Salas F, Cunningham F, Marth G, Stein L, Flicek P, Yandell M, and Eilbeck K. (2010) *A standard variation file format for human genome sequences. Genome Biology 2010, **11**:R88 doi:10.1186/gb-2010-11-8-r88Published: 26 August 2010*

This paper is available as a PDF here: http://www.yandell-lab.org/publications/index.html

A series of analyses of the 10Gen dataset; a good paper to get an idea of what lurks within a variant file:

Moore B, Hu H, Singleton M, De La Vega FM, Reese MG, Yandell M**.** *Global analysis of disease-related DNA sequence variation in 10 healthy individuals: Implications for whole genome-based clinical diagnostics* Genet Med. 2011 Mar;13(3):210-7.

This paper is available as a PDF here: http://www.yandell-lab.org/publications/index.html