VASP 1.0 Manual

Manual Table of Contents:

1) About VASP

VASP (Variant Analysis of Sequenced Pedigrees) is a flexible open-source software package capable of producing a summary of pedigree variation, providing relevant information such as compound heterozygosity, genome phasing, and disease inheritance patterns. Designed to aggregate data across a sequenced pedigree, VASP allows both powerful filtering and custom prioritization of both SNVs and small indels. VASP is designed to work on small to medium sized pedigrees with one or two generations although there is technically no limit on pedigree size or number of generations.

2) Installation

VASP is available from github at the following url https://github.com/mattmattmattmatt/VASP and when installed is roughly 400Mb including test data.

There are two ways to install:

1. If you have git installed, checkout the code:

```
$ git clone https://github.com/mattmattmattmatt/VASP
$ cd VASP
```

2. To download files as a zip go to https://github.com/mattmattmattmatt/VASP and select the 'Download ZIP' button. Open the zip file and type

```
$ cd VASP-master
```

To check whether the installation is successful for default case run:

```
$ test_vasp.pl
```

To test local install of vep

```
$ test_vasp.pl -vep_bin /path/to/local/vep
```

To test using bam files (preferred)

```
$ test_vasp.pl -samtools /path/to/local/samtools -ref_fasta /path/to/local/fasta
```

If successful will notify the user of the successful installation

To see basic help information type:

```
$ vasp.pl –h
```

To see more details type:

```
$ vasp.pl -man
```

3) Dependencies

1. **External software dependencies**: VASP is a perl package and has been tested on MAC, CentOS, Redhat, and Ubuntu platforms using perl versions ranging from 5.10 to 5.14.

   In the simplest use-case VASP has no dependencies providing the user has a ped family file, a vcf variant file, and an annotation file from ENSEMBL's variant effect predictor (VEP).

   However, there are two use-cases with external dependencies, first when using BAM files for obtaining pileup information and secondly if the user chooses to install VEP locally.

   Case #1: BAM files: Input pedigree BAM file list with '-bam_list file' option (preferred method). Passing in a list of BAM files for each pedigree member format=SampleName Full_BAM_path) requires the program samtools to be installed and also the fasta file for the reference genome used in the alignments to be passed to VASP.
   a. Samtools is available from http://samtools.sourceforge.net/ or http://www.htslib.org/ and once installed needs to be passed in using the parameter '-samtools /path/to/samtools'.
   b. The reference fasta file used for the alignment needs to be passed in using the parameter '-ref_fasta /path/to/fasta'.

   NOTE: Using BAM files to generate allele information takes considerably longer than using a vcf file with GT tags (and ideally AD tags). For large vcf files it can add ~10 minutes per bam file although this operation only needs to run once so if running VASP again with different parameters the existing pileup files are used.

   Case #2: VEP local: Running locally installed VEP instance. Note that there are three ways of dealing with VEP within VASP

a. Input pre-generated VEP file (see section 4.3 for details on VEP parameters to utilize)
b. Create vep file from vcf using web interface. For GRCh38 go to http://www.ensembl.org/Homo_sapiens/Tools/VEP and GRCh37 http://grch37.ensembl.org/Homo_sapiens/Tools/VEP
    1) Click 'browse' in the upload file box and select vcf file
    2) Open 'Extra options' box and check 'identify canonical transcripts'
    3) Click 'run'
    4) In the download box click 'VEP'
    5) Save this file and input it to VASP
c. Locally install VEP beforehand and pass the binary path into VASP. For download and installation instructions please go to http://www.ensembl.org/info/docs/tools/vep/script/vep_download.html Again refer to section 4.3 for details on VEP parameters to utilize

2. **Genome Build**: Please note VASP is able to work with data from any genome build (currently tested for GRCh37 and GRCh38) although users need to select the correct build to match their vcf file when running VEP.

4) Input requirements

VASP requires two input files and either a VEP annotation file or the path to a locally installed instance of VEP.

1. **-ped ped_file** -> REQUIRED:
   ped file describing pedigree structure (see http://www.sph.umich.edu/csg/abecasis/Pedstats/tour/input.html)

2. **-vcf vcf_file** -> REQUIRED:
   multisample vcf variant file
   If not inputting bam files (which is the preferred method) your vcf must include the genotype information (GT tag) for each sample with the sample ordering the same as the ped file. If using the VASP '-min_read_depth' the vcf file must contain the allele depth flag (AD tag)

3. **-vep vep_file** -> REQUIRED (or -vep_bin):
   vep annotation file (see http://www.ensembl.org/info/docs/tools/vep/index.html)
   a. ideally vep will be run with the following parameters: '--canonical --sift b --polyphen b --symbol --gmaf' although only '--canonical' is actually a requirement
   b. Mac users should add the additional flag of "--compress 'gunzip –c'" due to issues with zcat for Mac
   c. '--gmaf' flag is required for use with the VASP '-min_allele_freq' command line option
   d. '--poly b' flag is required for use with the VASP '-polyphen' command line option
   e. '--sift b' flag is required for use with the VASP '-sift' command line option

4. **-vep_bin vep_binary** -> REQUIRED (or -vep):

Full path to the variant effector predictor binary

By default vep will run with '--canonical --sift b --polyphen b --symbol --gmaf --offline --cache --i input_vcf '
Any extra vep parameters can be added using the vasp '-vep_extra' parameter. For example to use 4 threads use "-vep_extra '--fork 4'"

5. **-vep_extra extra_vep_arguments:**
   Extra arguments to pass to VEP when running VEP locally within VASP. Make sure to place quotes around the arguments
   e.g. -vep_extra '--fork 4 --dir path_to_local_cache'

6. **-bam_list text_file:**
   Text file containing sample name and bam file locations.
   i.e Sample1_name full_file_path1
   Sample2_name full_file_path2

7. **-ref_fasta fasta_file:**
   Single fasta file for the reference genome used for alignment

8. **-samtools samtools_bin:**
   Path to samtools binary (default=/usr/bin/samtools)

9. **-vcf_cutoff integer** (default=20):
   Variant quality cutoff to employ. This filter is ignored if quality is '.'

10. **-min_num_aff integer** (default=0)
    The minimum number of affected individuals containing the variant to include in report

11. **-polyphen polyphen_category** (default=OFF):
    Filter on polyphen category (i.e. probably_damaging, possibly_damaging, or benign). Will filter out any variants without polyphen scores

12. **-sift sift_category** (default=OFF):
    Filter on polyphen category (i.e. deleterious or tolerated). Will filter variants without SIFT scores

13. **-comhet** (default=OFF):
    Filter for variants found in genes determined to be either possible or definite compound hets (see section 6.1 for full definitions)

14. **-max_allele_freq float[0-1]** (default=1):
    Filter for variants below the gmaf frequency. Variants where the reference allele is the minor allele have their frequencies corrected to reflect this

15. **-phase_var_num integer** (default=2):
    Minimum number of consecutive variants to constitute genome block

16. **-min_phase_block integer** (default=10000bp):

Minimum size in bp to constitute genome block

17. **-denovo** (default=OFF):
    Filter for candidate denovo mutations (defined as at least one case of ref parents and het child)

18. **-inheritance inheritance_pattern** (default=OFF):
    Filter on inheritance type (options=ad,ar,xd,xr) for autosomal-dominant, autosomal-recessive, x-linked-dominant, and x-linked-recessive

19. **-coord string** (default=OFF):
    Filter by genomic range (format=chr:start-end)

20. **-gene_list text_file** (default=OFF):
    File containing list of genes, one on each line. Genes can be ensembl transcripts, ensembl genes, or hgnc names as long as vep was run with '--symbol'

21. **-chrom chrom** (default=OFF):
    Filter by chromosome number (e.g. 1,2,..,X,Y,M)

22. **-out filename** (default=vasp.tsv):
    Name of output tsv file

5) Running VASP

1. **Parameters:**
   Cutoffs: Several important cutoffs are utilized:
   - **a.** Variant classification: For a coordinate to be called variant by VASP at a specific coordinate 10%* or more of the total bases must be different from the reference base. Non-reference bases include a single base other than the reference, an insertion of any size, or a deletion of any size. Note that this definition of variant likely differs from the definition applied by a variant caller like GATK or samtools.

   - **b.** Zygosity classification: There are three categories for zygosity
     - **i.** ref: >90%* of the bases match the reference
     - **ii.** hom: >90%* of the bases are the same variant base
     - **iii.** het: everything not in the above categories is called het (i.e. variant base frequency is between 11%-89%)

   *Note the values of 10% and 90% are empirically derived from validation experiments within our production system however these values may not be ideal for low coverage variants. To avoid issues with low coverage variants use the '-min_read_depth' filter of possible.

2. **Filtering strategies:** Output files are typically large with lots of information but there are ways to prioritise for the variants most likely to be causal. Please see the column definitions in the next section for descriptions of the columns mentioned below. Based on previously analysed cohorts where we've found the causal variant(s), they usually fall into the following categories.

Cases likely to contain variants of interest:

- **a.** Variants flagged as de novo in mendelian_inheritance column → beware cases with low coverage as this often results in unreliable zygosity calls, use -min_read_depth to filter out these cases
- **b.** Variants common to affected individuals and absent in unaffected individuals; use columns number_affected_variant and number_unaffected_variant for this
- **c.** Variants flagged as auto-dominant, auto-recessive, X-linked-dominant, or X-linked-recessive in the disease_inheritance column
- **d.** Variants where the mutant allele for all affected samples comes from a single parent while the unaffected samples get the reference allele. Look for Father gives affected allele or Mother gives affected allele in the parent_allele_common_to_affected
- **e.** Compound hets with low gmaf frequencies found in the definite_compound_het and possible_compound_het column
- **f.** Variants either not in dbsnp (NOVEL) or rare alleles (RARE) in the category column. There is also an inconclusive NO_FREQ category with represent known variants with no reported frequency.
- **g.** Nonsense mutations; use the aa_changes column and look for /*
- **h.** Missense mutations determined to be damaging by polyphen and/or SIFT. Both pieces of software are know to have false positive rates of ~20% but perform better when both programs determine a variant is damaging; use the column polyphen (look for probably_damaging and high scores close to 1 and sift (look for deleterious and score close to 0)

Cases NOT likely to contain variants of interest:

- **a.** Low coverage and/or missing allele variants. Use the sample_zyg columns to check for either low coverage levels or no data entries. These low coverage levels often result in incorrect zygosity calls. To avoid including low coverage variants use '-min_read_depth' which removes all variants where at least one member have less than this coverage level.
- **b.** Variants in genes with a large number of mutations; use the gene_total_variants_in_gene column to check for high number
- **c.** Variants where the reference allele is the rare allele in the population. Use the allele_freq column to check for high values (0.5-1)

3. **Filtering examples:**
   - **a.** To run with default parameters:

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep
```

   - **b.** Filter by chromosome:

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -chrom 22
```

   - **c.** Filter by genomic region:

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -coord 22:10000000-20000000
```

   - **d.** Filter by vcf cutoff:

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -vcf_cutoff 40
```

e. Filter for compound heterozygous genes:

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -comhet
```

f. Filter for polyphen 'probably_damaging'

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -polyphen probably_damaging
```

g. Filter for SIFT 'deleterious'

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -sift deleterious
```

h. Filter for max_allele_frequencies <=10%

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -max_allele_freq 0.1
```

i. Filter for genes in gene list file

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -gene_list sample/gene_list
```

j. Filter for inheritance type autosomal-dominant

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep  \
sample/sample.vep -inheritance ad
```

k. Change definition of genome block to need 5 variants and be 50000bp

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -phase_var_num 5 -min_phase_block 50000
```

l. Filter for variants where every pedigree member has at least 10 reads

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -min_read_depth 10
```

m. To use bam file for generating zygosity, inheritance, parental alleles, etc (preferred method but for large vcf file can take a long time....)

```
$ vasp.pl -ped sample/sample.ped -vcf sample/sample.vcf -vep \
sample/sample.vep -bam_list sample/bam_list -ref /path/to/fasta \
-samtools /path/to/samtools
```

6) Output format

Default output file is named “vasp.tsv” but can be overridden with '-out newfile' flag. The file contains a summary of all snvs or indels called in at least one member of the cohort. Each line in this file represents everything we know about a particular variant for the entire cohort. Output files are designed for importing into excel or libreoffice using tab as a delimiter. Please note that columns reported depend on the constitution of the pedigree (e.g pedigrees without parents will not have column regarding phasing).

All possible columns are listed below with (NOT_REQ) next to the columns that are not always reported for all pedigrees.

Output Columns

A. **category**: Four possible variant categories: novel, rare (0-2% MAF), no frequency (known variant but no MAF available), and common (>2% MAF).

B. **variant_samples**: List of all samples containing the variant according to the pileup files generated or GT tags

C. **number_affected_variant/total_affected** (NOT_REQ): Number of affected samples defined as variant (see definition above) / Total number of affected samples in the cohort

   e.g. 2 out of 2 → both affected samples in the cohort contain this variant
   e.g. 1 out of 3 → 1 out of a total 3 affected samples contain this variant

D. **number_unaffected_variant/total_unaffected** (NOT_REQ): Number of unaffected samples defined as variant (see definition above) / Total number of unaffected samples in the cohort

   e.g. 1 out of 3 → 1 out of a 3 unaffected samples contain this variant
   e.g. 0 out of 1 → the single unaffected sample in the cohort is not variant at this coordinate

E. **mendelian_inheritance** (NOT_REQ): Columns indicates whether Mendelian inheritance rules are observed within the cohort. If Mendelian inheritance rules are not followed one of the exception(s) will be reported. For example hom_parent ref_child would indicate there is a parent who is homozygous at a particular allele while their child is reference which shouldnt happen with normal Mendelian inheritance. Cases following a de novo pattern are also flagged (defined as both parentsbeing reference and the child being heterozygous) and represent potential variants of interest.

   e.g. mendelian_rules_followed → allele distribution within the cohort follow Mendelian rules
   e.g. de novo (ref_parents het_child) → de novo case where affected child has de novo mutant
   e.g. hom_mother ref_child → broken Mendelian rule; here its a hom_mother and a ref_child
   e.g. missing allele_info → some samples contain no sequence information

F. **disease_inheritance**: Disease inheritance pattern observed for each variant. There are five distinct possible values for this field any of which may also be additionally annotated as def_com_het or pos_com_het (see column definitions below). The five options are auto-recessive, auto-dominant, x-linked-recessive, x-linked-dominant, or none. Auto-recessive is defined as all affected being homs and all unaffected being non homs while auto-dominant is defined as all affected being heterozygous and all unaffected reference. The x-linked categories are the same except the variant falls on the X chromosome.

   e.g. auto-dominant
   e.g x-linked-recessive

e.g. auto-recessive,pos_com_het
e.g. x-linked-dominant,def_com_het

G. **gene**: The mutated gene name from hgnc (if available in vep file), otherwise ensemble gene name

H. **total_gene_variants**: Total number of mutations in the gene for the entire cohort. For example if two individuals share three different mutations in the same gene the value will be 6.

I. **unique_coord_gene_variants**: Total number of distinct coordinates mutated within the cohort for a single gene. For example if two individuals share three different mutations in the same gene the value will be 3.

J. **Sample_zyg** (with no BAM files) or **Sample_pileup(zyg)** (with BAM files): These columns report the zygosity call (see definitions above) for each sample and optional summarized pileup string if bam files used or allele depth string if vcf file with AD tags is input. There will be one column for each sample. The format of these columns is a string that shows the sequence content and their respective counts at that coordinate for the sample followed by a zygosity call.

   e.g with BAM or vcf AD tags: ref:10_3:C_4:+2AG (het) → shows there are 10 reference base, 3 Cs, and 4 AG insertions at that coordinate and that the zygosity is heterozygous
   e.g. with no BAM or AD tags: het

K. **mother_allele** (NOT_REQ): Contains the allele inherited for each child from the mother when this can be determined.

   e.g. affected1(A),affected2(ref),unaffected1(?) → affected1 get a mutant A allele from the mother, affected2 gets the ref allele from the mother, and we cant determine what allele unaffected1 gets from the mother
   e.g. affected1(+1A),affected2(+1A),unaffected1(-2AA) → affected1 and affected2 get an insertion of an A from the father, and unaffected1 gets deletion of AA from the mother

L. **father_allele** (NOT_REQ): Same as above column but for the father

M. **parent_allele_common_to_affected** (NOT_REQ): Flags cases where the following conditions are met:
   -All affected children inherit the same mutant allele from the same parent
   -All unaffected inherit the reference allele from this parent
   -All samples inherit only the reference allele from the other parent
   These cases often cluster and are likely to be regions of interest for causal variants. Possible values are no, ?,Mother gives affected allele, and Father gives affected allele

   e.g. If we have mother_allele = affected1(A),affected2(A),unaffected1(ref) and father_allele =affected1(ref),affected2(ref),unaffected1(ref) then the value will be Mother gives affected allele

N. **affected_allele_block** (NOT_REQ): Contains information on genomic blocks of cases described in column above i.e. cases where variant allele goes to affected children only from the same parent. These blocks may contain inconclusive variants but not cases where we can definitely state conditions described in above column are not met.

   e.g. 4 variants; Mother; 22:21988602-22049369; 60767bp → means 4 consecutive variants in the ~60kb region from 22:21988602-22049369 are from the Mother

O. **definite_compound_het** (NOT_REQ): To a gene to be defined as definitely compound het the following conditions must be met:
   I) Gene contains at least one heterozygous variant inherited from each parent
   II) The variant must not be homozygous in any unaffected individuals
   III) The variant must be heterozygous in all affected individuals
   IV) Unaffected and affected siblings must not share the exact same heterozygous variants
   The variants where the mother gives the mutant allele begin with M= while F= represents cases where the father gives the mutant allele. This calculation is independent of variant type (i.e. a child can inherit an indel allele from their mother and a snv allele from their father). The strings contain lots of information and can get very long for genes with lots of mutations.

   e.g.affected1(M=13:40261945:snv:gmaf=0.2766(G),F=13:40229957:snv:gmaf=0.4638(A))→ this means child affected1 gets a mutant allele from their mother (M=) for a snv at chr13:40261945 with gmaf of 0.2766 AND gets the mutant allele from their father (F=) for another snv at chr13:40229957 with gmaf of 0.4638.

P. **possible_compound_het** (NOT_REQ): Cases similar to the above except that condition I) cannot be guaranteed due to parent and child both being hets meaning we cannot definitively know which allele the child got from each parent. These variants are flagged as AMB in the string below.

   e.g.affected2(F=13:25876011:snv:gmaf=0.4075(A),AMB=13:25882049:DEL:gmaf=N/A) → this means child affected gets a mutant allele from their mother (F=) for a snv at chr13:25876011 with gmaf of 0.4075 AND possibly gets the mutant allele from their father (AMB=) for a deletion at chr13:25882049 with no known gmaf score

Q. **chr**: chromosome

R. **start_coord**: start coordinate

S. **end_coord**: end coordinate

T. **var_type**: SNV, DEL, or INS

U. **variant**: The base change of the variant
   For SNV: ref_base->var_base (e.g. A->T)

For INS: +BASES (e.g. +AT)
For DEL: -BASES (e.g. -TCG)

V. **var_quality**: Variant quality score from vcf

W. **allele_freq**: Variant allele frequency (NOT necessarily MAF). For cases where the reference base is the minor allele the variant allele frequency is approximated as 1-GMAF. This avoids prioritising variants where the reference variant is rare in the larger population.

X. **dbsnp**: rs id from dbsnp

Y. **ens_transcript**: ENSEMBL transcript

Z. **vep_classifier**: Classifier from vep

AA. **aa_change**: Amino acid change for non-synonomous mutations. Nonsense mutations have stop codons annotated as '*'

BB. **polyphen**: Polyphen classfication and score (range 0-1, more damaging closer to 1)

CC. **sift**: SIFT classification and score (range 0-1, lower is more damaging)

Default sorting
Variants are segregated into four categories: novel, rare (0-2% freq), no frequency (known variant but no freq available), and common (>2% freq). By default variants are internally sorted on four measures: variant category (novel, rare, no frequency, then common), descending order on the number of variant affected samples, ascending order on the number of unaffected variant samples, and lastly on the variant population frequency

7) Contact
Contact: matt.field@anu.edu.au

8) Citation
Citation: Please cite Field et al. Bioinformatics….. when publishing results VASP

9) Acknowledgements

We thank the National Computational Infrastructure (Australia) for access to significant computation resources and technical expertise. This work supported by NHMRC Australia Fellowship 585490, National Institutes of Health [grant number U19 AI100627], and Bioplatforms Australia.