User Guide

Introduction

VarSifter is a program designed to view massively parallel sequencing variation output. It allows sorting on any field (as well as combinations of fields), and filtering on different types of information (variant type, inheritance, etc). Additionally, it allows custom filtering. The program is written in Java, and should run on any platform with a current Java Virtual Machine.

Downloads

VarSifter is available by anonymous FTP at
ftp://ftp.nhgri.nih.gov/pub/software/VarSifter/".

Requirements

Basic VarSifter functionality requires Java JRE 1.5 or newer, but Java JDK 1.6 is recommended, as the custom query/filter functionality requires Java JDK 1.6 or better (JDK is the full Java Development Kit). Although any operating system with a version of Java should work, it is known to work on Windows XP and XP 64-bit, Mac OS X, and the CentOS and Gentoo distributions of GNU/Linux. VarSifter has been successfully tested on Oracle Java and the IcedTea build of the open source OpenJDK.

VarSifter runs best with a current 64-bit processor, and a 64-bit Java version. The amount of RAM required depends on the number of samples you wish to view. A machine with 1 GB RAM will allow all variants in ~30 exomes; 8GB RAM will allow all variants in ~180 exomes.

Additionally, the Java Universal Network/Graph Framework 2 (JUNG2) is also required (VarSifter has been tested with version 2.0.1.) This is included in the "package" download, but can also be downloaded from http://jung.sourceforge.net/.

Installation and Start

To install VarSifter from the package file, download the '.zip' file into the desired directory on your computer. Unzip the file (usually by double-clicking it.)
Mac OS X users should then double-click the '.command' file.
Windows users should then double-click the '.bat' file.

This will open a terminal window, and execute the Java commands to run VarSifter. Leave the terminal window open until you are finished with VarSifter, as important messages and any detailed error messages will appear here.

Once VarSifter is started, you should choose a file. To do this, click "File" on the menu bar, and then "Open File". Then, choose the .vs or .vcf file with the variant data (descriptions below). Click this file, click 'Open', and you're ready to start sifting! (Data files can be specified in the helper files - see "Manual Start".)

Manual Start

To run VarSifter manually, open your Terminal (OS X) or Command Prompt (Windows), and navigate to VarSifter's location using the command 'cd'. Then, type the following:
java -Xmx800M -jar VarSifter<version>.jar <data file>

Xmx800M requests a maximum of 800Mb of RAM, which should be sufficient for exome analysis of 3-4 samples. If you notice problems, or Java errors involving "heap space", increase this number (though it should be lower than your total system RAM).

You can create a helper file, which can be double-clicked to start VarSifter from the command-line:

Mac OS X users can create a file called vs.command with the following:

```
#!/bin/bash cd `dirname $0`
java -Xmx800M -jar VarSifter<version>.jar
```

Windows users can create a file called vs.bat with the following:

```
java -Xmx800M -jar VarSifter<version>.jar
```

The path to a data file can be added at the end of the line starting with 'java', just as can be done when typing the command manually. Additionally, 'java' can be changed if you wish to define a specific full-path to the Java executable.

<version> should be replaced with the VarSifter version you are using, the memory should be adjusted to the appropriate amount, and the file should be saved in the same directory as the VarSifter jar. You can now double-click this file to run VarSifter easily using a desired amount of memory.

Usage

VarSifter is designed to be easy to use. In the upper main window, you will see all variant positions, one position per row. This window displays one row per position (and per gene, per variant). Clicking any one row will display the genotypes, genotype score, and depth of coverage for each sample in the lower main window.

The right side of the screen contains the filtering options, as well as the action buttons. Generally, you select the filters you are interested in, and then click 'Apply Filter'. The main upper window will now display only those variants that meet your filter criteria. Note that the row count is displayed on the bottom of the screen next to "Number of Variant Positions". Clicking 'Clear All' will unselect all filters.

There are several types of filters. The first box of "Include" filters (Stop, DIV, etc.) will show any positions that meet any of the checked criteria. These criteria mostly relate to the type of variant. (If no boxes are checked, none of these filters are applied.) Filters in the "Exclude" box, when checked, will remove those variants from the display. The next "Include" box applies more to samples, and includes inheritance filters, Tumor/Normal pair filters, case/control filters, etc. Some of these boxes may be grayed out, as the data file does not contain the filter information (inheritance analysis can only be performed with family data, for example.) Selecting files with the buttons on the bottom right will activate some of these disabled buttons.

The box titled "Search gene names for:" is special. It allows you to use Java regular expression syntax to make your gene name search much more powerful. Regular expression syntax is beyond the scope of this guide, but worth learning! As of this writing, Oracle provides a guide at the following Web site: http://java.sun.com/docs/books/tutorial/essential/regex/index.html.

Columns in the main annotation window and lower sample window are sortable. Click once on a column header to sort, once to sort in reverse order, and once more to return to the unsorted state. One can sort hierarchically by CTRL-clicking additional column headers.

The annotation window has two views, "Show Variants", which shows the variants by position, and "Show Genes", which shows genes, and the number of variants in each gene that pass your filters. Clicking on the 'View Variants for Selected Gene' button will bring up another VarSifter window showing all the variants for the selected gene (or the gene the selected variant belongs to).

There are two file filters available: 'Choose Gene File Filter' and 'Choose Bed File Filter'. Clicking one of these buttons and choosing an appropriate file will enable the checkboxes, allowing one to filter using these lists. A bedfile is a list of regions in the standard BED format (http://genome.ucsc.edu/FAQ/FAQformat.html#format1), and is useful if you have a list of interesting regions (linkage regions, GWAS peaks, etc). A gene file is a list of gene names, one per line. Note that the gene names must match exactly to those in the VarSifter file. (Case is not important.)

NOTE: After loading a BED or gene file, you still must apply the filter by checking the appropriate "Include" or "Exclude" box!

Useful Regular Expressions for Gene Name Searches

Note that these searches are case-insensitive.

Regular Expression Search Term	Search Term Meaning
^Cftr$	Gene name matches "Cftr" exactly; nothing before or after "Cftr". ('^' is the beginning of the line, '$' is the end)
Cftr|Apob	Gene name can be either Cftr or Apob (and things like vCftr or Apob2 if such genes exist). The '|' (pipe) character is used to separate alternate terms.
Apob[12]	Apob1 or Apob2 only, not Apob3

Sample Settings

The last two filters in the lower "Include" box on the right are the Affected/Normal pair filter and the Case/Control filter. To use these filters, you need to assign sample status, if this was not already defined in your data file. To define the status of each sample, click the "File" menu, and then "Sample Settings". You will see a box with one row per sample, and a number of checkboxes. Check the boxes that apply for each sample. Note that a sample cannot be Affected and Normal, nor can it be Case and Control. Also, every affected sample must have a normal pair, and these are assigned in order (after finishing, reopen the Sample Settings window, and check the "Aff/Norm Pair Group" column). There must be at least one Case and one Control to activate the Case/Control filter. After you are finished defining the samples, click "Ok". The appropriate filters will now be available. To filter using Affected/Normal pairs (such as for a Tumor/Normal pair), check the box, and select the minimum number of pairs in which a difference must be observed. For Case/Control, check the box, and define the minimum number of cases with a variant genotype, and then the maximum number of controls with the variant genotype.

Preferences

Preferences are available by clicking the "File" menu, and then "Preferences". Currently, you can set genotype score and (genotype score / coverage) cutoffs here in the top part of the window. Note that these are minimum scores, and must be seen in the indicated number of samples to include a variant. The two filters in the bottom part of the window control the minimum score for the Aff/Norm, Case/Control filters, and the low-lighting cutoff. To apply any of these settings, you must click "Apply Preferences", and then refilter with "Apply Filter"!! Currently, if the coverage is listed as 0, the genotype is not counted, even with a high score (since we cannot know what the coverage is.)

Custom Query Usage

If you have correctly unzipped the JUNG files into the same directory as the Varsifter.jar (or have unzipped the 'package' installation), and are using Java JDK 1.6 or higher, you can create custom queries/filters. Click 'View' in the menu, and then 'Custom Query'. There are two types of filters you can create. Sample filters (left side) allow you to compare sample genotypes. For example, you can click on one sample in the "Samples:" list, click 'Exactly Matches', and then click on another sample to create a filter that requires the two samples to have the same genotype. Annotation filters (right side) allow you to create filters based on the included annotation columns. Start by selecting and annotation column from "Annotations:". Then, if the data is textual, click either 'Exactly Matches' or 'Does Not Match' in the "Annotation Actions:" box. You can then either select a value present in the file in the "Annotation Values:" box, or enter a search term (regular expressions supported) in the "Annotation Actions:" box and click 'Apply Search Text'. If the column you originally selected contained numeric information, you can specify numeric relations in the "Annot. Numeric Actions" box. Click a comparison term, enter a number in the box, and click "Apply Number".

You now have a box in the middle of the screen. This is a query. You can use it now by clicking 'Finalize Query' and then clicking 'Apply Filter' back on the main window.

For more complicated queries, you can add additional query boxes. Query boxes must be linked with a logical, i.e., if both queries must be passed in order for a variant to pass, SHIFT-click each test (they will turn yellow), and then click 'AND'. The logic of the query is displayed for you. Selecting one or more boxes and clicking 'Delete Selected' deletes those boxes. When ready to run the query, click 'Finalize Query', ensure the "Custom Query" checkbox is checked, and then click 'Apply Filter' in the main window. Note that custom queries behave like any other, and can be combined with other filters.

When the pointer is over the graph window, you can use the middle mouse wheel to zoom in and out (the zooming is very fast, so scroll slowly!). Also note the pull-down box with the choices "Picking" or "Transforming". "Picking" allows choosing of boxes to link with logical statements. "Transforming" allows rearrangement of the boxes, although the rearrangement is reset on subsequent actions.

Queries can be saved by clicking 'Save Query'. This creates a binary Java object file. The query can be loaded by clicking 'Load Query', and choosing the appropriate file. Note that a saved query will only work on the EXACT data file it was created from! VarSifter checks this by comparing the data file names. Therefore, you should always use different names when creating/saving VarSifter data files.

VCF Files

VarSifter can read VCF files as defined by the VCF Standard 4.0. (http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-40). VarSifter shows one variant per line, and therefore splits VCF lines with multiple alternate alleles. To ensure the proper fields are also split, VarSifter asks the user to input information about the file before opening.

If an INFO field can contain multiple values, one for each alternate allele, this is considered "Multi-Allelic". An allele frequency field is an example of this, as you may expect to have a separate allele frequency for each alternate allele. For each of these "Multi-Allelic" fields, users should click the checkbox in the "MultiAllele" column of the popup window.

Additionally, if users are using an additional delimiter to further split individual info fields, VarSifter needs to know this. Enter the character in the "Sub-delimiter" column of the popup box and hit 'Return/Enter' or click a different box (to get Java to accept your entry). At the moment, VarSifter does not split these fields, but the presence of a sub-delimiting character helps to properly determine the type of data in the field.

When you have finished entering this information, click 'OK'. You will be asked if you want to save this configuration. If you choose "Yes", a config file will be written. The next time you open this data file, VarSifter will identify the saved config file, and ask if you wish to use it, avoiding the need to enter all of this information again. Config files are saved by appending ".vcf_config" to the original VCF file name.

Unfortunately, the version 4.0 VCF specification does not define specific tags for annotations, such as Gene Name, or variant type. If this information is included in a VCF file, custom queries can be used to filter those annotations.

VarSifter saves VCF files as VS files, described below.

VS File Format

VarSifter is also designed to read in "vs" files, a text file with some specific format rules used at NISC. The file is an uncompressed tab-delimited plain-text file with a single header line. Different types of data are stored in columns, and different variant positions are stored in rows. The first row is always a header, which labels the columns. In order for VarSifter to work most efficiently, data in a column should all be the same type, i.e., a column with numbers should only have numbers.

The order and requirement of column types is designed to be flexible, but some rules must be followed, and some fields are required for VarSifter to work correctly. Sample fields must come last, and each sample must have three fields in this order: genotype, genotype score, and coverage.

The headers for each column of sample data must currently end in ".NA" or ".NA.<word>" where <word> is letters or numbers. No other headers can have this combination!!

The order of samples in the file dictates order in VarSifter.

To make use of Affected/Norm filters and Case/Control filters, the three column headers for each sample, in addition to "NA", must have one of the following identifiers:

- **aff**: Affected member of a pair. This sample must be next to the corresponding normal sample.

- **norm**: Normal member of a pair. This sample must be next to its corresponding affected sample.
- **case**: This sample is a case.
- **control**: This sample is a control.

The Affected/Normal filter (designed for tumor/normal comparisons) requires aff/norm pairs, where the pairs are next to each other. The Case/Control filter is more flexible: any number of either tag is allowed, and not all samples need to be tagged.

Annotation fields precede the sample fields, and must include the following in any order: "Chr", "LeftFlank", "RightFlank", "Gene_name", "type", "muttype", "ref_allele", "var_allele".

Other custom annotation columns are allowed (must precede sample columns), and are sortable in VarSifter. Note that the column names must be unique!

If you wish to use the Mendelian filters, you must calculate the status of each variant row ahead of time, and store the information in these columns, as defined below: MendHomRec, MendDom, MendHetRec (and Index), and/or MendInconsis.
The following is a list of commonly used, and special column headers:

Required:

- **Chr**: chromosome
- **LeftFlank**: The position to the left of the variant. For indels, the position to the left of the ambiguous bases.
- **RightFlank**: The position to the right of the variant (and indel, as above).
- **Gene_name**: The name of the gene. We use the refseq gene symbol.
- **type**: Variant type. VarSifter uses the entries in the file to create filter buttons.
- **muttype**: SNP for single nucleotide variant or INDEL for insertion/deletion variant.
- **ref_allele**: Reference base at this position (plus strand).
- **var_allele**: variant base at this position (plus strand).

Reserved (VarSifter uses these for the indicated purpose):

- **dbID**: dbSNP ID (or other ID), or - if none.
- **Comments**: An editable field to write comments. Must save the file using File menu to save comments.
- **Index**: An numeric index numbering lines is used to allow compound het pairs to remain linked, even when a subview is saved. It is required if using MendHetRec.
- **MendHomRec**: Mendelian Homozygous Recessive. 1 if yes, 0 if no.
- **MendDom**: Mendelian Dominant. 1 if yes, 0 if no.
- **MendHetRec**: Mendelian Heterozygous Recessive (compound het). Comma separated list of pair-indices (from Index field) if yes, '0,' if no. Requires Index field to work properly.
- **MendInconsis**: Mendelian Inconsistent. Can be either artifacts, or de novo mutations. 1 if yes, 0 if no.

Tutorial

As part of the "Exome 101" course presented by researchers at NHGRI in the fall of 2011, the author of VarSifter (Dr. Jamie K. Teer) presented a session titled "Variant Annotation and Viewing Exome Sequencing Data". The full lecture series is available on

YouTube through the GenomeTV channel:
http://www.youtube.com/playlist?list=PL6E6AA89291FBA884
Dr. Teer's session
(http://www.youtube.com/watch?v=I7azpqTWFuM&list=PL6E6AA89291FBA884&index=3&feature=plpp_video) presents a brief demo of VarSifter starting at 26:50.

Troubleshooting

1. VarSifter will not start. I see error messages about "class" or "loading".
   - If you've copied and pasted the start command from the internet or E-mail, try typing it by hand. Sometimes, the minus sign is interpreted as a weird character that command lines do not recognize.
   - If you're using a Windows machine, Internet explorer may be "helpful" and rename the .jar file as .zip. This is bad. Either use Firefox to "Save As" the file (making sure it is not changing the filetype) or rename the file. To rename the file in Windows correctly, you either must be showing file extensions, or you have to use the "ren" command on the command prompt.
   - If you are using a helper file, make sure the VarSifter version in the helper file matches the version you want to be using (the latest version in your directory.)
2. VarSifter will not start. I see error messages about "heap space" or "memory".
   - VarSifter is not being started with enough memory. Increase the memory requested in the -Xmx command. If using a helper file, change this in the helper file. Note that this value must be less than the total system memory in your machine, and you may have to close other programs to be able to use this memory.
   - You may be using a 32-bit Java virtual machine. In 64-bit Mac OSX, open "Java Preferences" in Applications->Utilities. Make sure that the topmost java version is 64-bit. Note that any 32-bit operating system (includes Windows XP) will not be able to run a 64-bit Java, and will be limited in how much memory can be used!
3. VarSifter gives errors about Custom Querying, but I have the full Java 1.6 JDK installed.
   - Although the full Java Development Kit may be installed, your system may not be recognizing it. Ensure the 1.6 JDK is the system default. This is probably defined in your Java settings, which tend to be system specific. Alternately, find the correct java binary file, and point your scripts at the full path to that file. On OS X ~10.6, that may be in: /System/Library/Frameworks/JavaVM.framework/Versions/.
   - This can be especially problematic on Windows machines. Here is a workaround, courtesy of David Adams:
     1. Install java JDK
     2. Append JDK bin directory to system PATH environmental variable
     3. Go into JDK bin directory and make a copy of the java (java.exe if you have extensions visible) file and name the copy javasdk (javasdk.exe)
     4. Start VarSifter at command line with "java -Xmx1000M -jar VarSifter\ .jar"

- Alternatively, one can remove all Java versions, and then install only the Java JDK 1.6.