SpliceVista manual

Developed by Yafeng Zhu. Email: yafeng.zhu@scilifelab.se, yafeng.zhu@ki.se

Overview
SpliceVista is a tool for identification and visualization of splice variants based on shotgun proteomics data

SpliceVista was written in Python 2.7.2. It contains six python scripts: converter.py, mergepsm.py, download.py, clusterpeptide.py, mapping.py and visualization.py. The following python packages need to be installed for SpliceVista to work:
Biopython
Python Image Library (PIL)
numpy
scipy (only needed if you run step5-clusterpeptide.py)

Manual
Download SpliceVista using the following command: (you might need to install git first, type the command in a terminal: apt-get install git-core)

git clone https://github.com/yafeng/SpliceVista

Note: After you have download SpliceVista, DO NOT move or rename original and output files generated by SpliceVista.

Step1: prepare the input file for SpliceVista
First, you need to extract PSM data from the output file of database searching and format them in a tab-delimited text file like this:
peptide sequence protein accessions sample1 sample2 sample3

(put the standard or control in the first place, intensity will be normalized in next step)
The number of samples can vary (at least four samples if you do peptide clustering later), if label free method is used for quantification, precursor peak area can be used. In the end, they will be normalized as well.
Tip: arrange the needed columns in EXCEL, and then copy and save it in a tab-delimited text file.
It is OK to have multiple IDs in the Accession column, but it has to be separated by semi comma (;) symbol. Peptide sequence can contain letters in lower case.

Here, we will use the heavy_testfile.txt file (a tab-delimited text file as described in step1) to illustrate the workflow.

(If you have already grouped PSMs into peptides, just run step 2, rename output file from _psmdata.txt to _pepdata.txt and skip step 3)

Step2: insert gene symbol for each PSM- converter.py

Insert gene symbol for each PSM. Each protein accession will be assigned a gene symbol which will be used to retrieve its known splice variants in the next step. This is done by converter.py.

Command: Python converter.py --i heavy_testfile.txt --prefix heavy --database ensembl --n 2

The first argument --i is the input file, the second is the prefix of output file. Given the prefix in the example, you will get an output file named as heavy_psmdata.txt.
--database specify the database was used to search peptide spectra. Options are "ensemble", "uniprot", "IPI", "ECgene" and "ensemble+splice". Use "ensemble+splice" when reference database is the Ensembl72 human protein database concatenated by non-redundant predicted splice junction peptide database provided in the SpliceVista package. If "ECgene" is used, visualization of peptides and splice variant will not work due to ECgene database was built on assembly 18, but peptides can be still be mapped to genome (assembly 18).

--n has three options for normalization: if 0, no normalization is performed, if n is given 1 or 2, intensity will be normalized in method 1 or 2, see below for method:

use --n 0 when intensity of peptides are already normalized in input files.

Intensity will be normalized by the intensity of median protein assuming that is equal in all samples.
normalization method 1:
X(relative intensity)=intensity of X/ intensity of sample1,
usually used when sample1 is standard or control. X=1,2,3…
normalization method 2:
X(relative intensity)=intensity of X/mean of intensity of sample1 and sample2,
usually used when sample1 and sample2 are two replicates in control sample.
X=1,2,3…

Step3: group PSMs into peptides-mergepsm.py
Then for each peptide, calculate the mean of all PSMs' relative intensity and the standard deviation.

Command: Python mergepsm.py heavy
output: heavy_pepdata.txt

Use the same prefix in previous step to group PSMs into peptides, relative intensity of peptide is the mean of all the peptides PSM's relative intensity, standard deviation is calculated.
Before clusterpeptide.py is used, all peptides will be assigned to cluster 0.

Step4: Download data from EVDB and GenBank - download.py
The script in this step retrieves splice variants in EVDB by gene symbol and the translated sequences of these splice variants in GenBank.

Command: Python download.py --prefix heavy --organism 9606 --email your_email_address
Currently, nine species below are available to choose to search splice variant in EVDB.

organism="3702">A. thaliana

organism="9913">B. taurus

organism="6239">C. elegans

organism="7227">D. melanogaster

organism="7955">D. rerio

organism="9606">H. sapiens (default)

organism="10090">M. musculus

organism="39947">O. sativa

organism="10116">R. norvegicus

Use the same prefix in previous step and provide a valid email address to access NCBI GenBank.
Output: splicingvar.txt, subexon.txt, varseq.fa, gene_notfound.txt

Note: DO NOT download splice variants for different species under the same directory, or files will be overwritten. SpliceVista Package contains three empty files splicingvar.txt, subexon.txt, varseq.fa that will store downloaded splice variant information. The output files splicingvar.txt and subexon.txt contain exon composition of each variant, both genomic and transcript coordinates. The file varseq.fa will store the amino acid sequences of all splice variants. The gene_notfound.txt file contains gene symbols which are not found in the EVDB database. (You can download human splice variant data files directly from the following link: https://www.dropbox.com/sh/q8vo542udkdbdwl/7SYS9HF3O8. After that, copy and replace the three files in SpliceVista directory, then run step4 command again, now only splice variants that are not in the files will be downloaded)

Tip: if you have multiple sample files in one project, it is better to put all the files under one directory named by the project and use different sample names for each file. Because usually there is an overlap of identified proteins among different samples, the script download.py first check if the splice variants of identified protein and the variant sequences have been downloaded so that it avoids downloading the data for same proteins multiple times.
This step will take time depending the number of new splice variant to be downloaded from the database (on average, it takes 1 second to retrieve information from EVDB and NCBI for one splice variant). In order not to cause Internet traffic, it is highly recommended to download these files first from the link provided above.

Step5: cluster peptides based on quantitative pattern-clusterpeptide.py

This step mimics the PQPQ algorithm but much simplified. It only does the peptide clustering which groups peptides based on their quantitative patterns over samples, each peptide will be assigned a number to indicate which cluster it belongs to. If one protein has only one unique peptide, then this peptide will get a cluster '0' instead.
Command: python clusterpeptide.py --i heavy_pepdata.txt --o heavy_pepcluster.txt

--i argument should be _pepdata.txt file from step3
--o argument is the name of output file. There are some other options to use.
--metric defines the way to calculate the distance, "euclidean" is the default. Other options such as "cosine", "correlation" are also available. (if cosine is chosen, make sure vectors do not contain value greater than 1)
For all available clustering metric, check on website
http://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html?highlight=distance.pdist#scipy.spatial.distance.pdist
--method defines the way of calculating the distance between the newly formed clusters. Available options are "single", "complete", "average" and "weighted". "average" is the default.

Step6: Map peptides to its genomic coordinates

1） Map peptides to known splice variants in EVDB database-mapping_EVDB.py

The script in this step maps identified peptides from (uniprot, ensemble and IPI database) to splice variants in EVDB database (assembly hg19 for human).
NOTE: if step5 is skipped, the input file of this step is the output of step3, which is heavy_pepdata.txt. If step5 is not skipped, the input file of this step is the output of step5, which is heavy_pepcluster.txt.

Command: Python mapping_EVDB.py heavy_pepdata.txt heavy (if step5 is skipped)
Command: Python mapping_EVDB.py heavy_pepcluster.txt heavy (if step5 is not skipped)

The first argument is input file and the second is the prefix of output file
Output: heavy_mappingout.txt, heavy_genestatistic.txt (The content of these two output files can be seen in supplementary file 1 table S1 and S2.)
If you encounter “KeyError”, please run step4 again.

The file mappingout.txt is peptide based format in which each row is one unique peptide. This file will be used for visualization. The file genestatistic.txt is gene based format in which each row is one gene.
Very few peptides in mapping output might have no coordinates reported. Since peptide sequence are mapped to splice transcripts sequences extracted in Genbank, there might be very few peptides identified (depending on the reference database used in the searching engine) that are not found in Genbank. And peptides with unknown letters (X) in the sequence will not be given any coordinate.

2) Map peptides to predicted splice variants in ECgene database-mapping_ECgene.py

The script in this step maps identified peptides from ECgene protein sequence database in which peptide spectra was searched to its genomic positions. Before running this step, you need to download ECgene protein sequence file (same evidence

level used in searching peptide spectra) and gene structure file from ECgene database website. http://genome.ewha.ac.kr/ECgene/.

python mapping_ECgene.py --i control_pepdata.txt --prefix heavy --ECgene hg18_b1_high_gene.txt --ECprotein hg18_b1_high_pep.txt
--i input peptide data file name from step 3 or 5
--prefix prefix name for output file
--ECgene gene structure file from ECgene database
--ECprotein protein sequence file from ECgene database

Output: heavy_mappingout.txt control_genestatistics.txt

Step7: Visualize splice variants, peptides, peptide quantitative patterns
1) Use visualization.py when peptides are mapped to EVDB splice variants
Visualize the gene of interest. Given a gene symbol, it will generate a high quality picture which contains the exon composition of known splice variants in EVDB, transcriptional positions of identified peptides and quantitative patterns of peptides in each cluster.

Command: Python visualization.py heavy ARF5
Output:ARF5_pattern_heavy.tiff

Always specify prefix first and then the gene you would like to plot. Gene symbol should be exactly the same as the one you see in the file genestatistic.txt. Next release version will allow users to set parameters of output image (such as format, size and resolution). Look Fig.2 in the paper for interpretation of the figure.

2) Use Visualization-ECgene.py when peptides are mapped to ECgene splice variants.
If you want to visualize a peptide from sample heavy mapped to ECgene splice variant H6C12216.1, you can use the following command.
Command: python visualization-EC.py --sample heavy --id H6C12216 --scale 5 --gene-structure-file hg18_b1_high_gene.txt
You should always use the same gene structure file as used in mapping step.
--scale is to control the scaling of intron and exon size.
Output: H6C12216_pattern_heavy.tiff

All images generated have dpi=300 resolution.
For any problems related to the program, please contact:
yafeng.zhu@scilifelab.se