SNV-DA documentation v1.0.1

Matt R. Paul

28 February 2015

SNV-DA steps:

1. Call SNVs from RNAseq data using SNPiR and GATK tools
2. Annotate variants
3. Data Transformation
4. Model Creatu

1 SNPiR and GATK pipline

A description of the SNPiR pipeline can be found in their publication:

Piskol, R, Ramaswami G, and Li JB. Reliable identification of genomic variants from RNA-seq data. Am J Hum Genet. 2013;93(4):641-51.

Their software and documentation can be downloaded from http://lilab.stanford.edu/SNPiR/. To faciliate use of their pipeline, we have included a script **snpir_pipeline.sh** which runs one sample through the workflow starting from raw read files (fastq). Follow the steps for the installation of necessary software found in the SNPiR documentation. After doing so, include the necessary software in your system path and change location of software includeed in the script (indicated by "< >" brackets). GATK is currently developing best practices for SNV calling

from RNAseq data. Refer to their website for updates on their recommendations. For example, indel realignment, recalibration, and print read steps may be excluded.

Before running the script, make sure to follow the SNPiR documentation to create a customized reference sequence which concatanates the current human reference genome with known splice junctions and create appropriate index files. In our script, this is referred to as **SNPiR.fa.** Include all of the necessary annotation and reference files in a directory named **annot_files** in your working directory.

To run the tool for paired end reads:

sh snpir_pipeline.sh <forward_reads_file_fastq> <reverse_reads_file_fastq> <sample_name>

To run the tool for paired end reads:

sh snpir_pipeline.sh <reads_file_fastq> <sample_name>

Several steps in this workflow are computationally and time extensive. Make sure to run the script in the background to ensure continuation.

The output will be sent to a created "variants" directory in your current directory as **<sample_name>.variants.bed**.

Output file format:

1.) chromosome

2.) position-1

3.) position

4.) coverage, number of non-reference nucleotides

5.) reference nucleotide

6.) non-reference nucleotide

7.) frequency of non-reference nucleotide

Note: Make note of how SNPiR reports position as downstream tools expect different positioning, such as ANNOVAR.

2 SNV-DA tools

In the **test_data_and_pipeline** directory contains variant and coverage data for samples used in our publication. The script, **example.sh**, provides an example script of the steps to identify high coverage SNV loci, create and filter SNV matrices, and create training and tests for model creation.

2.1 Identify Unique SNV Loci

SNPiR tools output coverage information for all SNVs called. However, if one wants to directly compare SNVs to samples in which the SNV is not present, one needs to determine the read coverage of that loci. We first want to identify unique SNVs that have adequate read coverage in at least one sample:

python identify_high_cov_SNVs.py <directory_to_SNPiR_variants> <coverage>

It is recommended to set **coverage** to 10. This script will output the number and identify of SNVs with coverages $\geq$10, in this format:

1.) chromosome

2.) position-1

3.) position

4.) reference nucleotide

5.) non-reference nucleotide

This file, **high_cov_unique_snvs.bed**, will be used to limit inquires of coverage for other samples. The script will also output the same information in a slightly different format to be used as input for ANNOVAR. Finally, the script creates a dictionary **freqs** and a file for each sample in this directory which stores the allele fraction value for each SNV.

2.2 Determine Coverage of SNV Loci for All Samples

Bedtools genomecov will be used to determine coverage of all regions of the genome that have aligned reads. You will not be able to use the same bam file that was produced by aligning to the SNPiR reference. Because it contains added splice junction information, coverage values would be diluted. Use **bwa mem** to align to the normal human genome reference sequence. After sorting the bam files, run this command to identify which SNPs have adequate read coverage for each sample, where P is the number of threads to be used:

ls bams/* | xargs -I % -P 6 -n 1 bash get_cov.sh % high_cov_unique_SNVs.bed <coverage>

This will create a new dictionary that will contain a coverage file for each sample. The contents of each file contains coverage information on each SNV with adequate coverage .

2.3 SNV Annotation

Download ANNOVAR. To download necessary databases, run:

perl annotate_variation.pl -buildver hg19 -downdb -webfrom annovar refGene humandb/

To annotate, the unique SNV file, run:

perl table_annovar.pl high_cov_unique_SNVs.anno_input humandb/ -buildver hg19 -out unique_snvs -remove -protocol refGene -operation g -nastring . -csvout

The output of which will contain region of origin and type of SNV for each unique variant, in the file "unique_snvs.hg19_multianno.csv".

2.4 Create Matrix with High Coverage SNVs

To create a SNV matrix, SNVM, which contains allele fraction values for each SNV for each sample, run:

python create_SNVM.py <coverage_file_directory> <freqs_directory> unique_snvs.hg19_multianno.csv

This file will create a SNVM in csv format. SNV loci with subthreshold coverage values will be given a "NA" for that sample. The format of the SNVM:

1. SNV (GeneName_chromsome:bp_reference->mutant

2. annotation [intergenic, intronic, ncRNA, UTR3, UTR5, upstream, downstream, exonic, nonsyn_exonic, syn_exonic]

3-N. allele_fraction

For example:

AGRN_chr5:123456_A->G,nonsyn_exonic,.5,.75,NA,1,0,.6

2.5 Filter SNVM

After creating the SNVM for every unique SNV, you will notice that several of the SNVs will have NA values for more samples than desired. Thus, you will want to filter out those SNVs that do not have a threshold amount of samples with adequeate read coverage. After making sure that the SNPM is sorted by sample group, run:

python remove_NAs.py SNPM.unfiltered.csv <size of first group> <cut off for first group> <cut off for second group>

For example, the two groups being analyzed are sizes, 11 and 9, and we want remove SNVs that have more than 2 NA values per group, we would run:

python remove_NAs.py SNPM.unfiltered.csv 11 2 2

Now that the SNVM is appropriately filtered, we can now create training and test sets for cross-validation and model creation. Note: make sure that your matrix is sorted horizontally by group.

2.6 Univariate analysis and training set creation

There are two modes: **every_combo** and **rand_subset**. **every_combo** will produce a training set and corresponding test set for every combination of leaving one sample out from each group. Using the example above, there would be a total of $9x11 = 99$ cross-validations.

As this mode can be computationally extensive for larger group sizes, it is recommended to run the **rand_subset** mode in that case. If this mode is flagged, the user must specify the amount of cross-validations sets to create. The script will pull a combination without replacement from all combinations of leaving one sample out from each group.

The command:

python uni_rank_and_set_creation.py <SNVM_file> <first group size> <training_set_size(s)> <annotation(s)> <output_name> <mode> [<amount_of_cross-validations>]

where **training_set_size(s)** is a string of all training set sizes that the user wishes to construct (e.g. "10,25,50,75,100,250,500,750,1000") and **annotation(s)** is a string of all types of SNVs to include in the analysis, for example, if the user wanted to create sets that contain all SNVs one would expect from coding transcripts - "UTR5,UTR3,exonic".

The script creates a directory, **<outputName>_DiffSets**, which contains all of the sets. Within that directory, there are subdirectories for each training set size, which contains a training and test set for each combination, and the names of the features included in that training set.

3 Model Creation and Genetic Algorithm

Now that training and test sets are created, we are ready to run the MATLAB function, **SNV_DA**, which runs the genetic algorithm, creates an OPLS model, and cross validates sets. Before running this function, download our custom OPLS package from https://github.com/Anderson-

Lab/OPLS and change **path_to_opls** variable in the **SNV_DA.m** file to the directory of the matlab folder within the OPLS package.

In a MATLAB terminal, enter the **SNV-DA** directory and run the function:

SNV_DA(data_dir, export_dir, popSize,numClass1, numSamps, desiredN, randAmount)

Where **data_dir** is the directory of the training and test sets and **export_dir** is a name of the model that you are testing. SNV-DA will create a new directory with this name in the current directory. **popSize** refers to the amount of starting chromosomes desired in the genetic algorithm. This paramter is correlated with both the likelihood of arriving at the optimal subset of features and time for computation. It is recommended to set this parameter to 250, though this can adjusted when considering trade-offs. **numClass1** is the number of samples in the first group, and numSamps is the total amount of samples in the SNVM. The **desiredN** parameter specifies the amount of features the should be chosen from the genetic algorithm. In our publication, 10 was chosen because that represents a experimentally feasible amount of SNVs to validate in further studies. Additionally, the **randAmount** paramter provides some flexibility to the desired amount of features by adding more features uniformly random from 1 to **randAmount**.

The function outputs several files:

1. A plot depicting the ROC curve.

2. A report of sufficient statistics of the models created, including prediction accuracy, AUC, 95% CI, and group predictive sensitivies.

3. A ranked list of all the features chosen during model creation and their frequencies. The top features can then be implicated as potential biomarkers.