SNV-DA Reference Manual

Matt R. Paul

14 September 2016

Outline

SNV-DA steps:
1. Call SNVs from RNAseq data using SNPiR and GATK tools
2. Annotate variants
3. Data Transformation
4. Model Creation and Evaluation

SNPiR and GATK pipelines

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 included 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 file of reference sequences by concatenating 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 single-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.

Preprocessing

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 test sets for model creation.

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 directory **freqs** and a file for each sample in this directory which stores the allele fraction value for each SNV.

Determine Coverage of SNV Loci for All Samples

Bedtools **genomecov** is 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 deflated. 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 <cov- erage>

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.

SNV Annotation

Download ANNOVAR. To download necessary databases, run:

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

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".

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 sub-threshold 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
After creation of the SNVM, make sure that none of the SNVids include commas because that compromises the format of the matrix.

SNV-DA

Now that we have a SNVM, we can find the optimal value for the number of features to select, create and evaluate models that identify SNVs that lead to accurate classification. Running **Rscript snvDA.R -h** outputs information on all of the parameters.
Usage: snvDA.R [options]
SNV-DA is used to create and evaluate sPLS-DA models to identify single nucleotide variations (SNVs) that accurately classify phenotype. The steps of the pipeline include:
1) Finding optimal value of number of features to be selected in the model (K)
2) Evaluating the model using cross-validations and optimal value of K
3) Find and rank the K features that are correlated with predictive accuracy
4) Permutation tests to determine if model is discriminate towards the true grouping of labels.

The script has several parameters that allows the user to design their own analysis:
1) The number of non-zero values by which to filter the matrix
2) The number of NA values allowed for each SNV
3) The type of SNV to include in the model
4) The range and/or values of K tested.
5) Cross-validation design (number of samples to take out from each group, the stratification of test samples [force test samples to be pulled equally from each group], and the number of cross-validations).

Options:
Options: -O, --findOptimalK
If flagged, SNV-DA will run nest cross-validations to determine optimal K.
-J, --evaluatePerformance
If flagged, SNV-DA will evaluate performance of model the value of K found from running --findOptimalK or user supplied --optimalK.
-P, --permutationTests
If flagged, SNV-DA will run permutation tests by randomly permuting sample classes then running cross-validations. Tests are run --permIter number of times. Performances are then compared to the value of AUC found by --evaluatePerformance of user supplied --AreaUnderCurve.

-x, --predictTestSet
If flagged, SNV-DA will train a model with optimal K on all samples in the main matrix and then predict the labels of the second matrix.
-C, --getSNVs
If flagged, SNV-DA will ONLY run sPLS-DA on all samples to identify K top predictive features
-W NFOLDCV, --NfoldCV=NFOLDCV
If specified, N-fold cross-validations will be performed by partitioning the input matrix into NfoldCV setsi during the optimization of K and performance evaluation.
-I ITERNFOLDCV, --iterNfoldCV=ITERNFOLDCV
If NfoldCV is specified, NfoldCV will be run iterNFoldCV times during performance evaluation and finding of optimal K (default=1).
-V INTERNALNFOLDCV, --internalNfoldCV=INTERNALNFOLDCV
If specified, N-fold cross-validations will be performed during the interval CV while finding optimal K by partitioning the input matrix into iternalNfoldCV setsi during the optimization of K and performance evaluation. If left unspecified, leave-one-out crossvalidations will be used for the interval cross-validation.
-j INTERNALITERNFOLDCV, --internalIterNfoldCV=INTERNALITERNFOLDCV
If internalNfoldCV is specified, internalNfoldCV will be run internalIterNFoldCV times during performance evaluation and finding of optimal K (default=1).
-M MATRIX, --matrix=MATRIX
SNVM input (csv)
-y TESTMATRIX, --testMatrix=TESTMATRIX
matrix to predict labels (csv)
-D SIZECLASS1, --sizeClass1=SIZECLASS1
The number of samples of class 1. (Used to find the number of samples of both classes).
-L MINK, --minK=MINK
The minimum value of K tested.
-B MAXK, --maxK=MAXK
The maximum value of K tested.
-E EVERYKTH, --everyKth=EVERYKTH
Specifies every Nth K tested. Or use:
-R RANGE, --range=RANGE
A comma separated list of K's to be added to the Ks tested, e.g. 5,10,15,20,25,50,75,100,250,300.
-Z NUMNONZEROS, --numNonZeros=NUMNONZEROS
Limit SNVs to those that have at least X number of samples with allele fractions greater than zero (default = 3). Decreases runtime. Suggested value is a 1/3 the number of samples in the smaller group.
-N NUMNAS, --numNAs=NUMNAS
Limit SNVs to those that have at most X amount of samples with 'NA' values (default=0).
-G INCLUDE, --include=INCLUDE
SNVs with this substring are included, e.g. "exonic" for nonsynonymous and synonymous exonic SNVs
-X EXCLUDE, --exclude=EXCLUDE
SNVs with this substring are excluded, e.g. ncRNA to remove intronic$_n cRNA$
-A STUDYNAME, --studyName=STUDYNAME
One word name of the analysis used in output file names and figures.
-U NAMECLASS1, --nameClass1=NAMECLASS1
Name of class 1 (default = Class$_1$).
-H NAMECLASS2, --nameClass2=NAMECLASS2
Name of class 2 (default = Class$_2$).
-T THREADS, --threads=THREADS
Number of threads for cross-validations (default = 1).
-K OPTIMALK, --optimalK=OPTIMALK
Used when --evaluatePerformance or --permuationTests is flagged. Specifies the value of K used for cross-validation (default = 500).
-q PERMITER, --permIter=PERMITER
Used when --permuationTests is flagged. Specifies the number of permutation tests to be run

-a AREAUNDERCURVE, --AreaUnderCurve=AREAUNDERCURVE
Performance of true model. Compared to permuted models when --permutationTests are run. If unflagged, value is determined by --evaluatePerformance

-F, --produceFigures
If flagged, allele fraction box plots, heatmaps, and kernal density figures are produced.

-h, --help
Show this help message and exit

1 Examples

This section will outline a couple of examples of how **snvDA.R** can be used.

1.1 Finding Optimal K and Evaluation

We can run:

Rscript snvDA.R -M SNVM.csv **-A** intronic_SNV_model **-D** 11 **-U** Disease_Free **-H** Relapse **-G** intronic **-Z** 3 **-O -J -W** 10 **-V** 10 **-I** 15 **-T** 22 **-L** 10 **-B** 1000 **-E** 40

This command will run SNV-DA on the input matrix **SNVM.csv**. We first specify the name of the run, the number of samples in the first class, and the names of two classes. We can then select SNVs that have the word "intronic" in their description as well as being non-zero in 3 samples. We then specify that we want to find the optimal number of features, K, in the model using internal cross-validation (-O) and then evaluate that optimal number of K using cross-validation (-J). We can then specify the number of folds to be used for the internal (-V) and training set cross-validations (-W) to be run for (-I) iterations. To speed up the whole process, we specify that we would like to run the cross-validations using 22 threads (-T). Finally, we specify the range of Ks that we would like to evaluate, in this case, we would like to start with K=10 then go up to K=1000, evaluating every 40th K.

1.2 Permutation Tests

We can run:

Rscript snvDA.R -M SNVM.csv **-A** intronic_SNV_model **-D** 11 **-U** Disease_Free **-H** Relapse **-G** intronic **-Z** 3 **-P -q** 1000 **-K** 250 **-a** 0.8345 **-W** 10 **-T** 22

This command sets up the model just like the previous example, except now we are going to evaluate the results using permutation tests (-P). We first want to specify the number of iterations (-q), the optimal number of features (-K), and the AUC of the true model (-a). This will run a 1000 permutations test, where for each iteration, the class labels are shuffled and a model is trained and tested with K using 10-fold CV. The amount of permuted models with AUC greater than the AUC provided determines the permutation p-value.