miRWoods Instruction Manual

Author:
Jimmy Bell

June 23, 2019

Contents

1 Dependencies

miRWoods has the following dependencies:
1. bowtie
2. samtools 0.1.18
   Edit the Makefile so that CFLAGS= '-g -Wall -O2 -fPIC #-m64 #-arch ppc'
   The reason for using version v0.1.18 is that Bio::DB::Sam requires an earlier version
   than what's currently available.
3. The Vienna RNA package with the RNA Perl Library
4. Cutadapt is used for adapter trimming and may be downloaded using the following command:
   pip3 install –user –upgrade cutadapt

The following perl libraries are required:
1. Bio::DB::Sam
2. Statistics::R
3. Math::CDF

you will also need to download R and install the following libraries:
1: ROCR
2: randomForest
3: mlbench

2 Installation

An install script is being worked on but not yet ready. For now do the following:
1. Add the Scripts directory from miRWoods into your path
2. Add the PerlModules directory to your PERL5LIB path

3 Preparing Bam Files

3.1 Preparing Bam Files Using createBam.pl

Example:
fastq-dump SRR326279
createBam.pl SRR326279.fastq ATCTCGTATGCCGTCTTCTGCTTGT 30 hg19

Directions:
createBam.pl is a script which runs each process described in section 3.2. createBam.pl is run in the following way:
createBam.pl ⟨fastq file⟩ ⟨adaptor⟩ ⟨quality score⟩ ⟨bowtie index⟩

3.2 Preparing BamFiles without Using createBam.pl

Example:

fastq-dump SRR326279
cutadapt -e 0.2 -m 17 -q 10 -a ATCTCGTATGCCGTCTTCTGCTTGT SRR326279.fastq -o SRR326279_trimmed.fastq
fastqAvgQualityFilter.pl SRR326279_trimmed.fastq SRR326279_qualFilt.fastq 30
bowtie -n 1 -e 50 -l 18 -m 10 -a -S –best –strata hg19 SRR326279_qualFilt.fastq SRR326279.sam
addNHTags.pl SRR326279.sam SRR326279_NHTags.sam
trimMirReads.pl SRR326279_NHTags.sam SRR326279_trimmed.sam

samtools view -bS -o SRR326279_trimmed.bam SRR326279_trimmed.sam
samtools sort SRR326279_trimmed.bam SRR326279_sorted
samtools index SRR326279_sorted.bam

Directions:

The following are instructions on a step by step process to creating the bam files. The user may use their own methods if they wish. If using your own method, please note that the bam file requires NH tags for each read showing the number of hits to the genome.

1. After the fastq file is downloaded, reads are trimmed using cutadapt with the following options:
cutadapt -e 0.2 -m 17 -q 10 -a ⟨adaptor⟩ ⟨fastq file⟩ -o ⟨fastq output file⟩

Setting -e to 0.2 allows for 20% error in the adapter sequence to be cut. Setting -m to 17 removes all sequences less than 17 nucleotides in length. Setting -q to 10 removes any nucleotides from the $3'$ end with quality scores less than 10 prior to removing the adapter. The -a option specifies the adaptor sequence.

2. Reads without an average quality score of 30 may be removed using the following command:
fastqAvgQualityFilter.pl ⟨input fastq⟩ ⟨output fastq⟩ 30

3. Fastq files are aligned to the genome using bowtie with the options:
bowtie -n 1 -e 50 -l 18 -m 10 -a -S –best –strata ⟨bowtieIndex⟩ ⟨fastq file⟩ ⟨output sam⟩

Setting -l to 18 specifies a bowtie seed length of 18. Setting -n to 1 allows for 1 mismatch within the bowtie seed region. Setting -e to 50 allows for a total quality score of 50 for all mismatched nucleotides. Setting -m to 10 suppresses all mappings that hit more than 10 places in the genome. The -a options sets bowtie to report all valid alignments. The -S option sets bowtie to output the mapped reads in SAM format. The –best and –strata set bowtie to report all valid alignments based on the previously mentioned options.

4. A custom script is used to add NH tags to the sam file so that miRWoods will have information about the number of mappings for each read (Note: this script must be used prior to sorting):
addNHTags.pl ⟨input sam file⟩ ⟨output sam file⟩

5. If there are any mappings with several low quality mismatches on the end, they may be trimmed down using the following script:
trimMirReads.pl ⟨input sam file⟩ ⟨output sam file⟩

6. The sam file is converted to a bam file using the following command:
samtools view -bS -o ⟨output bam file⟩ ⟨input sam file⟩

7. The bam file is sorted using:
samtools sort ⟨input bam file⟩ ⟨output prefix⟩

8. Finally, the bam file is indexed:
samtools index ⟨bam file⟩

4 Creating a Bam File List

A list of all bam files is used as input for miRWoods. The bam file list is a tab delimited file containing sample names followed by their location.

Example:

hsa_cellLines1	/⟨path to directory⟩/hsa_cellLines1.bam
hsa_cellLines2	/⟨path to directory⟩/hsa_cellLines2.bam
hsa_cellLines3	/⟨path to directory⟩/hsa_cellLines3.bam
hsa_cellLines4	/⟨path to directory⟩/hsa_cellLines4.bam
hsa_cellLines5	/⟨path to directory⟩/hsa_cellLines5.bam
hsa_cellLines6	/⟨path to directory⟩/hsa_cellLines6.bam

(see example_bam.txt)

5 Creating a Config File

miRWoods takes its inputs in the form of a config file. This is a tab delimited file with the name of each parameter and its value separated by an equal sign

For example:

scriptDir	=	/⟨path to miRWoods Dir⟩/miRWoods/Scripts
hairpinRF	=	/⟨path to miRWoods Dir⟩/miRWoods/Models/hairpinRF.model
productRF	=	/⟨path to miRWoods Dir⟩/miRWoods/Models/productRF.model
SizesFile	=	/⟨path⟩/hg19.chrom.sizes
genomeDir	=	/⟨path to Genome Dir⟩/hg19
geneModels	=	/⟨path to Annotations⟩/hg19_ensembl.gff3
mirbaseGff	=	hsa.gff3
bamListFile	=	bamlist.txt
minReadLength	=	14
maxReadLength	=	28
HPDVCutoff	=	0.28
ARPMCutoff	=	0.11

(see example_config.txt)

6 Running miRWoods

6.1 Running miRWoods using miRWoods.pl Script

The easiest way to run miRWoods is to give it a config file using the -L option. This method runs all the scripts in section 6.2.

Example:
miRWoods -L config.txt

6.2 Running miRWoods using One Script in the Pipeline at a Time

The following are instructions on how to run each script in the miRWoods pipeline. This is not necessary unless you're troubleshooting an error.

Steps in the pipeline:
1. checkMiRBaseGff.pl checks the miRBase for issues that might affect miRWoods and in some cases creates a new miRBase file to run with miRWoods:
checkMiRBaseGff.pl ⟨configFile⟩

2. countMappedReads.pl counts the number of read mappings in each bam file and creates a tab delimited file of library sizes:
countMappedReads.pl ⟨bam file list⟩

3. printReadRegions.pl find all regions of contiguous read loci and prints them out as a bed file:
printReadRegions.pl -L ⟨configFile⟩

4. extractProductFeatures.pl groups products together and scores them:
extractProductFeatures.pl -L ⟨configFile⟩

5. generateProductTrainData.pl creates training data in order to create a model for the mature product random forest (MPRF). This is only needed if you wish to build a new model:
generateProductTrainData.pl -L ⟨configFile⟩

6. evaluateProductsWithRF.pl evaluates products based on the product random forest model. if the trainModels parameter in the config is set to 1 it will also create a new model. If you are training a new model be careful not to set productRF to one you don't wish to have overwritten:
evaluateProductsWithRF.pl -L ⟨configFile⟩

7. processReadRegions.pl takes the products that were positive hits according to the mature product random forest (MPRF), combines them with surrounding products to create hairpins, and then scores each hairpin:
processReadRegions.pl -L ⟨configFile⟩

8. generateAnnotations.pl annotates Hairpins. A later script will apply these annotations after the hairpin precursor random forest (HPRF) scores each hairpin:
generateAnnotations.pl -L ⟨configFile⟩

9. extractHairpinFeatures.pl does further processing on the hairpins and creats a scores file. It also counts neighboring loci and includes this score as a feature called neighborCount:
generateAnnotations.pl -L ⟨configFile⟩

10. generateHairpinTrainData.pl creates training data in order to create a model for the hairpin precursor random forest (HPRF). This is only needed if you wish to build a new model:
generateHairpinTrainData.pl -L ⟨configFile⟩

11. evaluateHairpinsWithRF.pl evaluates hairpins based on the hairpin precursor random forest model. If the trainModels parameter in the config is set to 1 it will also create a new model. If you are training a new model be careful not to set productRF to one to don't wish to have overwritten:
evaluateHairpinsWithRF.pl -L ⟨configFile⟩

12. printMiRWoodsPredictions.pl gathers and prints the final output files of miRWoods:
printMiRWoodsPredictions.pl positivePredictions.gff positivePredictions_hairpins.txt positivePredictions_products.txt

7 Viewing Output

The final output from miRWoods is in two files:

1. predictedMiRs.gff - a gff file containing the hairpins and products predicted by miRWoods
2. predictedMiRs_productInfo.txt - a tab delimited file containing read count information for each product. This file includes loop and anti-sense products.