RNAmapper step-by-step instructions for RNA-Seq based mapping and identification of mutants using the command line.

Knowledge of the use of the command line in a linux/unix environment is necessary. Each command will require modification to allow for local installation of the programs, environmental paths, and reference data. The overall procedure uses two parallel analysis packages - see Supp.Fig.1 in Miller et al. (Miller *et al.*, Genome Research, PMID 23299976) for a visual representation of the pipeline.

Note that while this pipeline works in our, and now other people's hands, this is a work in process and it is not developed to the level of a professional program. You are responsible for understanding each of the steps, what is happening with the data, and using/modifying as appropriate for you data. We cannot offer any guarantees, warrantees, or user support.

Good luck!

PREREQUISITES

System and Programs:

1) unix/linux based system – can be run on desktop computer but will tie up computer for ~3-5 days to run a single mutant - ideal is to have linux server
2) R 2.x - http://www.r-project.org/
3) RNAmapper scripts - http://www.RNAmapper.org/
4) FastQC - http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
5) Bowtie2.x - http://bowtie-bio.sourceforge.net/index.shtml
6) TopHat 2.x - http://tophat.cbcb.umd.edu/
7) Cufflinks/Cuffcompare/Cuffmerge/Cuffdiff 2.x - http://cufflinks.cbcb.umd.edu/index.html
8) samTools/bcfTools - http://samtools.sourceforge.net/
9) Variant Effect Predictor (VEP) - http://www.ensembl.org/info/docs/variation/vep/index.html
#optional 10) SNPeff as an alternative to VEP (has not been implemented in command-line procedure)
#optional 11) SIFT to evaluate deleteriousness of missense mutations - http://sift.jcvi.org/ (has not been implemented)

Data files and folders:

1) Start with a single empty folder (top directory – can be called anything).
2) Place the RNA-Seq fasta reads for wildtype and mutant in separate directories called wt_fasta and mut_fasta within top directory.
2.1) Alternatively, you skip directly to step 4 of the below protocol if you already have reads aligned to the genome (.bam format).
3) A reference genome (FASTA) and gene model (GTF) are required: http://www.ensembl.org/info/data/ftp/index.html

4) A SNP list must be supplied to allow for filtering. You can find "mine" at http://www.RNAmapper.org/snps, or generate your own. This must be in the top directory.

Place these in any useful, accessible directory.
5) The genome must be indexed by SAMtools, and the pipeline will perform faster if supplied with a bowtie index (http://tophat.cbcb.umd.edu/manual.html --transcriptome-index).

Shell and paths:
```
# see specifications for each program for details
bash
export PATH=$PATH:/path/to/FastQC
export PATH=$PATH:/path/to/bowtie
export PATH=$PATH:/path/to/TopHat
export PATH=$PATH:/path/to/samtools
export PATH=$PATH:/path/to/samtools/bcftools
export PATH=$PATH:/path/to/cufflinks
export PATH=$PATH:/path/to/ensembl
source setPERL5LIB.sh
```

Environmental variables:
```
bowtieGenome=/path/to/Zv9.6x/bowtie/
refGenome=/path/to/Zv9.6x/Danio_rerio.Zv9.6x.fa
gtfFile=/path/to/Zv9.6x/Danio_rerio.Zv9.6x.gtf
```

STEPS

1) Create a master list of wildtype SNPs to use as a filter. This will be used to compare against SNPs identified in the linked region of the mutant genome, and remove them from further consideration.
- The SNP.vcf files need to be in the top directory, in strict vcf 4.0 or 4.1 format.
- Currently each SNP list needs to be stored as an individual chromosome file.
- My current SNP lists can be downloaded from http://www.RNAmapper.org/snps or you can create your own.

- Instead, you can create your own list using the R script VCFmerge.R which requires that you place multiple VCF formatted SNP files in same folder. It will then concatenate them into one super list. Note that currently the R script will need to be modified internally to accept the files being input.

commands
```
Rscript --vanilla /path/to/VCFmerge.R
```

1.1) *** another method that is likely generalizable to other people's SNP lists ***
You can download a genome wide SNP list from:
https://wiki.med.harvard.edu/SysBio/Megason/MegaMapper

Then “grep” out each individual chromosome file, allowing the scripts below to utilize the SNP lists

commands

```
for i in {1..25}; do grep -w ^$i WT_SNP_set.vcf > SNPs_chr$i.vcf; done
```

Quality control and alignment

2) FastQC and filtering - filter the reads for quality

- If .bam alignment file is already created, skip directly to step 4.
- wt fastq files must be in directory wt_fasta (for below to work).
- mut fastq files must be in directory mut_fasta (for below to work)..
- The below assumes paired-end reads.

commands

```
cd wt_fasta
fastqFiles1=`ls -m *_R1_*.fastq.gz | sed -e 's/,/ /g' | tr -d '\n'`
fastqFiles2=`ls -m *_R2_*.fastq.gz | sed -e 's/,/ /g' | tr -d '\n'`
mkdir -p fastqc
fastqc -t 8 --casava -o fastqc $fastqFiles1
fastqc -t 8 --casava -o fastqc $fastqFiles2
mkdir -p filtered
for i in *fastq.gz
do
        i2=${i//.gz/}
        zgrep -A 3 '^@.*[^:]*:N:[^:]*:' $i | zgrep -v '^\-\-$' >
filtered/$i2
done
gzip -r filtered
cd..

cd mut_fasta
fastqFiles1=`ls -m *_R1_*.fastq.gz | sed -e 's/,/ /g' | tr -d '\n'`
fastqFiles2=`ls -m *_R2_*.fastq.gz | sed -e 's/,/ /g' | tr -d '\n'`
mkdir -p fastqc
fastqc -t 8 --casava -o fastqc $fastqFiles1
fastqc -t 8 --casava -o fastqc $fastqFiles2
mkdir –p filtered
for i in *fastq.gz
do
        i2=${i//.gz/}
        zgrep -A 3 '^@.*[^:]*:N:[^:]*:' $i | zgrep -v '^\-\-$' >
filtered/$i2
done
gzip -r filtered
cd ..
```

3) TopHat - align reads to the genome and create a .bam alignment file.
- The below assumes paired-end reads.
- Our library prep creates ~200bp fragments, meaning ~80bp inner distance (-r). This may need to be altered depending on your library.

commands

```
cd wt_fasta/filtered
mkdir -p ../../tophat_wt
tophatOut=../../tophat_wt
fastqFilesR1=`ls -m *_R1_*.fastq.gz | sed -e 's/, /,/g' | tr -d '\n'`
fastqFilesR2=`ls -m *_R2_*.fastq.gz | sed -e 's/, /,/g' | tr -d '\n'`
tophat -r 80 --num-threads 8 -G $gtfFile -p 8 --library-type fr-unstranded -o
$tophatOut $refGenome $fastqFilesR1 $fastqFilesR2
cd ../..
mkdir -p bam
cd tophat_wt
mv accepted_hits.bam ../bam/wt.bam
cd ../bam
samtools sort wt.bam
samtools index wt.bam
cd ..

cd mut_fasta/filtered
mkdir -p ../../tophat_mut
tophatOut=../../tophat_mut
fastqFilesR1=`ls -m *_R1_*.fastq.gz | sed -e 's/, /,/g' | tr -d '\n'`
fastqFilesR2=`ls -m *_R2_*.fastq.gz | sed -e 's/, /,/g' | tr -d '\n'`
tophat -r 80 --num-threads 8 -G $gtfFile -p 8 --library-type fr-unstranded -o
$tophatOut $refGenome $fastqFilesR1 $fastqFilesR2
cd ../../tophat_mut
mv accepted_hits.bam ../bam/mut.bam
cd ../bam
samtools sort mut.bam
samtools index mut.bam
cd ..
```

Mapping

4) Use samtools/mpileup/bcftools to output ALL CALLS made in the wildtype and mutant RNA-Seq data.
- I have found that using bcftools to call variants removes a large number of good data points – I call SNPs using my own R script below. This creates very large files, but the data is better for mapping.
- The -A option is critical as it allows for “anomalous pairs” to be mapped, that is RNA-Seq paired reads that map "far" apart. For RNA-Seq, any pair of reads across a large intron is far apart and would be removed if –A is not used.

commands

```
samtools mpileup -ABuf $referenceGenome wt.bam | bcftools view -b - > wt_raw.bcf
samtools mpileup -ABuf $referenceGenome mut.bam | bcftools view -b - > mut_raw.bcf
```

5) Use the bash shell and bcftools to loop through and create a .vcf format file for each individual chromosome. Again, we found that using the bcftools variant caller inappropriately excluded many SNPs. We therefore first extract all call information for all positions along the chromosome and save an individual chromosome file. Chromosome files are kept separate due to their large size.

commands

```
for i in {1..25}; do bcftools view wt_raw.bcf | grep -w ^$i > wt_chr$i.vcf; done &
for i in {1..25}; do bcftools view mut_raw.bcf | grep -w ^$i > mut_chr$i.vcf; done
```

6) Use RNAmappe.R to identify SNPs and map their frequency in the mutant pool
- This assumes strict .vcf 4.0 or 4.1 format for each individual chromosome file, no headers are necessary.
- The script must be run from directory where chromosome files (step5) reside.
- Within the directory where the individual chromosome files reside there must be a directory called _settings
- Within _settings there must be a text file called mappeRsettings.txt with 2 lines:

```
coverage=25
zygosity=25
```

-- This file allows the user to change parameters of the coverage to use as a SNP marker. We recommend coverage of 25, meaning that any marker used was read at least 25 times. The user can also modify the zygosity of the marker, meaning the percentage of alternative calls to be considered a good marker. Again, 25 percent seems to give the best results.

commands

```
mkdir _settings
cd _settings
# 25 is recommended for both
# coverage of 25 means at least 25x coverage for calling mapping marker
# zygosity of 25 means at least 25% alternative call at marker
echo “coverage=25” > mappeRsettings.txt
echo “zygosity=25” >> mappeRsettings.txt
cd ..
for i in {1..25}
        do Rscript --vanilla /path/to/RNAmappe.R wt_chr$i.vcf,mut_chr$i.vcf
done
```

Comparing expression between wildtype and mutant transcriptomes

7) Use the Cufflinks package to examine transcription start sites, splicing patterns,

isoform usage, and expression differences between mutants and wildtype siblings.
- The data from steps7-10 will be used below in step13 but its good to get it running now as it takes some time to process.

commands
cufflinks -o _cufflinks_wt -p 8 -g $gtfFile -u [wt.bam]
cufflinks -o _cufflinks_mut -p 8 -g $gtfFile -u [mut.bam]

8) Use the Cuffmerge/Cuffcompare program to create a new, merged, GTF file from reference plus RNA-Seq data.
first need to tell cuffmerge where the GTF files are located

commands
echo "cufflinks_wt/transcripts.gtf" > assemblies.txt
echo "cufflinks_mut/transcripts.gtf" >> assemblies.txt
cuffmerge -o _cuffmerge -g $gtfFile -p 8 -s $refGenome assemblies.txt

9) Use the Cuffdiff program to compare expression form mutant and wildtype siblings.

commands
cuffdiff -o _cuffdiff -b $refGenome -L wt,mut -p 8 -u cuffmerge/merged.gtf [wt.bam] [mut.bam]

Data analysis break
Congrats, you have completed the first part of the method!
Now you must assess your mapping and determine the linked region of the genome.
- Once RNAmapper has finished, it will output a file called **_RNAgenomeFrequency.jpg**
This contains the mapping information for the whole genome. The frequency of the mutant SNP Markers are plotted against chromosomal position. Regions linked to the mutation should approach homozygosity, or a frequency of 1.0, near the top of the graph. This should show one region of the genome linked to your mutation.
- There are also individual chromosome allele frequency graphs called **mut_chrX.jpg** (X from 1 to 25)
These will allow you to assess the fine structure of linkage on interesting chromsomes.
- Additionally there are files called **mut_chrX_stats.txt**
These contain SNP information for individual chromsomes.
- Finally, there are files called **mut_chrX_atMarkers.txt**
These are individual chromosome tables sorted by chromosomal position containing the mutant allele frequencies at each SNP Marker – this is the data used to generate the graphs above. The table is essentially a .vcf formatted file with additional columns. For details on the first eight columns see this :

http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41

Briefly, some columns of interest:
#CHROM – chromosome
POS – chromosomal position
REF – the reference allele at this position
ALT – the alternative allele call at this position
INFO – the critical information on the number of reads called at this position. This is the data that is extracted and used for allele frequency calculations by the RNAmappe.R script.
Fref – The number of reference reads aligned in the forward orientation.
Rref – The number of reference reads aligned in the reverse orientation.
Falt – The number of alternative reads aligned in the forward orientation.
Ralt – The number of alternative reads aligned in the reverse orientation.
refTot – Total number of ref reads.
altTot – Total number of alt reads.
Tot – Total number of reads.
refRat – The ratio (frequency) of ref alleles.
altRat – The ratio (frequency) of alt alleles.
highAllele – The greater of refRat and altRat. This is plotted against chromosomal position as black marks in the individual chromosome graphs.
highAve – An average of 50 neighboring highAllele points, with a step size of 1. This smooths the noise and is plotted in the genome wide graph as well as the red line of the individual chromosome graphs.

From the above data you have (hopefully) identified a single region of the genome that is linked to your mutation of interest. We will use the scripts below to extract information from the RNA-Seq data within the region of linkage to identify candidate mutations that may be causative for the phenotype of interest.

Currently we focus on user-defined regions of the genome. While this could be done computationally, the user is the best judge of the extent of linkage.

---

10) Use RNAidentifie.R to extract all SNPs and INDELs from linked region. Note that INDELs are problematic with current short read platforms, but the script outputs them for the user to evaluate.
**** Recent problems from users caused me to disable INDEL output currently. As mentioned before this data is problematic so for now this will remain off. *** SNPs are still processed with the script.*

-For this script, the user inputs the chromosome they determined to be linked (X), coordinate of left edge of region of interest (Y), and coordinate of right edge of region of interest (Z)

-To input these regions, a text file must be created within the previously made (above) directory _settings
The text file must be called identifieRsettings.txt and contain 3 lines:
chromsome=X
left edge of linkage=Y
right edge of linkage=Z
- RNAidentifie.R creates a directory _RNAidentifieR where files will be output.

commands
cd _settings
echo “chromsome=X” > identifieRsettings.txt
echo “left edge of linkage=Y” >> identifieRsettings.txt
echo “right edge of linkage=Z” >> identifieRsettings.txt
cd ..
Rscript --vanilla /path/to/RNAidentifie.R

11) Use ENSEMBL's Variant Effect Predictor to examine SNP consequences.
-The below must be run from _RNAidentifier folder.
-The file VEPinput.vcf is output by RNAidentifie.R above.

commands
cd _RNAidentifier
variant_effect_predictor.pl -i VEPinput.vcf -o VEPoutput.txt -species zebrafish -check_existing

12) VEPsorte.R sorts the output from VEP and puts it back together with the sequencing information.
- This must be run from the top directory

commands
cd ..
Rscript --vanilla /path/to/VEPsorter.R

Congrats, you now have some candidate mutations to evaluate!
Use excel to open the output table called _SNPcandidates.txt found in the _RNAidentifier directory.

This file will allow you to identify candidate SNP mutations that affect genes within the linkage region you defined. The list will be sorted with SNPs that create nonsense mutations at the top, followed by missense mutations, and followed by other potential changes. Generally, the number of nonsense and missense changes is small and these are of most interest. But there are many changes identified in the UTRs and the less-conserved regions of the genome, which are included for the user to evaluate. However, changes that affect these other categories are 1) many and 2) hard to know if they are informative. Caution needs to be used in interpreting the list.

THE LIST OUTPUT BY THE ABOVE PROCESS IS FAR FROM PERFECT. *There are several steps where SNP data was retained that does not represent good candidate mutation data.* As one example, when multiple alternative alleles are called in the mutant data there is no way (within the pipeline) to differentiate the number of calls for ALT call #1 or ALT call #2. Lets say that within the mutant data at position X there were 0 REF (C) calls and 14 ALT calls of which 13 were A, and 1 T. Within the pipeline there is no way to know the number of A or T reads, however if the codon at this site is CAG, then the alternative codons would be AAG and TAG. Thus, the latter codon would come through the pipeline as a stop. However, given that there is only 1 T call, this data is likely read error generated by the Illumina process and must be removed from further consideration. **YOU THE USER MUST REMOVE THIS BY EXAMINING THE TABLE DIRECTLY AND EVALUATING THE OUTPUT FOR QUALITY.**

To evaluate each candidate use IGV (http://www.broadinstitute.org/igv/). IGV allows for aligned RNA-Seq data to be viewed aligned to the genome. Each position identified as a candidate above must be viewed in IGV – this will allow for errors like, but not limited to, the one above to be identified and removed from further consideration. I recommend loading the mutant and wildtype sibling .bam files into IGV. This allows for each site to be evaluated for quality. Additionally, I load independent wildtype sequencing data to assess for known wildtype alternative alleles at the site – if there are independent wildtype alleles that are the same as that which is becoming homozygous in the mutant data, it is unlikely to cause the phenotype of interest and should be thought of as a very low priority candidate.

Quality candidates will have ALT calls that approach homozygosity and do not have known wildtype alleles of the same type at the site.

The table is organized by likely severity of consequence, with SNPs creating/destroying STOPs at the top, followed by missense, followed by many other categories as output by VEP such as UTRs, Upstream, etc. The table consists of many columns, but is basically a modified VEP output file, with the sequencing information attached for easy comparison. Some of the less obvious columns are described below:

The first 14 columns are information on the nature of the SNP
POS – chromosomal position, from RNA-Seq mapping data, and put as the first column to more quickly understand where the potential candidate lies within the linkage region.
#Uploaded_variation – all SNPs identified at this location and sent to VEP. The next column (Allele) is being evaluated in this row, the other alleles are elsewhere in the list and can be accessed by searching for the POS.
Allele – the SNP that is being evaluated in this row
Gene – ENSEMBL's gene ID

Feature – ENSEMBL's transcript ID. Note that often the same position is repeated many times in the table – this is because VEP evaluates each transcript, and reports the effect on each. Thus, if there are three transcripts for gene X, and a nonsense mutation is created that is in frame for each, all three will be reported. You will notice that the same POS/Gene is repeated three times in such a case, with the transcript varying in each row.
Feature_type – Transcripts, other.
Consequence – effect of the SNP on the feature. STOPs are sorted to the top of this table, followed by NON_SYNONOMOUS (missense) followed by many other categories that could be of interest but are likely not useful. However, they are output for completeness.
Existing_variation – Ensemble compares the uploaded variation against their dbSNP information and marks here if there is known variation. However, it does not report the nature of the variation, but this information can be used to look at ENSEMBL's records and find if the SNP identified in your RNA-Seq is a known variation, and therefore unlikely to be causative.

The next 19 columns are RNA-Seq information for evaluating the quality of the candidate, and are the same columns as described after step 9. As discussed above, each position should be further investigated with IGV for quality.

The next 7 columns are information from the known WT variant used to filter the candidate list, with the columns similar to those described after step 9. This information is included to quickly evaluate if there is known WT variation at the location. Because of limitations in processing, some WT variation cannot be used as a filter. For example, if there are two WT ALT alleles called in addition to the REF allele (e.g. REF is A, ALT is T,G), then there is no information on the quality of the T or G call, and this cannot be used as a filter – the T or the G may be sequencing error. Again, this information should be evaluated using IGV, but is included here to quickly assess the likelihood of problems.

Now lets identify candidates that effect RNA expression levels, splicing, etc.
** Make sure Cuffdiff has completed before moving to the below steps**

---

13) Use RNAeffecto.R to extract all linked transcript information output by Cuffdiff.
- Again, we rely on the user inputted chromosome (X), coordinate of left edge of region of interest (Y), coordinate of right edge of region of interest (Z) as input above in 10.
- RNAeffecto.R creates a directory _RNAeffectoR where files will be output.
- This must be run from the top directory.
- cuffdiff must be finished before running this command.

-- NOTE that this command simple extracts information generated by cuffdiff from the region of interest – you can always look back to the original cuffdiff files for more info.

commands
Rscript --vanilla /path/to/RNAeffecto.R

Congrats, you now have some MORE candidate mutations to evaluate!

Here we are taking standard Cuffdiff outputs and evaluating expression changes only within the user defined linkage region. See the Cuffdiff page for more information on their formatted output files (http://cufflinks.cbcb.umd.edu/). Very briefly, Cuffdiff compares the transcripts between the inputted RNA-Seq datasets (here the mutant pool and wildtype pool) and finds differences in the expression profiles. The RNAeffector.R script then looks at these files and outputs several gene lists within the linked regions that include differences in annotated genes (gene_exp.diff_link.txt), in genes defined by expression found within the RNA-Seq datasets (cds_exp.diff_link.txt), in isoform usage (isoform_exp.diff_link.txt), or in transcriptional start site usage (tss_group_exp.diff_link.txt). The above files are all found in the _RNAeffectoR directory.

The most useful lists are:
gene_exp.diff_link.txt
cds_exp.diff_link.txt
These two have information on expression level differences for genes within the region. For example, in our manuscript we used a nonsense mutation in the *egr2b* gene known to undergo nonsense-mediated decay – in both of the above lists *egr2b* came out to be significantly downregulated within the region of linkage. Genes that are downregulated need not be due to nonsense changes – this method holds promise for identifying mutations that affect the expression level of a gene due to effects on enhancers, etc.
However, note that within this pipeline we have compared only one replicate of each condition and the data is noisy which could cause false positives, or negatives.

The isoform_exp.diff_link.txt has several limitations. First, exonically complex genes tend to always show up on the list – this is likely due to limited data from only one sample per condition. Second, we used a mutation in the *vangl2* locus that is known to have a splicing defect and include intronic sequence by creating a splice acceptor. The Cuffdiff package did not detect this alternative splicing event whereas visually assessment of the transcripts using IGV showed the known splicing defect (see paper). The reason for this problem is not clear and I have not been able to identify a way to reliably pull out such splicing changes.
Therefore, it is critical to visually evaluate your linked region, using IGV, for splicing defects. Using IGV I have identified several candidate mutations affecting splicing – these are currently being validated, but regardless of whether they are the

causative lesion, there is useful information to be had for the gene hunter. I recommend loading up your mutant alignment file, wildtype sibling, and separate wildtype sequence, then visually scanning through the entire linked region paying attention to the exon edges. I have found candidate alterations by comparing the wildtype "exon edges" to the mutant edges, with inappropriate splicing showing up either as inclusion of intronic sequence or removal of part or all of an exon.

The tss_group_exp.diff_link.txt seems to be flawed, as above, in that it includes many false positives likely due to limited data from only one sample per condition. It is included for the user to evaluate.

So you have some candidates… now what?

The method here great expedites the process of mapping and candidate identification. However, proof of causality is imperative. There are several things to do to moving forward.

1) Validate the changes of interest. Identify mutants and sequence the most interesting changes to ensure they are present in the mutant animals.
2) The linkage identified here is likely large. Further fine mapping will more closely link a region of the genome to the mutation – and this will further refine the list of candidates.
3) If there a really great candidate? Perhaps a STOP mutation in a gene that just makes a lot of sense for the process of interest then you should do things like:
    a) Rescue with wildtype RNA
    b) Morpholino and phenocopy
    c) TALEN and phenocopy

As an example, I used RNA-Seq mapping to identify a region of linkage for a mutation from a screen for synapse defects. A single STOP mutation was identified, a TALEN was created against the same gene, and animals carrying likely deleterious deletions were identified. These TALEN DEL carrying fish were crossed to the ENU induced STOP mutation and the phenotype of the DEL/ENU-STOP fish was the same as homozygous STOP animals. This is *very* strong evidence that gene in question is causal for the synaptic defect phenotype.