Differential expression with splice variant analysis aug2012

Differential expression with splice variant analysis at the same time: the Tuxedo pipeline

The **Tuxedo Pipeline** is a suite of tools for RNA-seq analysis, also known as the Tophat/Cufflinks workflow. It can be run in a variety of ways, optionally including *de novo* splice variant discovery. If an adequate set of splice variants is also available, it can also be run without splice variant detection to perform simple differential gene expression.

Resources

Useful RNA-seq resources are summarized on our Resources tool list, Transcriptome anaalysis section. The most important of these resources for Tuxedo are:

1. the original RNAseq analysis protocol using Tuxedo article in Nature Protocols, and
2. the URL for Tuxedo resource bundles for selected organisms (gff annotations, pre-built bowtie references, etc.)
3. the example data we'll use for this tutorial came from this experiment which has the raw fastq data in the SRA.

Objectives

In this lab, you will explore a fairly typical RNA-seq analysis workflow using the Tuxedo pipeline. Simulated RNA-seq data will be provided to you; the data contains 75 bp paired-end reads that have been generated in silico to replicate real gene count data from Drosophila. The data simulates two biological groups with three biological replicates per group (6 samples total). This simulated data has already been run through a basic RNA-seq analysis workflow. We will look at:

1. How the workflow was run and what steps are involved.
2. What genes and isoforms are significantly differentially expressed

Introduction

Overall Workflow Diagram

This **overview of tophat cufflinks workflow Diagram** outlines the Tuxedo pipeline. We have annotated the image from the original paper to include the important file types at each stage, and to note the steps skippin in the "fast path" (no *de novo* junction assembly).

This is the full workflow that includes *de novo* splice variant detection. For simple differential gene expression, Steps 2 (**cufflinks**) and 3-4 (**cuffmerge**) can be omitted.

Tuxedo data requirements

What is required for this pipeline?

- One or more datasets (best with at least two biological replicates and at least two conditions) in fastq format
- A reference genome, indexed for the Bowtie or Bowtie2 aligner (see this Tophat Resource Bundles page)
- Optionally, a set of known splice variants in the form of a GTF (gene transfer format) or GFF (gene feature format) file. These are also packaged as part of the resource bundles.

Paths through the Tuxedo workflow

There are three major paths through this workflow:

1. Simple differential gene expression analysis against a set of known splice variants.
    - A GTF/GFF file is provided, and you specify that *no novel junctions should be explored*
    - This is by far the fastest path through the workflow.
2. Same as 1), but novel splice junctions should be explored *in addition to* known splice junctions
    - A GTF/GFF file is provided, and you let the tool search for novel junctions also
3. Use the input data to construct *de novo* splice junctions without reference to any known splice junctions
    - No GTF/GFF is provided

What tophat does

The 1st, Tophat step is always required and sets the stage for all that follows. Tophat does a transcriptome-aware alignment of the input sequences to a reference genome using either the Bowtie or Bowtie2 aligner (in theory it can use other aligners, but we do not recommend this).

Split Read Alignment (Splice Finding)

To do this, Tophat goes through several steps:

1. The input sequences are aligned to the transcriptome for your reference genome, if you provided a GTF/GFF file. Remember, the transcriptome annotation describes known full-length unspliced transcripts.
    - sequences that align to the transcriptome are retained, and their coordinates are translated to genomic coordinates
    - sequences that do not align to the transcriptome are subjected to further analysis below
2. Remaining sequences are broken into sub-fragments of at least 25 bases, and these sub-fragments are aligned to the reference genome.
    - if two adjacent sub-fragments align to non-adjacent genomic locations, they are "trans frags" that will be used to infer splice junctions

At the end of the Tophat process, you have a BAM file describing the alignment of the input data to genomic coordinates. This file can be used as input for downstream applications like Cuffmerge, which will assemble parsimonious consensus fragments from the BAM file coordinates.

What cufflinks and cuffmerge do

Cuffmerge Assembly

For each separate dataset representing a specific replicate and condition, **cufflinks** assembles a map of genomic areas enriched in aligned reads. **cuffmerge** then takes the set of individual assemblies and merges them into a consensus assembly for all the provided datasets. The consensus may include known splice variant annotations if you have provided those to the program.

What cuffdiff and cummeRbund do

Next, **cuffdiff** uses the consensus splice variant annotations (and/or the known splice variants) to quantify expression levels of genes and isoforms, using FPKM (fragments per kilobase per million reads) metrics.

Finally, **cummeRbund** creates pretty differential expression plots of the FPKM data using R.

Notes on FASTQ preparation

Although we won't cover these issues here, there are some issues you should consider before embarking on the Tuxedo pipeline:

1. Should my FASTQ sequences be trimmed to remove low-quality 3' bases?

    Possibly, if FastQC or other base quality reports show the data is really poor. But generally the fact that Tophat splits long reads into smaller fragments mitigates the need to do this.
2. Should I remove adapter sequences before running Tophat?

    This is usually a good idea because un-template adapter bases have a more drastic effect on reducing mappability than do low-quality 3' bases.
3. Should I attempt to remove sequences that map to undesired RNAs before running Tophat? (rRNA for example)

    This is also usually a good idea, because such rRNA sequences can be a substantial proportion of your data (depending on library prep method), and this can skew **cuffdiff**'s fragment counting statistics.
4. How would, for example, rRNA sequence removal be done?

    Maybe something like this:

    - Align your sequences to a reference "genome" consisting only of rRNA gene sequences.
    - Extract only the sequences that do not align to the rRNA reference into a new FASTQ file and use that as Tophat input.
5. What other pre-processing steps might I consider?

    There are many, and it will depend on your data and what you want to get out of it.

    If you have paired-end data, **tophat** asks you to provide the mean fragment (insert) size and the standard deviation for insert sizes in your library. One common pre-processing step to achieve this would be to do a quick paired-end alignment of, for example, about 1 million sequences to a reference genome. Then you could calculate the mean and standard deviation of insert sizes for properly paired reads from the resulting BAM file records, and pass these values to Tophat.

Some Logistics...

Six raw data files were provided as the starting point:

- c1_r1, c1_r2, c1_r3 from the first biological condition
- c2_r1, c2_r2, and c2_r3 from the second biological condition

Due to the size of the data and length of run time, *most of the programs have already been run for this exercise*. The commands run are in different *. commands files. We will spend some time looking through these commands to understand them. You will then be parsing the output, finding answers, and visualizing results.

Data for this section is all located under **$BI/ngs_course/tophat_cufflinks/** So cd into this directory now.

Commands used are in different *.commands files located in **$BI/ngs_course/tophat_cufflinks/run_commands**

Some output, like bam files can be gotten from the URL http://loving.corral.tacc.utexas.edu/bioiteam/tophat_cufflinks/ and directly loaded into IGV, using **Load from URL**.

If you generate your own output and would like to view them on IGV, you will need to scp the files from lonestar to your computer.

On your computer's side:

Go to the directory where you want to copy files to.

```
scp my_user_name@lonestar.tacc.utexas.edu:/home/.../stuff.fastq ./
```

Replace the "/home/..." with the "pwd" information obtained earlier.

This command would transfer "stuff.fastq" from the specified directory on Lonestar to your current directory on your computer.

For help, remember to type the commands and hit enter to see what help they can offer you. You will also almost certainly need to consult the documentation for tophat, cufflinks and cummeRbund:

- http://tophat.cbcb.umd.edu/manual.html
- http://cufflinks.cbcb.umd.edu/manual.html
- http://compbio.mit.edu/cummeRbund/manual.html

Exercise Workflow

Here are the steps for the full workflow. Steps 2 and 3 can be omitted if you don't need to explore novel splice junctions. However here we will explore the full set of steps:

1. map reads against the Drosophila genome (**tophat**)
2. assemble the transcripts (**cufflinks**)
3. merge the assemblies (**cuffmerge**)
4. compute differential expression (**cuffdiff**)
5. inspect the results
6. examine the differential expression results (using Linux, IGV and **cummeRbund**)
7. (Extra) compare assembled transcripts to annotated transcripts to identify potentially novel ones (**cuffcmp**)

Items 1-4 have already been performed in advance to save time.

Step 1: Run tophat

On lonestar, to run tophat, cufflinks etc, following modules need to be loaded.

```
module load boost/1.45.0
module load bowtie
module load tophat
module load cufflinks/2.0.2
```

General syntax for tophat command

```
tophat [options] <bowtie_index_prefix> <reads1> <reads2>
```

Look at run_commands/tophat.commands to see how it was run.

cat $BI/ngs_course/tophat_cufflinks/run_commands/tophat.commands

Take a minute to look at the output files produced by one tophat run.

```
cd $BI/ngs_course/tophat_cufflinks/C1_R1_thout
ls -l

-rwxr-xr-x 1 daras G-803889 323M Aug 16 11:47 accepted_hits.bam
-r-xr-xr-x 1 daras G-803889 237K Aug 16 11:46 accepted_hits.bam.bai

-rwxr-xr-x 1 daras G-803889   52 Aug 16 11:46 deletions.bed
-rwxr-xr-x 1 daras G-803889   54 Aug 16 11:46 insertions.bed
-rwxr-xr-x 1 daras G-803889 2.9M Aug 16 11:46 junctions.bed
-rwxr-xr-x 1 daras G-803889   70 Aug 16 11:46 left_kept_reads.info
drwxr-xr-x 2 daras G-803889  32K Aug 16 11:46 logs
-rwxr-xr-x 1 daras G-803889   70 Aug 16 11:46 right_kept_reads.info
-rwxr-xr-x 1 daras G-803889 9.7K Aug 16 11:46 unmapped_left.fq.z
-rwxr-xr-x 1 daras G-803889 9.9K Aug 16 11:46 unmapped_right.fq.z
```

Exercise 1a: Providing a transcript annotation file

Which **tophat** option is used to provide a transcript annotation file (GTF file) to use?

Remember that you can type the command and hit enter to get help info about it.
Or Google **tophat** to find its online manual
The **-G <GTF filename>** to use the annotated splice junctions in the supplied GTF file

Exercise 1b: Using only annotated junctions
How would I tell **tophat** to *only* use a specified set of transcript annotation and *not* assemble any novel transcripts?

Specify **-G <gtf filename>** to have **tophat** use the annotated transcripts in the supplied GTF file
Also add the **--no-novel-juncs** option to suppress *de novo* junction detection

As you can see there are *many many* other options for running **tophat**!

Exercise 2: Examine a GTF file

The GTF file for our *Drosophila* genome (dm3) is in $BI/ngs_course/tophat_cufflinks/reference/genes.gtf. What does it look like?

Any one of these:

```
less $BI/ngs_course/tophat_cufflinks/reference/genes.gtf  # :q to exit
cat $BI/ngs_course/tophat_cufflinks/reference/genes.gtf | head
cat $BI/ngs_course/tophat_cufflinks/reference/genes.gtf | cut -f 1-8 | more
cat $BI/ngs_course/tophat_cufflinks/reference/genes.gtf | cut -f 9 | more
```

Exercise 3a: Examine a BAM file
Examine a few lines of the C1_R1 alignment file.

samtools view

```
cd $BI/ngs_course/tophat_cufflinks/C1_R1_thout
ls -la
samtools view -x accepted_hits.bam | less  # :q to quit
```

Exercise 3b: Spliced sequences
How is a spliced sequence represented in the BAM file?

The 6th BAM file field is the CIGAR string which tells you how your query sequence mapped to the reference.
Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

The CIGAR string "58M76N17M" representst a spliced sequence. The codes mean:

- **56M** - the first 58 bases match the reference
- **76N** - there are then 76 bases on the reference with no corresponding bases in the sequence (an intron)
- **17M** - the last 17 bases match the reference

Exercise 4: Count spliced sequences
How many spliced sequences are there in the C1_R1 alignment file?

samtools view and **cut** and **grep**

```
cd $BI/ngs_course/tophat_cufflinks/C1_R1_thout
samtools view accepted_hits.bam | cut -f 6 | grep 'N' | wc -l
```

Step 2: Run cufflinks

General syntax for cufflinks command

```
cufflinks [options] <hits.bam>
```

Look at $BI/ngs_course/tophat_cufflinks/run_commands/cufflinks.commands to see how it was run.

cat $BI/ngs_course/tophat_cufflinks/run_commands/cufflinks.commands

Take a minute to look at the output files produced by one cufflinks run.
The important file is transcripts.gtf, which contains Tophat's assembled junctions for C1_R1.

Cufflinks output files

```
cd $BI/ngs_course/tophat_cufflinks/C1_R1_clout
ls -l


drwxrwxr-x 2 nsabell G-801021    32768 May 22 15:10 cuffcmp
-rwxr-xr-x 1 daras G-803889  14M Aug 16 12:49 transcripts.gtf
-rwxr-xr-x 1 daras G-803889 597K Aug 16 12:49 genes.fpkm_tracking
-rwxr-xr-x 1 daras G-803889 960K Aug 16 12:49 isoforms.fpkm_tracking
-rwxr-xr-x 1 daras G-803889    0 Aug 16 12:33 skipped.gtf
```

Step 3: Merging assemblies using cuffmerge

Create a file listing the paths of all per-sample transcripts.gtf files so far, then pass that to **cuffmerge**:

```
cd $BI/ngs_course/tophat_cufflinks
find . -name transcripts.gtf > assembly_list.txt
cuffmerge <assembly_list.txt>
```

```
cat $BI/ngs_course/tophat_cufflinks/assembly_list.txt
```

Take a minute to look at the output files produced by **cuffmerge**. The most important file is merged.gif, which contains the consensus transcriptome annotations **cuffmerge** has calculated.

cuffmerge output

```
cd $BI/ngs_course/tophat_cufflinks/merged_asm
ls -l

-rwxrwxr-x  1 daras G-803889  1571816 Aug 16  2012 genes.fpkm_tracking
-rwxrwxr-x  1 daras G-803889  2281319 Aug 16  2012 isoforms.fpkm_tracking
drwxrwxr-x  2 daras G-803889    32768 Aug 16  2012 logs
-r-xrwxr-x  1 daras G-803889 32090408 Aug 16  2012 merged.gtf
-rwxrwxr-x  1 daras G-803889        0 Aug 16  2012 skipped.gtf
drwxrwxr-x  2 daras G-803889    32768 Aug 16  2012 tmp
-rwxrwxr-x  1 daras G-803889 34844830 Aug 16  2012 transcripts.gtf
```

Step 4: Finding differentially expressed genes and isoforms using cuffdiff

```
cuffdiff [options] <merged.gtf> <sample1_rep1.bam,sample1_rep2.bam> <sample2_rep1.bam,sample2_rep2.bam>
```

Exercise: What does **cuffdiff -b** do?

-b is for enabling fragment bias correction.

We will be inspecting the cuffdiff results further in a little bit.

Step 5: Inspect the mapped results

Before you even start, do a sanity check on your data by examining the bam files from the mapping output.

We've included the directory "bwa_genome" containing the output from bwa of the C1_R1_1 and C1_R1_2 files mapped directly to the genome using bwa. Tophat output is in C1_R1_thout for C1_R1 sample.

Report the total number of mapped reads for each sample using "samtools flagstat" for the bwa output (bam file) and the tophat output (bam file). There is a big difference between the bam file from tophat and the bam file from bwa. Can you spot it?

```
samtools flagstat C1_R1.bam
```

You might want to load the bwa bam file on IGV alongside the tophat bam file for the same sample to see the differences between mapping to the transcriptome and mapping to the genome.

Genome track

- Use D.melanogaster (dm3)

Data files (use Load from URL option)

- Load tophat bam file from URL: http://loving.corral.tacc.utexas.edu/bioiteam/tophat_cufflinks/C1_R1_accepted_hits.bam
- Load bwa bam file from URL: http://loving.corral.tacc.utexas.edu/bioiteam/tophat_cufflinks/bwa_genome/C1_R1.bam
- To load another tophat bam file from URL: http://loving.corral.tacc.utexas.edu/bioiteam/tophat_cufflinks/C2_R1_accepted_hits.bam

If you would like, you can also load another bam file output from the tophat run (remember, tophat calls bowtie for mapping). These are large file, so it may slow your computer down. Since we've already loaded C1_R1, let's load one from the C2 condition, say, C2_R1, to see genuine expression level changes.

Keep this open to come back to, when we further explore differential expression results.

Do this on your local machine, not on TACC

Downloading and running IGV

```
wget http://www.broadinstitute.org/igv/projects/downloads/IGV_2.1.14.zip
unzip IGV_2.1.14.zip
cd IGV_2.1.14
java -Xmx2g -jar igv.jar
```

Examine the differential expression analysis results

The **cuffdiff** output is in a directory called diff_out. We are going to spend some time parsing through this output. So, copy it over to your scratch directory and move to your SCRATCH directory.

```
cds
cp -r $BI/ngs_course/tophat_cufflinks/diff_out .
ls diff_out
```

Note that cuffdiff has performed a statistical test on the expression values between our two biological groups. It reports the FPKM expression levels for each group, the log2(group 1 FPKM/ group 2 FPKM), and a p-value measure of statistical confidence, among many other helpful data items.

Take a minute to look at the output files produced by cuffdiff.

cuffdiff output

```
cds
cd diff_out
ls -l

-rwxr-x--- 1 daras G-801020  2691192 Aug 21 12:20 isoform_exp.diff  : Differential expression testing for
transcripts
-rwxr-x--- 1 daras G-801020  1483520 Aug 21 12:20 gene_exp.diff     : Differential expression testing for genes
-rwxr-x--- 1 daras G-801020  1729831 Aug 21 12:20 tss_group_exp.diff: Differential expression testing for
primary transcripts
-rwxr-x--- 1 daras G-801020  1369451 Aug 21 12:20 cds_exp.diff      : Differential expression testing for
coding sequences

-rwxr-x--- 1 daras G-801020  3277177 Aug 21 12:20 isoforms.fpkm_tracking
-rwxr-x--- 1 daras G-801020  1628659 Aug 21 12:20 genes.fpkm_tracking
-rwxr-x--- 1 daras G-801020  1885773 Aug 21 12:20 tss_groups.fpkm_tracking
-rwxr-x--- 1 daras G-801020  1477492 Aug 21 12:20 cds.fpkm_tracking

-rwxr-x--- 1 daras G-801020  1349574 Aug 21 12:20 splicing.diff  : Differential splicing tests
-rwxr-x--- 1 daras G-801020  1158560 Aug 21 12:20 promoters.diff : Differential promoter usage
-rwxr-x--- 1 daras G-801020   919690 Aug 21 12:20 cds.diff       : Differential coding output.
```

Here is a basic command useful for parsing/sorting the `gene_exp.diff` or `isoform_exp.diff` files:

Linux one-liner for sorting cuffdiff output by log2 fold-change values

```
cat isoform_exp.diff | awk '{print $10 "\t" $4}' | sort -n -r | head
```

Exercise 1: Find the 10 most up-regulated genes, by fold change that are classified as significantly changed. Look at one example of a up-regulated gene, **regucalcin**, on IGV.

Top 10 upregulated genes
scf
CG8979
CG4389
crc
KdelR
Vha68-2
CG3835
Df31
by
Dhpr

One-line command to get 10 most up regulated genes

```
cat gene_exp.diff |grep 'yes'|sort -k10nr,10|head
```

One-line command to get 10 most down regulated genes

```
cat gene_exp.diff |grep 'yes'|sort -k10n,10|head
```

Exercise 2: Find the 10 most up-regulated isoforms, by fold change that are classified as significantly changed. What genes do they belong to?

simj
CG2177
sPLA2
Nipsnap
Pde8
by
CG15814
Dhpr
eIF-4E
spir

One-line command to get 10 most up-regulated isoforms

```
cat isoform_exp.diff |grep 'yes'|sort -k10nr,10|head
```

Step 6: Using cummeRbund to inspect differential expression data.

A) First load R and enter R environment

```
module load R
R
```

B) Within R environment, set up cummeRbund

```
source("http://bioconductor.org/biocLite.R")
biocLite("cummeRbund")
```

C) Load cummeRbund library and read in the differential expression results. If you save and exit the R environment and return, these commands must be executed again.

```
library(cummeRbund)
cuff_data <- readCufflinks('diff_out')
```

D) Use cummeRbund to visualize the differential expression results.

NOTE: Any graphic outputs will be automatically saved as "Rplots.pdf" which can create problems when you want to create multiple plots with different names in the same process. To save different plots with different names, preface any of the commands below with the command:

```
pdf(file="myPlotName.pdf")
```

And after executing the necessary commands, add the line:

```
dev.off()
```

Thus, to use the csScatter command and save the results in a file called scatterplot.pdf, one would execute the following commands:

```
pdf(file="scatterplot.pdf")

csScatter(genes(cuff_data), 'C1', 'C2')

dev.off()
```

To pull out significantly differentially expressed genes and isoforms

```
gene_diff_data  <- diffData(genes(cuff_data))
sig_gene_data  <- subset(gene_diff_data, (significant ==  'yes'))
nrow(sig_gene_data)
isoform_diff_data <-diffData(isoforms(cuff_data))
sig_isoform_data <- subset(isoform_diff_data, (significant == 'yes'))
nrow(sig_isoform_data)
```

To draw a scatterplot

```
csScatter(genes(cuff_data), 'C1', 'C2')
```

To plot gene level and isoform level expression for gene regucalcin

```
mygene1 <- getGene(cuff_data,'regucalcin')
expressionBarplot(mygene1)
expressionBarplot(isoforms(mygene1))
```

To plot gene level and isoform level expression for gene Rala

```
mygene2 <- getGene(cuff_data, 'Rala')
expressionBarplot(mygene2)
expressionBarplot(isoforms(mygene2))
```

Take cummeRbund for a spin...

CummeRbund is powerful package with many different functions. Above was an illustration of a few of them. Try any of the suggested exercises below to further explore the differential expression results with different cummeRbund functions.

If you would rather just look at the resulting graphs, they are at the URL: http://loving.corral.tacc.utexas.edu/bioiteam/tophat_cufflinks/ as exercise5_Rplots.pdf, exercise6_Rplots.pdf, and exercise7_Rplots.pdf.

You can refer to the cummeRbund manual for more help and remember that ?functionName will provide syntax information for different functions.
http://compbio.mit.edu/cummeRbund/manual.html

You may need to redo Step C) every time you reopen an R session.

Exercise 5: Visualize the distribution of fpkm values across the two different conditions using a boxplot.

R command to generate box plot of gene level fpkms

```
csBoxplot(genes(cuff_data))
```

R command to generate box plot of isoform level fpkms

```
csBoxplot(isoforms(cuff_data))
```

Use csBoxplot function on cuff_data object to generate a boxplot of gene or isoform level fpkms.

Exercise 6: Visualize the significant vs non-significant genes using a volcano plot.

csVolcano(genes(cuff_data), "C1", "C2")
Use csVolcano function on cuff_data object to generate a volcano plot.

Exercise 7: Pull out a subset of the genes using a ln_fold_change greater than 1.5. Generate expression bar plots for just those genes.

One possible solution

```
gene_diff_data  <- diffData(genes(cuff_data))
sig_gene_data  <- subset(gene_diff_data, (ln_fold_change > 1.5))
head(sig_gene_data)
sig_geneids <- c(sig_gene_data$gene_id)

myGenes <- getGenes(cuff_data, sig_geneids)
expressionBarplot(myGenes)
```

If you have trouble sourcing cummeRbund, try this:
module swap gcc intel
Reenter R and redo above steps.

Running the Tuxedo pipeline at TACC

One important consideration running the Tuxedo pipeline at TACC is the 24-hour clock time limit on running jobs. Several steps in the pipline, especially the initial **tophat** alignment step, can be very time consuming, especially under one or more of these conditions:

- The reference genome is large (e.g. human data)
- Your input dataset(s) are large and/or numerous (many replicates)
- Your data is paired end (each end is processed separately before pairing is attempted)
- *De novo* splice junction detection is requested

If the 24-hour clock time limit becomes a problem for your data, you can do one or more of the following to reduce the time taken by any one step:

- Give **tophat** and/or **cufflinks** only one dataset at a time
  - then use **cuffmerge** to merge the results, if you're using novel splice junctions detection
  - or just merge the resulting individual accepted_hits.bam files if novel splice junctions are not being explored (before **cuffdiff**)
- If even one dataset is too large, split it into pieces and run each separately then **cuffmerge** or BAM merge as above
  - Talk the TACC folks into allowing your jobs to run > 24 hours (good luck!)

Be aware that splitting up your data can reduce the power of the pipeline to find novel splice junctions, so you should try in general to process as much data as possible together.

Some tophat performance data

As we have seen above, most Tuxedo tools include a **-p** option for specifying the number of processes to use. When running at TACC, each process should be assigned to a separate core on a compute node, so you should coordinate your choice of **-p** with the "wayness" (number of tasks per node) option of launcher_creator.py.

This plot of **tophat** performance as a function of number of processes specified and options specified shows how these parameters interact when running Tophat on some basic RNAseq data.