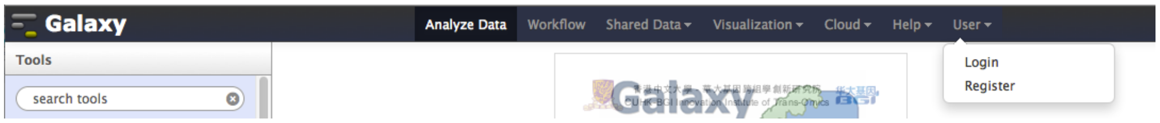
Using Galaxy for NGS Analyses

Luce Skrabanek

Registering for a Galaxy account

Before we begin, first create an account on the main public Galaxy portal.
Go to:
https://main.g2.bx.psu.edu/
Under the User tab at the top of the page, select the Register link and follow the instructions on that page.



This will only take a moment, and will allow all the work that you do to persist between sessions and allow you to name, save, share, and publish Galaxy histories, workflows, datasets and pages.

Disk quotas

As a registered user on the Galaxy main server, you are now entitled to store up to 250GB of data on this public server. The little green bar at the top right of your Galaxy page will always show you how much of your allocated resources you are currently using. If you exceed your disk quota, no further jobs will be run until you have (permanently) deleted some of your datasets.

Note: Many of the examples throughout this document have been taken from published Galaxy pages. Thanks to users james, jeremy and aun1 for making these pages and the associated datasets available.

Importing data into Galaxy

The right hand panel of the Galaxy window is called the Tools panel. Here we can access all the tools that are available in this build of Galaxy. Which tools are available will depend on how the administrator of the Galaxy instance you are using has set it up.

The first tools we will look at will be the built-in tools for importing data into Galaxy. There are multiple ways of importing data into Galaxy. Under the Get Data tab, you will see a list of 15+ tools that you can use to import data into your history.

Upload from database queries

You can upload data from a number of different databases. For example, you can retrieve data from different sections of the UCSC Genome Browser, using the UCSC Main, UCSC Archaea and BX Main tools.

Retrieving data from UCSC

The Galaxy tool will open up the Table Browser from UCSC in the Galaxy window. At this point, you can use the Table Browser exactly as you would normally. For example, let's say we want to obtain a BED-formatted dataset of all RefSeq genes from platypus.

1. Open the Get Data → UCSC Main tool.
2. Select platypus from the Genome pulldown menu.
3. Change the Track pulldown menu to RefSeq genes.
4. Keep all other options fixed (including region: genome and output format: BED and the 'Send output to Galaxy' box checked).
5. Click on the 'Get Output' button.
6. Select the desired format (here we will choose the default one BED record per whole gene).
7. Click the 'Send query to Galaxy' button.

Similarly, you can access BioMart through the BioMart and Gramene tools. In these cases, the Galaxy window will be replaced by the query forms for these databases, but once you have retrieved your results, you will get the option to upload them to Galaxy.

Other database query systems that open up within Galaxy include:
FlyMine, MouseMine, RatMine, YeastMine, EuPathDB, EpiGRAPH, GenomeSpace.

The others will open up in their own window, replacing the Galaxy window, but will have an option to export the results from your query to Galaxy:
WormBase, modENCODE fly, modENCODE modMine, modENCODE worm.

Upload data from a File

The Get Data → Upload File tool is probably the easiest way of getting data into a history. Using this tool, you can import data from a file on your computer, or from a website or FTP site. Here we will import a file from the website associated with this Galaxy workshop.

1. Open the Get Data → Upload File tool.
2. Enter the following URL into the text-entry box (either by typing it or right-click the link on the course webpage, select 'Copy Link' and paste the link into the text-entry box.)
   http://chagall.med.cornell.edu/galaxy/rnaseq/GM12878_rnaseq1.fastqsanger
3. Change the File-Format pulldown menu from 'Auto-detect' to 'Fastqsanger'. Although Galaxy is usually quite good at recognizing formats, it is always safer, if you know the format of your file, not to leave its recognition to chance. This is especially important with FastQ formats, as they all look similar but will give different results if Galaxy assumes the wrong format (see FastQ quality scores section.)
4. Click Execute.

Galaxy will usually have some kind of help text below the parameters for any tool. However, the help text is not always up-to-date or covers all the options available. This is the case here, where the help text details many of the formats listed in the File Format pulldown menu, but not all of them.

As mentioned, we chose the fastqsanger format here. The FastQ format is a format that includes not only the base sequence, but also the quality scores for each position. You will see five possible different FastQ formats in the pulldown menu [fastq, fastqcssanger, fastqillumina, fastqsanger, fastqsolexa]. The fastq format is the "default" FastQ format. If your dataset becomes tagged with this format, it will have to be converted into one of the other named formats before you can begin to do analysis with it. The fastqcssanger format is for color space sequences and the other three are FastQ formats that encode quality scores into ASCII with different conversions.

History

The right hand panel of the Galaxy window is called the History panel. Histories are the way that datasets are stored in an organized fashion within Galaxy.

1. To change the name of your history, click on the Unnamed history text. This will highlight the history name window and allow you to type the new name of the history.

Histories can also be associated with tags and annotations which can help to identify the purpose of a history. As the number of histories in your Galaxy account grows, these tags/annotations become more and more important to help you keep track of what the function of each history is.

Every dataset that you create gets a new window in this history panel. Clicking on the name of the dataset will open the window up to a slightly larger preview version, which shows the first few lines of that dataset.

There are many icons associated with each dataset.

1. The eye icon will show the contents of that dataset in the main Galaxy panel.
2. The pencil icon will allow you to edit any attributes associated with the dataset.
3. The X icon deletes the dataset from the history. Note that a dataset is not permanently deleted unless you choose to make it so.
4. The disk icon allows you to download the dataset to your computer.
5. The "i" icon gives you details about how you obtained that dataset (i.e., if you downloaded it from somewhere, or if it is the result of running a job within the Galaxy framework.)
6. The scroll icon allows you to associate tags with the dataset. Similarly to the tags for the history itself, these tags can help to quickly identify particular datasets.
7. The post-it-note icon allows you associate annotations with a dataset. These annotations are also accessible via the Edit Attributes pencil icon.

All datasets belonging to a certain analysis are grouped together within one history. Histories are sharable, meaning that you can allow other users to access a specific history (including all the datasets in it). To make a history accessible to other users:

1. Click the gear icon at the top of the history panel that you want to share.
2. Choose the Share or Publish option.
    a. To make the history accessible via a specific URL, click the Make History Accessible via Link button.
    b. To publish the history on Galaxy's Published Histories section, click the Make History Accessible and Publish option. To see the other histories that have been made accessible at this site, click the Shared Data tab at the top of the page and select Published Histories.

To go back to your current Galaxy history, click the Analyze Data tab at the top of the page. To see a list of all of your current histories, click the gear icon at the top of the history panel and select Saved Histories.

All initial datasets used in this class, as well as prepared mapping runs, are available via two shared histories.
https://main.g2.bx.psu.edu/u/luce/h/workshopdatasets
https://main.g2.bx.psu.edu/u/luce/h/mappingresults

You can import these histories into your own account, using the green "+" icon at the top of the page. You can now either directly start using these histories, or copy any dataset from these histories into any other history you happen to be working with, by using the Copy Datasets command, accessible from the gear icon in the History panel.

Changing dataset formats, editing attributes

If Galaxy does not detect the format of your dataset correctly, you can always change the format of your dataset manually by editing the attributes of the dataset (the pen icon to the upper right of the dataset link in your history.) Specifically, there is a “Change data type” section, where you can select from a pulldown menu the correct format that you want associated with your dataset.

If not already included in the details section of a dataset, it can also be helpful to include other notes about a) where you got the data, b) when you got the data (although the dataset should have a Created attribute, visible from the “view details” icon, the “i”), c) if importing from a database, what query you used to select the data.

Exercises

Update the attributes of the GM12878 dataset that we previously downloaded.

1. Click the pencil icon for the GM12878 dataset.
2. Change the name of the dataset to GM12878 in the Name field.
3. Associate the dataset that we just downloaded with the human hg19 genome in the Database/Build field.
4. Add to the Notes field that we downloaded it from a website. Include the website URL.
5. Click Save.

Examining and Manipulating FastQ data

Quality Scores

The FastQ format provides a simple extension to the FastA format, and stores a simple numeric quality score with each base position. Despite being a "standard" format, FastQ has a number of variants, deriving from different ways of calculating the probability that a base has been called in error, to different ways of encoding that probability in ASCII, using one character per base position.

PHRED scores

Quality scores were originally derived from the PHRED program which was used to read DNA sequence trace files, and linked various sequencing metrics such as peak resolution and shape to known sequence accuracy. The PHRED program assigned quality scores to each base, according to the following formula:

$$Q\_PHRED = -10\ log10\ (Pe)$$

where Pe is the probability of erroneously calling a base. PHRED put all of these quality scores into another file called QUAL (which has a header line as in a FastA file, followed by whitespace-separated integers. The lower the integer, the higher the probability that the base has been called incorrectly.

PHRED Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90 %
20	1 in 100	99 %
30	1 in 1000	99.9 %
40	1 in 10000	99.99 %
50	1 in 100000	99.999 %

While scores of higher than 50 in raw reads are rare, with post-processing (such as read mapping or assembly), scores of as high as 90 are possible.

Quality scores for NGS data are generated in a similar way. Parameters relevant to a particular sequencing chemistry are analyzed for a large empirical data set of known accuracy. The resulting quality score lookup tables are then used to calculate a quality score for *de novo* next-generation sequencing data.

Solexa scores

The Solexa quality scores, which were used in the earlier Illumina pipelines, are calculated differently from the PHRED scores:

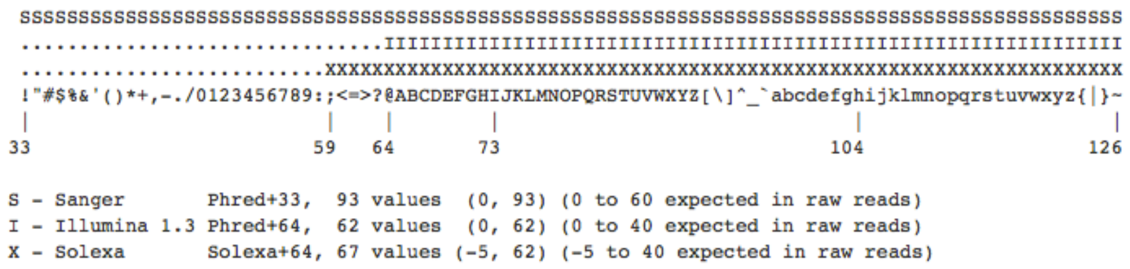
$$Q\_SOLEXA = -10\ log10\ (\frac{Pe}{1-Pe})$$

FastQ Conversion

Changing between FastQ formats

Galaxy is able to interchange between the different FastQ formats with a tool called FASTQ Groomer, found under the NGS: QC and manipulation tab. Since Galaxy tools are designed to work with the Sanger FastQ format, it is advisable to convert any FastQ datasets in another FastQ format to Sanger FastQ. The FASTQ Groomer tool takes as input any dataset designated as FastQ format and converts it according to the equations found in Cock et al. NAR 2009.

Description Galaxy format name	ASCII characters		Quality score	
	Range	Offset	Type	Range
Sanger standard/Illumina 1.7+ fastqsanger	33 to 126	33	PHRED	0 to 93
Solexa/early Illumina fastqsolexa	59 to 126	64	Solexa	-5 to 62
Illumina 1.3+ fastqillumina	64 to 126	64	PHRED	0 to 62



```
 SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
 ...............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
 ..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
 |                         |    |        |                              |                     |
33                        59   64       73                            104                   126

S - Sanger        Phred+33,  93 values  (0, 93) (0 to 60 expected in raw reads)
I - Illumina 1.3 Phred+64,  62 values  (0, 62) (0 to 40 expected in raw reads)
X - Solexa        Solexa+64, 67 values (-5, 62) (-5 to 40 expected in raw reads)
```

FastQ Quality Control

There are two standard ways of examining FastQ data in Galaxy: using the Summary Statistics method and the FastQC method.

Summarizing and visualizing statistics about FastQ reads

A very important tool that Galaxy provides for FastQ dataset is the NGS: QC and manipulation → FASTQ Summary Statistics tool. For every column in a set of sequences, this tool will calculate the minimum, maximum, mean, median and first and third quartile quality scores, as well as an overall count of each type of base found for that column. This tool is especially useful for determining at which base sequences should be trimmed so that only high quality sequence is used in your NGS analysis.

The output from the Summary Statistics tool is designed to be used as input to the Graph/Display Data → Boxplot tool. This tool creates a boxplot graph from tabular data. For our purposes, its main function is to visualize the statistics from the Summary Statistics tool. Much of the output from the summary statistics tool is not used by the boxplot tool, since to draw a boxplot, you only need to specify the median, first and third quartiles, whiskers and outliers (if any). The output will be a PNG image viewed in GnuPlot.

Exercise

Run a quality control on the GM12878 dataset that we previously downloaded.

1. Open the NGS: QC and manipulation → FASTQ Summary Statistics tool. Make sure the GM12878 dataset is selected.
2. Click Execute.
3. Open the Graph/Display Data → Boxplot tool. Make sure the input dataset is the output from the Summary Statistics tool in the last step.
4. Change the X-axis label to "read position" and the Y-axis label to "quality score".
5. Click Execute.

FastQC quality control

Another way of looking at your data to determine the overall quality of your run and to warn you of any potential problems before beginning your analysis is to use the NGS: QC and manipulation → FastQC tool, from Babraham Bioinformatics. It takes as input either a FastQ dataset, or BAM or SAM datasets. FastQC bundles into one executable what many of the individual tools available in Galaxy do for specific FastQ formats.

Basic Statistics

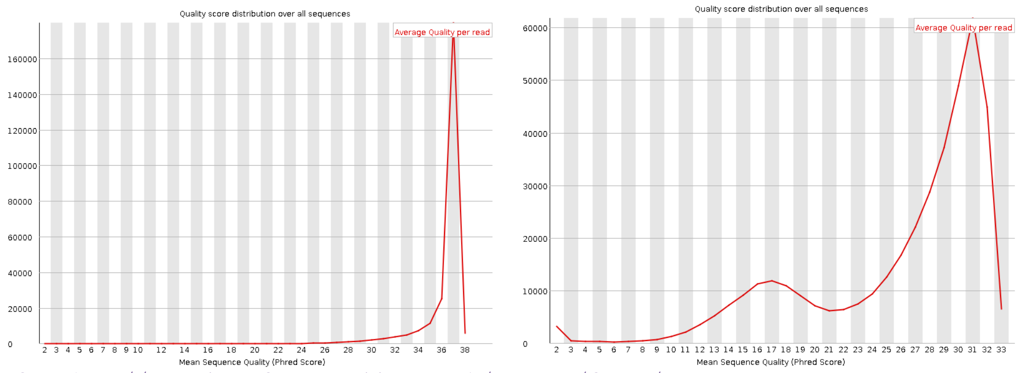
This section gives some simple composition statistics for the input dataset, including filename, filetype (base calls or colorspace), encoding (which FastQ format), total number of sequences, sequence length and %GC.

Per Base Sequence Quality

This plot shows the range of quality values over all bases at each position (similar to the boxplot from the BoxPlot tool.)

Per Sequence Quality Scores

This plot allows you to see if there is a subset of your sequences which has universally low scores (perhaps due to poor imaging). These should represent only a small number of the total sequences. See for comparison the "good" dataset and the "bad" dataset, below. All of the reads in the "good" dataset have a mean quality score of 37; the "bad" dataset has 10,000+ reads with a mean quality score of around 17.



(from http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/)

Per Base Sequence Content

This plot shows the proportion of each base at each position in the read. In a random library, you would expect that all frequencies are approximately equal at all positions, over all sequences. If you see strong biases which change for different bases then this usually indicates an overrepresented sequence which is contaminating your library. A bias which is consistent across all bases either indicates that the original library was sequence biased, or that there was a systematic problem during the sequencing of the library.

Per Base GC Content

This plots shows the GC content of each position in the run. In a random library, there should be minimal difference between positions, and the overall GC content should reflect the GC content of the genome under study. As in the Per Base Sequence Content plot, deviations across all positions could indicate an over-represented sequence. Similarly, if a bias is consistent across all bases, this could either indicate that the original library was sequence biased, or that there was a systematic problem during the sequencing of the library.

Per Sequence GC content
This plots the GC content across each sequence and compares it to a modeled GC content plot. In a random library, the plot should look normal and the peak should correspond to the overall GC of the genome under study. A non-normal distribution may indicate a contaminated library or biased subset.

Per Base N Content
This plots the percentage of base calls that were Ns (i.e., a base call could not be made with certainty) at every position. Ns more commonly appear towards the end of a run.

Sequence Length Distribution
This plots the distribution of all read lengths found.

Duplicate Sequences
This counts the number of times any sequence appears in the dataset and shows a plot of the relative number of sequences with different degrees of duplication. In a diverse library most sequences will occur only once in the final set. A low level of duplication may indicate a very high level of coverage of the target sequence, but a high level of duplication is more likely to indicate some kind of enrichment bias (e.g., PCR over-amplification).

Overrepresented Sequences
This creates a list of all the sequences which make up more than 0.1% of the total. A normal high-throughput library will contain a diverse set of sequences. Finding that a single sequence is very over-represented in the set either means that it is highly biologically significant, or indicates that the library is contaminated, or not as diverse as you expected.

Overrepresented Kmers
This counts the enrichment of every 5-mer within the sequence library. It calculates an expected level at which this k-mer should have been seen based on the base content of the library as a whole and then uses the actual count to calculate an observed/expected ratio for that k-mer. In addition to reporting a list of hits it will draw a graph for the top 6 hits to show the pattern of enrichment of that k-mer across the length of your reads. This will show if you have a general enrichment, or if there is a pattern of bias at different points over your read length.

Exercise
Run a quality control on the GM12878 dataset using the FastQC tool.

1. Open the NGS: QC and manipulation → FastQC tool.
2. Select the GM12878 dataset.
3. Click Execute.

FastQ Manipulation

Trimming the ends off reads

In general, we hope that most of the bases at the start of a run are of predominantly high quality. If, as the run progresses, the overall quality of the reads decreases, it may be wise to trim the reads before engaging in downstream analysis.

NGS: QC and manipulation → FastQ Trimmer [under Generic FASTQ manipulation]

This tool trims 3' and 5' ends from each read in a dataset. This is especially useful with Illumina data which can be of poorer quality towards the 3' end of a set of reads. For fixed-length reads such as Illumina and SOLiD data, the base offsets should be defined by the absolute number of bases that you want to take off either end, whereas for variable length reads like 454, the number of bases to be trimmed off the end is defined by the percentage of the entire length. Foe example, to take the last 20 bases off the end of each read, the offset from the 3' end is changed to 20.

Removing individual sequences

It is possible that you may want to get rid of some reads which contain one or more bases of low quality within the read. This is done using the Filter FastQ tool

NGS: QC and manipulation → Filter FastQ [under Generic FASTQ manipulation]

This tool allows the user to filter out reads that have some number of low quality bases and return only those reads of the highest quality. For example, if we wanted to remove from our dataset all reads where the quality of any base was less than 20, we change the Minimum quality box to 20. The "Maximum number of bases allowed outside of quality range" allows us to select how many bases per read must be below our threshold before we discard that read. If we leave this at the default setting of 0, all bases in a read must pass the threshold minimum quality score to be kept.

More complex manipulations

The Manipulate FastQ tool in Galaxy also allows much more complex manipulation of FastQ data, whether manipulating the read names, the sequence content or the quality score content. This tool also allows the removal of reads that match some given criteria.

NGS: QC and manipulation → Manipulate FASTQ [under Generic FASTQ manipulation]

This tool can work on all, or a subset of, reads in a dataset. The subset is selected using regular expressions on either the read name, sequence content, or quality score content. If no 'Match Reads' filter is added, the manipulation is performed on all reads. One of the more common manipulations used from this suite is the DNA to RNA manipulation on sequence content, which will have the effect of replacing all Ts in a read with Us.

Exercises

The GM12878 dataset seems to be of poor quality.

1. Trim the reads in the GM12878 dataset using the Generic FASTQ manipulation → FastQ Trimmer tool. Determine from the boxplot and FastQC figures where the quality of the reads begins to drop off sharply. Calculate how many bases have to be trimmed from the end and use that number as the Offset from 3' end.
2. Using the Generic FASTQ manipulation → Filter FastQ tool, filter out all sequences with any bases that have a quality less than 20. How many sequences do you have left in your dataset?
3. Run another QC (summary statistics and boxplot) on both of the new datasets from steps 1 and 2.
4. Modify the trimmed dataset from step 1 so that all Thymines are replaced by Uracils.

Operating on tabular/interval data

One of the advantages of Galaxy is that it makes available many database operations quite transparently. In a database setting, we would use those operations for "joining" tables on a common field, but another common usage is to join information from tables by finding overlapping genomic segments. In Galaxy, tables that include genomic intervals are said to be in interval format.

Joining two tables with interval data

In this exercise we will upload some TAF1 sites that have been detected using a ChIP-Seq experiment, and try to identify the genes that are associated with these sites.

1. Create a new history by clicking the gear icon and selecting Create New.
2. Name your new history to Simple database operations.
3. Upload the file from the following URL by using the Get Data → Upload File tool and entering http://galaxy.psu.edu/CPMB/TAF1_ChIP.txt into the URL/text entry box.
4. Click Execute.
5. Once the dataset has been downloaded, we can take a look at it. It is a tabular dataset where each line represents a location of a predicted TAF1 binding site in the genome. For this information to be meaningful, we need to tell Galaxy which genome build these genomic locations are referring to. In this case, this dataset was generated using the hg18 dataset.
    a. Change the attributes of the dataset by clicking the pencil icon.
    b. Move the URL from the Name field to the Annotation/Notes field.
    c. Change the name of the dataset to something manageable, like 'TAF1 Sites'.
    d. Change the Database/Build field to hg18 (rather than trying to scroll through the list in the pulldown menu, click in the pulldown menu box and start typing hg18; the list of options in the pulldown menu matching this keyword are quickly narrowed).
    e. Change the data type of the dataset (from the Datatype tab) from tabular to interval to tell Galaxy that this is a specific type of tabular data that contains genomic intervals.
6. Click on Save. Genomic interval data is defined by having a chromosome location, and start and stop positions. Galaxy will attempt to figure out which columns correspond to these data. In this case, the chromosome column is 2, the start column is 3, the end column is 4, and the name is located in column 5.

We now need to get the gene annotations from UCSC.

1. Open the Get Data → UCSC Main tool.
2. Change the assembly to March 2006 (hg18) in the assembly pull down menu.
3. Select RefSeq genes as the track to be queried.
4. Make sure the output format is set to BED and the Send Output to Galaxy box is checked.
5. Click 'get output'.
6. On the following page, make sure that you are downloading one BED record per whole gene.
7. Click the Send query to Galaxy button.
8. Use the Edit attributes function to rename the dataset to 'RefSeq genes', once this dataset of RefSeq genes has finished downloading.

Next we want to get a set of putative promoter sequences. Here, we are going to use the upstream sequence as a simple definition of a promoter sequence. (Note that, although in this example we are using a Galaxy tool to accomplish this, we could also have done this through the UCSC Main interface.)

1. Open the Operate on Genomic Intervals → Get Flanks tool to extract the upstream region of the RefSeq genes.
2. Change the length of the flanking region to 1000.
3. Click Execute.
4. Change the name of the resultant dataset to RefSeq promoters.

Now we will use the database join operation to join the original TAF1 binding site dataset with the promoter regions of the RefSeq genes.

1. Open the Operate on Genomic Intervals → Join tool. Note that this tool can only be used with interval datasets.
2. Select the promoters dataset as the first dataset and the TAF1 sites dataset as the second dataset to be joined.
3. Click Execute.

This operation returns a dataset of all those records where the genomic interval overlaps by at least one base, one pair per line. The UCSC "promoter" dataset contains a lot of extra exon information, so you will have to scroll all the way to the right to see the information about the TAF1 site that is being associated with each gene promoter region.
We can tidy this table up by selecting only a handful of columns to display in our final dataset.

1. Open the Text Manipulation → Cut tool.
2. Select c1,c2,c3,c4,c6,c15,c16,c17 as the columns to be cut. This will select only the columns with the chromosome information, start and stop positions of the promoter, the gene name, the strand the gene is coded on, the start and stop positions of the TAF1 site and the TAF1 site name.

Performing calculations on the result of interval data joining

The next example is slightly more complicated. Let's say we want to list all the exons on chromosome 22, sorted by the number of single nucleotide polymorphisms they contain. What genes do the five exons with the highest number of SNPs come from?

First, create a new history. Call it "SNPs in coding exons".

Next retrieve all coding exons from UCSC.

1. Use Get Data → UCSC Main tool.
2. Change the position to search to chr22.
3. Click the get Output button.
4. On the output format screen, change it to one BED record per coding exon.
5. Click Send query to Galaxy.
6. Rename this exon dataset to "Exons_22".

Now retrieve the SNP data, again from UCSC.

1. Use Get Data → UCSC Main tool.
2. This time, change the group to Variation and Repeats.
3. Change the position to chr22 again.
4. On the output format page, the one BED record per gene should again be selected. In this case, "gene" is actually "SNP" (or "feature").
5. Click the Send query to Galaxy button.
6. Rename this dataset to "SNPs_22".

To find the exon with the highest number of SNPs, we must first associate the SNPs with their exons. We do this using the database join command, which will associate with an exon any SNP whose given genomic interval overlaps that of any exon by at least one base (in this case, each SNP location is only one base in length).

1. Open the Operate on Genomics Intervals → Join tool.
2. Use the exons dataset as the first dataset and the SNPs dataset as the second dataset.
3. Click Execute.

Each line in this new dataset is the "join" of an exon and a SNP that is found within that exon (more exactly, whose genomic region overlaps that of the corresponding exon). The first six columns are those associated with the exon, the next six are those associated with the SNP.

Next, we want to count the number of SNPs associated with each exon.

1. Open the Join, Subtract, and Group → Group tool.
2. Change the Group by Column to c4 (i.e., the exon name).
3. Add a new operation.
4. Change the type of the new operation to Count and the column to c4. This groups the dataset on the exon name, and count how many exons of the same name are in each unique group.

The resulting dataset will have two columns: one is a list of all the distinct exon names, and the other is the number of times that exon name was associated with a SNP. We can now rearrange this dataset such that the exons with the highest number of SNPs are at the top.

1. Open the Filter and Sort → Sort tool.
2. Sort the dataset by column 2 (SNP count), in descending order (i.e., highest first).

We can see that the first exon (cds_0) of gene uc003bhh.3 has the highest number of SNPs with 26.

Finally, we want to select the five exons with the most SNPs.

1. Open the Text Manipulation → Select First tool.
2. Make sure the sorted dataset is selected.
3. Change the number of lines to select to 5.
4. Click Execute.

This dataset will simply be the first 5 lines of the previous dataset.

If we want to view these five exons in a genome browser, we will have to get their genomic coordinates, which we lost when we grouped the data. However, we still have this information in our original dataset, and we can re-associate this information with our exons of interest.

1. Open the Join, Subtract and Group → Compare two Datasets tool.
2. Choose c4 from our Exons_22 dataset and c1 from our final set of five exons (where c4 and c1 are the exon name column in each respective dataset).
3. Click Execute.

This extracts from our Exons_22 dataset any lines which match the exon name from our final dataset.
To visualize these regions, we can click on any of the visualization tools listed for that dataset (UCSC, GeneTrack, Ensembl or RViewer).

Converting to interval format

Not all datasets will be in interval format, and will have to be converted before they can be used in analyses like the above.
Download the variations from the ENSEMBL database for chromosome 22. We can access it through the BioMart portal in Galaxy.

1. Open the Get Data → BioMart tool. This will take us away from the Galaxy page and to the query interface to the BioMart databases.
2. Choose the Ensembl Variation 69 → Homo sapiens Somatic Variation (GRCh37.p8) database.
3. Click the Filters link. This is the section where you restrict the entries that will be retrieved.
    a. Open the Region section.
    b. In the chromosome pulldown menu, choose 22. The chromosome checkbox should be automatically checked as soon as you choose a chromosome.
4. Click the Attributes link. This is the section where you determine the fields that will be shown in the output.
    a. Check the Variation ID, Chromosome name and Position on Chromosome (bp) checkboxes, if they are not already checked.
5. Click the Results tab. The first 10 results will be shown.
6. In the Export results section, make sure that Galaxy and TSV are selected and click the Go button.

You will be redirected back to Galaxy and the dataset will be imported to your current history.
If for some reason the results are shown in HTML format, they will not be imported correctly into Galaxy. If this happens, reload the page and reselect the Results tab.

The variation data that we imported from BioMart is in tabular format, with the position of the variation being noted by a single column. The interval data format expects a start and an end position for every feature. To convert the tabular data to interval format:

1. Open the Text Manipulation → Compute tool.
2. Add the expression c3+1 as a new column to the current dataset, where c3 is our original position column.
3. Set Round Result to Yes to ensure that the new position column is an integer.
4. Click Execute.

This will add a new final column to the dataset. We now have to indicate to Galaxy that this dataset is in interval format, which we can do by clicking the pencil icon and editing the attributes.

1. Change the data type to interval.
2. Click Save.
3. Change the corresponding columns to name, chromosome, start and end. In this case, the name is in column 1, the chromosome is in column 2, the start in 3 and our newly computed end position is in column 4.
4. Click Save.

In the event that we had not included the chromosome field when importing from BioMart, we could add a chromosome column using the Text Manipulation → Add Column tool to add a new column, fully populated with a single number, the chromosome of interest.

Workflows

Workflows are great a great method to redo analyses automatically and simply on many datasets. They are also a good way to ensure reproducibility, especially if you share the workflows with your colleagues.

Extract Genomic Sequence Workflow

In this exercise, we are going to extract the genomic sequence flanking all the SNPs in the rhodopsin gene and then learn how to package those steps into a reusable workflow.

We will upload the data using the database query method.

1. Create a new history.
2. Open the Get Data → UCSC Main tool, which will bring up the query interface for the UCSC Genome Browser.
3. Making sure that the Mammal, Human and Feb 2009 options are selected, choose the Variation and Repeats dataset, which should change the track to the most recent SNP dataset (135).
4. Change the region radio button to position and enter uc003emt.3 in the textbox. Click the lookup button to automatically convert this to the assocaiated genomic region (uc003emt.3 is rhodopsin). Make sure that the position radio button is still checked.
5. Make sure the output format is set at BED and that the Send to Galaxy checkbox is checked.
6. Click the Get Output button. This takes you to a page which the information to be contained in the output. We want one BED record per gene which, in this case, is actually one record per SNP.
7. Click the Send query to Galaxy button.

Once the data has been downloaded, the history item will turn green. We can now edit the attributes for this dataset by clicking the pencil icon. Because this dataset is in BED format, it assumes that these are genomic regions, and has labeled the columns accordingly. In general, it is a good idea to make some notes in the Info field about where you got the information from (although some of this should automatically have been filled in the Database/Build field). We can also change the name of the dataset to something simpler, such as 'SNP locations'. Once you have entered your changes, click Save.

Now we want to get 50 bases of flanking sequence around each of these SNPs.

1. Select the Operate on Genomic Intervals → Get flanks tool.
2. Change the Location of the flanking regions to both (since we want flanking sequence on both sides of the SNP), and leave all other values as default.
3. Click Execute.
4. Change the annotations for this new dataset, by clicking the pencil button associated with this operation in the history. Change the name to 'SNP flank regions'.

This dataset looks very similar to our initial dataset, but has twice the number of lines: one for each 50 base flank on either side of every SNP.

Now we get the DNA sequence of each of these flanking regions.

1. Open the Fetch Sequences → Extract Genomic DNA tool.
2. Making sure the second dataset is selected, change the Output data type to interval, which returns a tab delimited dataset with the sequence, as well as chromosome, start and stop positions and associated SNPs.
3. Change the name of this dataset to 'Flanking Genomic Sequence' by editing the attributes with the pencil icon.

To make this series of steps into a reusable workflow, click the gear icon at the top of the history window and choose Extract Workflow. There should be three steps; ensure that all are checked for inclusion in the workflow. Rename the workflow to 'Retrieve flanking SNP sequence' and click Create Workflow. This workflow has now been added to the list of your workflows at the bottom of the leftmost column.

To use this workflow with another gene of interest, let us get some SNP data for another gene, say p53.

1. Create a new history by clicking on the gear icon and selecting Create New.
2. Go to the Get Data → UCSC Main tool.
3. Change the group to Variation and Repeats, input uc010vug.2 (p53) into the text entry box, and click lookup.
4. Making sure that the output is set at BED and is being sent to Galaxy, click the Get Output button.
5. On the following page, click the Send query to Galaxy button.
6. Once the dataset has been downloaded, change the name in the attributes panel by clicking on the pencil icon.

To run the workflow with this new dataset, choose the Workflow tab from the top of the screen. Locate your newly created workflow, and choose Run from the pulldown menu. The steps in the workflow are now listed. Step 1 requires you to input a dataset. Choose the p53 SNPs dataset that we just downloaded from UCSC, and click Run Workflow. The workflow will automatically run both the step that defined the flanking regions for each SNP, as well as the retrieval of the genomic sequence. Once the workflow is complete, you can click the eye icon of the final dataset in your history to view the interval-formatted genomic sequence for the flanking sequence for the 46 SNPs found in p53 (i.e., 92 flanking regions).

Filtering Exons by Feature Count Workflow

For an even more complicated workflow, let's use the "SNPs in coding exons" example from the previous section. Go to the coding exons history by finding it in your Saved Histories list. Convert that history to a workflow, by choosing Extract Workflow from the list of options accessible by clicking the gear icon. Rename the workflow to Coding Exon SNPs and click Create Workflow. Your workflow is now accessible from the Workflows tab at the top of the page.

We can now edit this workflow by clicking on its name and choosing Edit from the pulldown menu. Clicking on any of the boxes in this figure will bring up the list of options associated with that tool. There are a number of ways to use and edit workflows, but one of the most useful is the ability to hide certain intermediate steps so that when you re-use the workflow, you don't end up with multiple intermediate datasets shown in your history when all you really want is the end result. If we want to hide all intermediate steps except the last one, click the asterisk in the lower right corner of the box for the final step. Note that as soon as we do this, it turns darker orange in the overview window. We can also ensure that any datasets are named in a sensible manner within the editor. For example, we can rename each input dataset to indicate what kind of data that dataset should contain. In this case, we can rename one input dataset as Exons, and the other as SNPs, by simply clicking on the box representing that dataset and changing the name in the right hand panel. We can also rename the final dataset by clicking on its representation and choosing Rename Dataset in the Edit Step Actions section, and clicking the Create button. This opens a small text entry box where we can enter what we would like the dataset resulting from this workflow to be called. Call it Top 5 Exons. Within this editor, you can also change the value of any parameter in a tool. Once our changes are complete, choose Save from the menu (the cog) at the top right of the editor.

We can now run this new workflow.

1. Create a new history.
    a. Click the Analyze Data tab.
    b. Choose Create New from the gear icon.
2. Before we run the workflow, we need to download some data.
3. Retrieve the coding exons from chromosome 21 (remember to choose one BED record per coding exon) from the Get Data → UCSC Main tool.
4. Retrieve the common SNPs for chromosome 21.
5. Once the two datasets are downloaded, rename them to shorter names.
6. Start the workflow.
    a. Go to the Workflow menu.
    b. Select the Coding Exon SNPs workflow.
    c. Choose Run.
7. Choose the Exons dataset as the first dataset, the SNPs dataset as the second.
8. Click Run Workflow.

While waiting to be run, each dataset will be shown in the history, whether we marked it to be hidden or not. Once an operation is finished, if it was marked to be hidden, it will disappear from the history pane. When the workflow has finished running, we will be left with only the dataset that we did not hide, i.e., the list of the five exons with the highest number of SNPs.

Mapping Illumina data with BWA

Next Generation Sequencing

Next gen sequencing experiments result in millions of relatively small reads that must be mapped to a reference genome. Since using the standard alignment algorithms is unfeasible for such large numbers of reads, a lot of effort has been put into developing methods which are fast and are relatively memory-efficient.

Mappers

There are two mappers available in Galaxy: Bowtie and BWA. The crucial difference between these mappers is that BWA performs gapped alignments, whereas Bowtie does not (although there is a version of Bowtie available which does perform gapped alignments, it is not the one available in Galaxy). This, therefore, gives BWA greater power to detect indels and SNPs. Bowtie tends to be much faster, and have a smaller memory footprint, than BWA. BWA is generally used for DNA projects, whereas Bowtie is used for RNA-Seq projects since the exonic aligner TopHat uses Bowtie to do the initial mapping of reads.

Aligning reads with BWA [Burrows-Wheeler Alignment]

Create a new history by clicking the gear icon and selecting Create New. We will be working with the Blood-PCR1 dataset from the mtProjectDemo library. This is a dataset derived from the blood of a single individual, which has been enriched for mitochondrial sequence using PCR. Each eukaryotic cell contains many hundreds of mitochondria with many copies of mtDNA. Heteroplasmy is the (not uncommon) presence of multiple mtDNA variants within a single individual. We are going to search this dataset for heteroplasmic sites. This will use a similar methodology as if we were looking for SNPs, although the frequency of heteroplasmic sites will be much lower than would be seen for SNPs.

1. Click the Shared Data tab.
2. Choosing Data Libraries
3. Clicking on the mtProjectDemo link.
4. Check the box beside Blood-PCR1.
5. Make sure the Import to current history option is selected.
6. Click Go.

Return to your Galaxy analysis page by clicking the Analyze Data tab. As usual, with a new dataset:

1. Do a quality control check using the NGS: QC and manipulation → FASTQ Summary Statistics tool.
2. Click Execute.
3. Visualize these data using the Graph/Display Data → Boxplot tool.
4. Inspect the resulting graph by clicking on the eye icon associated with the result of this tool. We can see that the read length is 76 bases, and the quality ranges from 35 at the 5' end to 18 at the 3' end.

Since we will be using these reads to look for heteroplasmic sites, we will map these reads to the human genome using BWA. This action will take some time to complete, as we are mapping 500,000 reads to the complete human genome.

1. Select the NGS: Mapping → Map with BWA for Illumina tool.
2. Choose hg19 Full as the reference genome.
3. Click Execute.

Understanding some of the options for BWA

BWA in Galaxy is designed for short queries up to ~200bp with low error rate (<3%). It performs gapped global alignment with respect to reads, supports paired-end reads, and also visits suboptimal hits. This is probably the most widely used and least understood mapping algorithm.

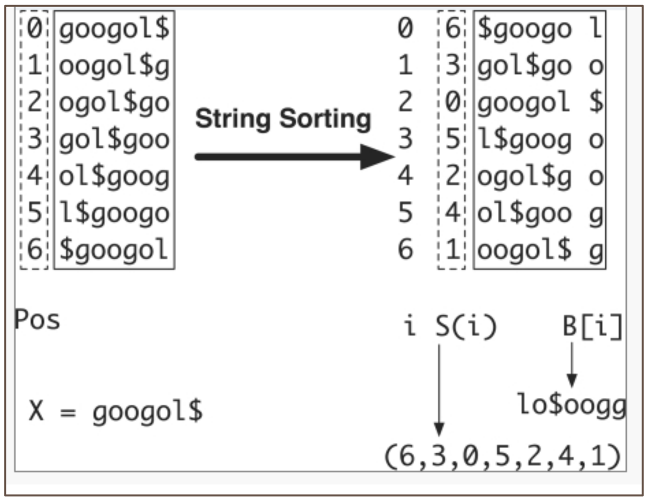
BWA is based on the Burrows-Wheeler transform, a reversible string transformation. This is a way of matching strings that has a small memory footprint and is able to count the number of matches to a string independent of the size of the genome.

The first step is to transform the genome. This is done via the index algorithm in BWA, and will usually take a few hours, but is already done in Galaxy for all the major genomes.

The Burrows-Wheeler transform essentially:

1. Adds a symbol to the end of the string to be transformed, which is lexicographically smaller than all the other symbols in the string.
2. Generates a list of strings, the same length as the original string (with added symbol), but where each letter is circularly moved forward one step.
3. Lexicographically sorts the generated strings.
4. We end up with four different variables to store:
    a. A suffix array for that string which lists the original indices of each of the sorted strings;
    b. The BWT string, made up of the last symbols of each of the newly ordered circulated strings;
    c. An indexed first column, where we have the start and end index of any given letter in the alphabet;
    d. A rank for each letter at each position, which tells us the number of occurrences of that letter above that row in the BWT.

Now, if we are trying to match a new string that is a substring of the original string, each occurrence of that substring will occur within an interval of the suffix array, because all the circulated strings that begin with that substring will have been sorted together. Once we find the suffix interval, we can deduce, from the suffix array, the position(s) of that substring in the original string.



Li and Durbin, Bioinformatics. 2009 Jul 15;25(14):1754-60. "Fast and accurate short read alignment with Burrows-Wheeler transform."

For exact matching, the interval is found by working backwards on your query string:

1. Find the start and end indices of the last character of your query in the indexed first column.
2. At the extremes of this range, find the rank for the next letter in the query string in the BWT.
3. Jump to these ranks of the next letter in the indexed first column.
4. Repeat until you have matched your whole query string. The positions of the final range in the index column will give the suffix array range for the query sequence, where the prefix of every row is our query. If at any time the ranks returned are the same, that means that the next character we want to find is not present in this range and our search stops.

For a very nice visual explanation of this algorithm, visit:
http://blog.avadis-ngs.com/2012/04/elegant-exact-string-match-using-bwt-2/

The 'aln' command finds the suffix array (SA) coordinates (i.e., the suffix interval) of good hits of each individual read, and the 'samse/sampe' command converts the SA coordinates to chromosomal coordinates, and pairs reads (for 'sampe') and generates the SAM-formatted alignments.

In the BWA tool in Galaxy, both the aln and samse/sampe commands are run to result in a SAM format output dataset.

For aln [default values]

-n NUM Maximum edit distance if the value is INT, or the fraction of missing alignments given 2% uniform base error rate if FLOAT. In the latter case, the maximum edit distance is automatically chosen for different read lengths. The maximum number of differences allowed is defined as: 15-37 bp reads: 2; 38-63: 3; 64-92: 4; 93-123: 5; 124-156: 6. [0.04]
-o INT Maximum number of gap opens. [1]
-e INT Maximum number of gap extensions, -1 for k-difference mode (disallowing long gaps) [-1]. This option is critical in allowing the discovery of indels.
-d INT Disallow a long deletion within INT bp towards the 3'-end. [16]
-i INT Disallow an indel within INT bp towards the ends. [5]
-l INT Take the first INT subsequence as seed. If INT is larger than the query sequence, seeding will be disabled. For long reads, this option is typically ranged from 25 to 35 for -k 2. [inf]
-k INT Maximum edit distance in the seed. [2]
-M INT Mismatch penalty. BWA will not search for suboptimal hits with a score lower than (bestScore-misMatchPenalty). [3]
-O INT Gap open penalty. [11]
-E INT Gap extension penalty. [4]
-R INT For paired-end reads only. Proceed with suboptimal alignments if there are no more than INT top hits. By default, BWA only searches for suboptimal alignments if the top hit is unique. Using this option has no effect on accuracy for single-end reads. It is mainly designed for improving the alignment accuracy of paired-end reads. However, the pairing procedure will be slowed down, especially for very short reads (~32bp).
-N Disable iterative search. All hits with fewer than the maximum allowed number of differences will be found. This mode is much slower than the default.

For samse/sampe:

-n INT (samse/sampe) Maximum number of alignments to output in the XA tag for reads paired properly. If a read has more than INT hits, the XA tag will not be written. [3] This is another critical parameter, and will determine whether suboptimal matches are returned.
-r STR (samse/sampe) Specify the read group, formatted as "@RG\tID:text\tSM:text". [null]
-a INT (sampe only) Maximum insert size for a read pair to be considered as being mapped properly. This option is only used when there are not enough good alignments to infer the distribution of insert sizes. [500]
-N INT (sampe only) Maximum number of alignments to output in the XA tag for disconcordant read pairs (excluding singletons). If a read has more than INT hits, the XA tag will not be written. [10]
-o INT (sampe only) Maximum occurrences of a read for pairing. A read with more occurrences will be treated as a single-end read. Reducing this parameter helps faster pairing. [100000]

[INT: integer; STR: string]

SAM [Sequence Alignment/Map] Format

The Sequence Alignment/Map (SAM) format is a generic nucleotide alignment format that describes the alignment of query sequences or sequencing reads to a reference sequence. It can store all the information about an alignment that is generated by most alignment programs. Importantly, it is a compact representation of the alignment, and can allow many of the operations on the alignment to be performed without loading the whole alignment into memory. The SAM format also allows the alignment to be indexed by reference sequence position to efficiently retrieve all reads aligning to a locus.

The SAM format consists of a header section and an alignment section.

Header section

The header section includes information about the alignment and the program that generated it. All lines in the header section are tab-delimited and begin with a "@" character, followed by tag:value pairs, where tag is a two-letter string that defines the content and the format of value.

There are five main sections to the header, each of which is optional:
@HD. The header line. If this is present, it must be the first line, and must include:
    VN: the format version.
@SQ. Includes information about the reference sequence(s). If this section is present, it must include two fields for each sequence:
    SN: the reference sequence name.
    LN: the reference sequence length.
@RG. Includes information about read groups. This can be used multiple times, once for each read group. If this section is present, each @RG section must include:
    ID: the read group identifier.
If an RG tag appears anywhere in the alignment section, there should be a single corresponding @RG line with matching ID tag in the header section.
@PG. Includes information about the program generating the alignment. Must include:
    ID: The program identifier.
If a PG tag appears anywhere in the alignment section, there should be a single corresponding @PG line with matching ID tag in the header section.
@CO. These are unstructured one-line comment lines which can be used multiple times.

Alignment section

Each alignment line has 11 mandatory fields and a variable number of optional fields. These fields always appear in the same order and must be present, but their values can be '0' or '*' (depending on the field) if the corresponding information is unavailable.

Mandatory Alignment Section Fields

Position	Field	Description
1	QNAME	Query template (or read) name
2	FLAG	Information about read mapping (see next section)
3	RNAME	Reference sequence name. This should match a @SQ line in the header.
4	POS	1-based leftmost mapping position of the first matching base. Set as 0 for an unmapped read without coordinate.
5	MAPQ	Mapping quality of the alignment. Based on base qualities of the mapped read.
6	CIGAR	Detailed information about the alignment (see relevant section).
7	RNEXT	Used for paired end reads. Reference sequence name of the next read. Set to "=" if the next segment has the same name.
8	PNEXT	Used for paired end reads. Position of the next read.
9	TLEN	Observed template length. Used for paired end reads and is defined by the length of the reference aligned to.
10	SEQ	The sequence of the aligned read.
11	QUAL	ASCII of base quality plus 33 (same as the quality string in the Sanger FASTQ format).
12	OPT	Optional fields (see relevant section).

FLAG field

The FLAG field includes information about the mapping of the individual read. It is a bitwise flag, which is a way of compactly storing multiple logical values as a short series of bits where each of the single bits can be addressed separately.

FLAG fields

Hex	Binary	Description
0x1	00000000001 (1)	The read is paired
0x2	00000000010 (2)	Both reads in a pair are mapped "properly" (i.e., in the correct orientation with respect to one another)
0x4	00000000100 (4)	The read itself is unmapped
0x8	00000001000 (8)	The mate read is unmapped
0x10	00000010000 (16)	The read has been reverse complemented
0x20	00000100000 (32)	The mate read has been reverse complemented
0x40	00001000000 (64)	The read is the first read in a pair
0x80	00010000000 (128)	The read is the second read in a pair
0x100	00100000000 (256)	The alignment is not primary (a read with split matches may have multiple primary alignment records)
0x200	01000000000 (512)	The read fails platform/vendor quality checks
0x400	10000000000 (1024)	PCR or optical duplicate

In a run with single reads, the only flags you will see are:

0	None of the bitwise flags have been set. This read has been mapped to the forward strand.
4	The read is unmapped.
16	The read is mapped to the reverse strand.

Some common flags that you may see in a paired experiment include:

69	1 + 4 + 64	The read is paired, is the first read in the pair, and is unmapped.
73	1 + 8 + 64	The read is paired, is the first read in the pair, and it is mapped while its mate is not.
77	1 + 4 + 8 + 64	The read is paired, is the first read in the pair, but both are unmapped.
133	1 + 4 + 128	The read is paired, is the second read in the pair, and it is unmapped.
137	1 + 8 + 128	The read is paired, is the second read in the pair, and it is mapped while its mate is not.
141	1 + 4 + 8 + 128	The read is paired, is the second read in the pair, but both are unmapped.

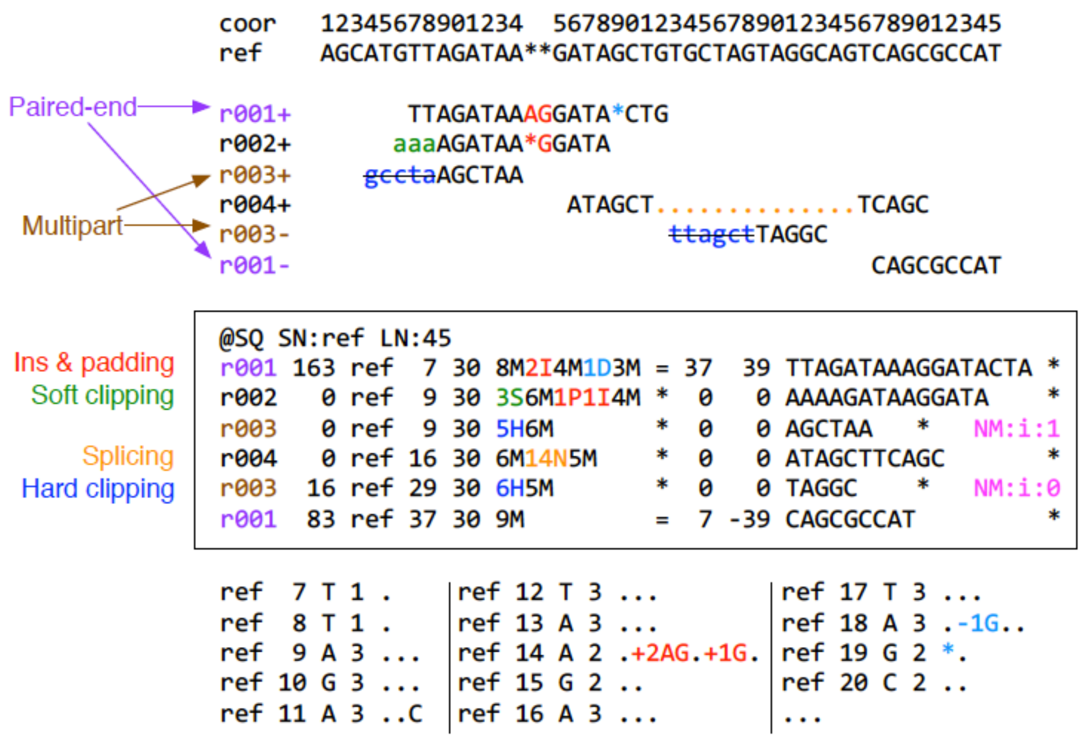
CIGAR [Concise Idiosyncratic Gapped Alignment Report] String

The CIGAR string describes the alignment of the read to the reference sequence. It is able to handle (soft- and hard-) clipped alignments, spliced alignments, multi-part alignments and padded alignments (as well as alignments in color space). The following operations are defined in CIGAR format:

CIGAR Format Operations

Operation	Description
M	Alignment match (can be a sequence match or mismatch)
I	Insertion to the reference
D	Deletion from the reference
N	Skipped region from the reference
S	Soft clipping (clipped sequences present in read)
H	Hard clipping (clipped sequences NOT present in alignment record)
P	Padding (silent deletion from padded reference)
=	Sequence match (not widely used)
X	Sequence mismatch (not widely used)

- H can only be present as the first and/or last operation.
- S may only have H operations between them and the ends of the CIGAR string.
- For mRNA-to-genome alignments, an N operation represents an intron. For other types of alignments, the interpretation of N is not defined.
- The sum of lengths of the M/I/S/=/X operations must equal the length of the read.



```
coor    12345678901234  5678901234567890123456789012345
ref     AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

r001+         TTAGATAAAGGATA*CTG
r002+        aaaAGATAA*GGATA
r003+      ~~gccta~~AGCTAA
r004+                    ATAGCT..............TCAGC
r003-                           ~~ttagct~~TAGGC
r001-                                         CAGCGCCAT
```

```
@SQ SN:ref LN:45
r001 163 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTA *
r002   0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003   0 ref  9 30 5H6M       *  0   0 AGCTAA    *   NM:i:1
r004   0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003  16 ref 29 30 6H5M       *  0   0 TAGGC     *   NM:i:0
r001  83 ref 37 30 9M         =  7 -39 CAGCGCCAT         *
```

```
ref  7 T 1 .    |ref 12 T 3 ...        |ref 17 T 3 ...
ref  8 T 1 .    |ref 13 A 3 ...        |ref 18 A 3 .-1G..
ref  9 A 3 ...  |ref 14 A 2 .+2AG.+1G. |ref 19 G 2 *.
ref 10 G 3 ...  |ref 15 G 2 ..         |ref 20 C 2 ..
ref 11 A 3 ..C  |ref 16 A 3 ...        |...
```

Li et al, Bioinformatics (2009) 25 (16): 2078-2079. "The Sequence Alignment/Map format and SAMtools"

OPT field

The optional fields are presented as key-value pairs in the format of TAG:TYPE:VALUE, where TYPE is one of:

• A	Printable character
• I	Signed 32-bin integer
• F	Single-precision float number
• Z	Printable string
• H	Hex string

The information stored in these optional fields will vary widely with the mapper.

They can be used to store extra information from the platform or aligner. For example, the RG tag keeps the 'read group' information for each read, where a read group can be any set of reads that use the same protocol (sample/library/lane). In combination with the @RG header lines, this tag allows each read to be labeled with metadata about its origin, sequencing center and library.

Other commonly used optional tags include:

NM:i	Edit distance to the reference
MD:Z	Number matching positions/mismatching base
AS:i	Alignment score
BC:Z	Barcode sequence
X0:i	Number of best hits
X1:i	Number of suboptimal hits found by BWA
XN:i	Number of ambiguous bases in the reference
XM:i	Number of mismatches in the alignment
XO:i	Number of gap opens
XG:i	Number of gap extensions
XT:A	Type of match (Unique/Repeat/N/Mate-sw)
XA:Z	Alternative hits; format: (chr,pos,CIGAR,NM)
XS:i	Suboptimal alignment score
XF:	Support from forward/reverse alignment
XE:i	Number of supporting seeds

Thus, for example, we can use the NM:i:0 tag to select only those reads which map perfectly to the reference (i.e., have no mismatches). If we wanted to select only those reads which mapped uniquely to the genome, we could filter on the XT:A:U (where the U stands for "unique").

Mapping example continued

Once the mapping is complete, we want to select only those reads that have mapped uniquely to the genome.
[Note: if your BWA mapping run hasn't finished yet, get a prepared result file from the shared history named MappingResults, using the Copy Datasets function]
Note that the header lines of our output include all the names of the reference sequences that our reads were being mapped to, i.e., every chromosome and its length.

In the final OPT column of the SAM output, BWA includes many additional pieces of information. For example, we can use the NM:i:0 tag to select only those reads which map perfectly to the reference (i.e., have no mismatches). In this case, we want to select only those reads which mapped uniquely to the genome. To do this, we filter on the XT:A:U (where the U stands for "unique").

1. Open the Filter and Sort → Select tool.
2. Input XT:A:U as the pattern to match.

Now that we have a filtered set of results, in SAM format, we want to convert them into BAM format (which is the binary indexed version of the SAM data).

1. Open the NGS SAM Tools → SAM to BAM tool. Make sure the filtered SAM dataset is selected.
2. Click Execute.

We can retrieve some simple statistics on our BAM file:

1. Open the NGS: SAM Tools → flagstat tool. This reads the bitwise flags from the SAM/BAM output and prints out their interpretation.
2. Click Execute.

We can see that 99.91% of our (filtered) reads were mapped to the reference genome.

The SAM and BAM formats are read-specific, in that every line in the file refers to a read. We want to convert the data to a format where every line represents a position in the genome instead, known as a pile-up.

1. Open the NGS SAM Tools → MPileup tool. The MPileup tool is a newer, more complex, version of the pileup tool which can handle multiple BAM files. It also introduces the concept of BAQ (base alignment quality) which takes into account local realignment of reads around putative SNP positions and will modify base qualities around these positions in an attempt to avoid calling false SNPs.
2. Set the reference genome to hg19.
3. Select the Advanced Options.
4. Change the coefficient for downgrading mapping quality for reads containing excessive mismatches to 50. This reduces the effect of reads with excessive mismatches and is a fix for overestimated mapping quality.
5. Click Execute.

Pileup Format

Each line in the pileup format includes information on:

1. The reference sequence (chromosome) name.
2. The position in the reference sequence.
3. The reference base at that position (uppercase indicates the positive strand; lowercase the negative strand). An asterisk marks an indel at that position.
4. The number of reads covering that position.
5. The read base at that position. This column also includes information about whether the reads match the reference position or not. A "." stands for a match to the reference base on the forward strand, a "," for a match on the reverse strand, "[ACGTN]" for a mismatch on the forward strand and "[acgtn]" for a mismatch on the reverse strand. A pattern "\[+-][0-9]+[ACGTNacgtn]+" indicates there is an insertion (+) or deletion (-) between this reference position and the next reference position. Finally, a "^" indicates the start of a new, or soft- or hard-clipped read, followed by the mapping quality of the read. The "$" symbol marks the end of a read segment.
6. The quality scores for each read covering that position.

For more detailed information on the pileup format, go to http://samtools.sourceforge.net/pileup.shtml

Analyze pileup

Since this is a heteroplasmic experiment, we reduce this dataset to ensure we only include the mitochondrial genome.

1. Use the Filter and Sort → Filter tool.
2. Set the search condition to c1 == 'chrM'. Make sure to include the quotes around the search term.

To see how well the various positions in the mitochondrial genome are covered

1. Open the Statistics → Summary Statistics tool.
2. Choose c4. This is the fourth column from the pileup (i.e., the column which shows the number of reads covering that position.)

In this case, we can see that the coverage is quite high (the mean is 1856 reads per base).

To extract positions that are likely to be heteroplasmic, we should include two pieces of information: the coverage (only keep those positions above a certain threshold of coverage) and the quality (only keep those base reads above a certain quality threshold, as they are more likely to be real heteroplasmic sites rather than sequencing errors). To filter out only those positions which pass these thresholds:

1. Open the NGS: SAM Tools → Filter pileup tool.
2. Make sure that the dataset that you are working on is the filtered pileup, not the summary statistics dataset.
3. Require that a base call must have a quality of at 30 to be considered.
4. Remove all bases covered by fewer than 200 reads.

5. Change the pulldown menus so that:
    a. the tool reports variants only;
    b. coordinates are converted to intervals (this makes it easier to join the results of this analysis with gene/promoter annotations);
    c. the total number of differences is reported;
    d. the quality and base strings are not returned (since these will be very large fields).

We can now remove any positions whose quality adjusted coverage does not meet our threshold of 200 using a secondary filtering step.

1. Open the Filter and Sort → Filter tool.
2. Use "c10>=200" as the filter condition.

We may also want to perform additional filtering steps, such as only keeping those sites which show at least a 0.15% frequency.

1. Open the Text Manipulation → Compute tool.
2. Enter (c11/c10) * 100.0 into the expression box. This will calculate the percentage of the quality adjusted coverage at each base that is represented by the total number of variants.
3. Do not round the result; keep the calculation as a floating point number.
4. Click Execute.
5. Open the Filter and Sort → Filter tool.
6. Enter c12 > 0.15 as the condition to filter on.
7. Click Execute.
8. Open the Filter and Sort → Sort tool.
9. Sort on c12 (the percentage column).
10. Sort in descending order.
11. Click Execute.

Note that this is a very simplistic way of filtering as it ignores the fact that some of the variant positions differ from the reference in every read (and will be seen here as 100% different) but show no heteroplasmy amongst themselves. Another possibility may be to filter out those sites which only have one difference from the reference.

Exercise

1. Make a workflow of this history. Leave out the summary statistics and boxplot steps. Name the workflow Mapping BWA Reads. Make sure all the steps connect correctly to one another.
Import the Blood-PCR2 dataset from the mtProjectDemo library into your history.
Rerun the workflow with this new dataset. (Run a quality control check on this dataset before running the workflow).

2. Associate the resulting heteroplasmies with gene / transcript / exon / intron information.

Filtering Reads by Mapping Quality Workflow

We can build a workflow that incorporates filtering on the SAM output.
Upload some PhiX reads and a PhiX genome from Emory using the following URLs:
http://bx.mathcs.emory.edu/outgoing/data/phiX174_genome.fa
http://bx.mathcs.emory.edu/outgoing/data/phiX174_reads.fastqsanger

1. Use the Get Data → Upload File tool.
2. Enter the URLs into the URL/Text entry box.
3. Click Execute.

The PhiX genome has been successfully identified as a FastA file. Change the name of this dataset to PhiX Genome. The reads have been identified as being in the FastQ format, but we need to specify which FastQ format. In the attributes panel, accessed by clicking the pencil icon for that dataset, change the name of this dataset to PhiX reads, and change the data type to fastqsanger.

We next align the reads to a PhiX genome. Although Galaxy has a built-in PhiX genome, we will use the one that we downloaded from Emory.

1. Select the NGS: Mapping → Map with BWA for Illumina tool.
2. Change the reference genome pulldown menu to say Use one from the history.
3. Make sure that the PhiX Genome dataset is selected in the second pulldown menu, and that the PhiX reads are selected as the FastQ file.
4. For this example, leave the settings at the default option.

This will generate a set of reads aligned to the PhiX genome in SAM format.

We can now mine the SAM data however we wish. Say we want to only select those reads that mapped perfectly to the genome. One of the optional pieces of information that is output by the BWA program is the edit distance to the reference (NM:i:x). If we want to select only those reads which matched the reference exactly, the number at the end of that tag should be zero.

1. Open the Filter and Sort → Select tool.
2. Input NM:i:0 as the string to be matched.
3. Click Execute.

Extract this workflow using the Extract Workflow option from the gear icon. Rename it to Subset Reads and save. Select the workflow from the Workflows tab. Clicking on the representation of the Select tool allows you to change the pattern that is being matched (for example, if we wanted to change it to select reads that were an edit distance of 1 away from the reference, we could change the pattern to NM:i:1. You can change the name of the output from this Select operation by choosing the Rename Dataset from the pulldown menu in the Edit Step Actions section and clicking Create. Make sure to save the updated workflow from the Options menu at the top right of the workflow editor.

RNA-Seq analysis with TopHat tools

RNA-Seq

RNA-Seq experiments are designed to identify the RNA content (or transcriptome) of a sample directly. These experiments allow the identification of relative levels of alleles, as well as detection of post-transcriptional mutations or detection of fusion genes. Most importantly, the RNA-Seq technique allows the comparison of the transcriptomes of different samples, most usually between tumor and normal tissue, to enable insight into the differential expression patterns seen in each state. RNA-Seq is the next generation version of other experimental techniques for describing transcriptomes, such as microarrays or EST sequencing, and can do so with fewer biases and at a higher resolution.

The alignment of RNA-Seq read data to a genome is complicated by the fact that the reads come from spliced transcripts and therefore there will be many intronic regions to deal with (i.e., regions that are present in the genome being aligned to, but not in the reads). One way of dealing with this problem is to align the reads against a set of (already spliced) known exonic sequences. The main drawback with this method is that if the set of known exonic sequences is incomplete, there will be many unalignable reads. Another approach is that taken by TopHat, which allows the identification of novel splice junctions.

Mapping reads to the transcriptome with TopHat

We are going to use two small datasets of under 100,000 single 75-bp reads from the ENCODE GM12878 cell line and ENCODE h1-hESC cell line, and compare the transcriptomes between the two cell lines.

1. Open the Get Data → Upload File tool.
2. Get the two RNA-Seq Analysis datasets by entering the following URLs in the text entry box.

http://chagall.med.cornell.edu/galaxy/rnaseq/GM12878_rnaseq1.fastqsanger
http://chagall.med.cornell.edu/galaxy/rnaseq/h1hESC_rnaseq2.fastqsanger

3. Change the File-Format pulldown menu from 'Auto-detect' to 'Fastqsanger'.
4. Do a quality control check on the data before beginning any analysis.
    a. Run the NGS: QC and manipulation → FASTQ Summary Statistics tool.
    b. Use the Graph/Display Data → Boxplot tool to visualize the quality score data summary.

In the case of the GM12878 dataset, there is a dramatic decrease in quality around base 60, so we want to trim off the last 16 bases from each read.

1. Open the NGS: QC and manipulation → FASTQ Trimmer tool.
2. Use 16 as offset from the 3' end.

In the case of the h1-hESC data, the data is mostly high quality with a single base in the middle with a median quality of 20, and tailing off to a median quality of around 22. These are all acceptable, so we will not trim the h1-hESC dataset at all.

The next step is to align the now filtered reads to the genome using TopHat.

1. Open the NGS: RNA Analysis → Tophat for Illumina tool.
2. Select the hg19 Full genome from the organism pull down menu.
3. Change the settings from defaults to display the full parameter list. In general, it is usually okay to keep most of the default parameters, but it is usually good practice to go down the list and make sure that they all look appropriate for the current analysis. Note that some of the options are only applicable to paired end reads (e.g., Library type and Use closure search).
4. Reduce the maximum intron length (both for initial (whole read) searches and split-segment searches) down to 100000.
5. Turn off indel search and coverage search, to speed up the analysis.
6. Do this for both datasets.

TopHat

TopHat is designed specifically to deal with junction mapping and overcomes the limitation of relying on annotation of known splice junctions. It does this by first aligning as many reads as it can to the genome, using the Bowtie aligner; those reads that align will be the ones that fit completely within an exonic region (any reads that are mapped non-contiguously are those that contain intronic regions). TopHat then tries to assemble the mapped reads into consensus sequences, using the reference sequence to determine consensus. To ensure that the edges of the exons are also covered, TopHat uses a small amount of flanking sequence from the reference on both sides to extend the consensus.

Once these alignment steps are complete, TopHat builds a database of possible splice junctions and tries to map reads against these junctions to confirm them. Specifically, TopHat tries to identify splice junctions with the known splice acceptor and donor sites GT-AG, GC-AG and AT-AC. TopHat has three ways in which it can define a potential junction. The first method of identifying/verifying potential junctions is when the short segments of a single read map far apart from each other on the same chromosome ("split-segment search"). For each splice junction, TopHat will search the initially unmapped reads to find any that can span that junction. The second method by which TopHat predicts junctions is called "coverage search", where TopHat tries to find possible introns within deeply sequenced islands. The third method is called "closure search", applicable only to paired end reads. If the two reads are mapped further apart from one another than the expected distance, TopHat assumes that they come from different exons and attempts to join them by looking for subsequences in the genomic interval between them that approximates the expected distance between them.

Options in TopHat

Mean inner distance between mate pairs. [PE only] If dealing with paired end reads, TopHat needs to be told the expected distance between those paired reads.
Library type. [PE only] Determines to which strand TopHat will attempt to align reads. fr-unstranded is the standard Illumina paired end situation where the left-most end of the read is mapped to the transcript strand and the right-most end is mapped to the other strand. fr-firststrand assumes that only the right-most end of a fragment is sequenced, whereas fr-secondstrand assumes that only the left-most end of the fragment is sequenced.
Anchor length. This is the minimum number of bases from aligned reads that have to be present on either side of a junction for it to be recognized by TopHat as a potential junction. Default: 8.
Maximum number of mismatches in anchor region. This defines the maximum number of mismatches allowed in the anchor region defined by the Anchor Length option. Default: 0.
Minimum intron length. This is the minimum amount of distance that can separate two exons (i.e., if two exons are closer than this, TopHat will not search for splice acceptor/donor sites between them and instead will assume that the exon has low coverage in the middle and attempt to merge it into one exon instead). Default: 70.
Maximum intron length. This is the maximum amount of distance that can separate two exons (i.e., if two exons are further apart than this, TopHat will not search for splice acceptor/donor sites between them, except in those cases where two shorter segments of a split-up read support such a distant pairing). Decreasing this distance will increase the speed of the search with a concomitant decrease in sensitivity. Default: 500000.
Allow indel search. Checking this option will allow TopHat to include insertions and deletions in your reads relative to the genome sequence. The length of the allowed insertions and deletions are determined by the two options that open up once this option is checked: Max insertion length and Max deletion length.
Minimum isoform fraction. For each junction, the average depth of read coverage is computed for the left and right flanking regions of the junction separately. The number of alignments crossing the junction is divided by the coverage of the more deeply covered side to obtain an estimate of the minor isoform frequency. The default value for this is set at 0.15, since Wang et al (2008) reported that 86% of the minor isoforms of alternative splicing events in humans were expressed at 15% or higher of the level of the major isoform.
Maximum number alignments. Discards from further analysis reads that map to an excessive number of different regions on the genome. This allows 'multireads' from genes with multiple copies to be reported, but tends to discard failed reads wich map to multiple low complexity regions. Default: 20.
Minimum intron length in split-segment search. The minimum intron length that may be found during split-segment search. Default: 50.
Maximum intron length in split-segment search. The maximum intron length that may be found during split-segment search. Default: 500000.
Number mismatches allowed in initial read mapping. The maximum number of mismatches that may appear in the initial (unsegmented) alignment. Default: 2.
Number of mismatches allowed in each segment alignment. The maximum number of mismatches that may appear in each segment of a segmented read. Default: 2.
Minimum length of read segments. The length of the segments that the reads are split into for the split-segment search.

Use own junctions. This allows you to give TopHat a set of junctions that it can add into its database of potential junctions. This is most commonly used when you have a mixed dataset (e.g., of paired end reads and single reads); run Tophat with one set of reads, save the potential junction database that TopHat produces, and then feed that database into the second run with the rest of original dataset.

Use gene annotation model. Available if supplying junctions to TopHat. TopHat uses the supplied exon records to build a set of known splice junctions for each gene and will add those junctions to its potential junction database.

Use raw junctions. Available if supplying junctions to TopHat. This allows you to supply TopHat with a list of raw junctions, usually from a previous TopHat run. Junctions are specified one per line, in a tab-delimited format. Records are defined as [chrom] [left] [right] [+/-], where left and right are zero-based coordinates, and specify the last character of the left sequence to be spliced to the first character of the right sequence, inclusive.

Only look for supplied junctions. Available if supplying junctions to TopHat. Forces TopHat to not add any more junctions from initial mapping results to its database and only use the junctions that you have supplied (either as gene annotations or as raw junctions.)

Use closure search. Enables the mate pair closure-based search for junctions. Closure-based search should only be used when the expected inner distance between mates is small (<= 50bp).

Use coverage search. Enables the coverage based search for junctions, so that TopHat can search for junctions within islands. Coverage search is disabled by default (such as for reads 75bp or longer), for maximum sensitivity. Enabling this will slow down the analysis dramatically.

Microexon search. With this option, the pipeline will attempt to map reads to microexons (very short exons, usually about 25 bases or less) by cutting up reads into smaller segments and attempting to align them to the genome. Works only for reads 50bp or longer.

Analysis continued

Each TopHat run will result in four files: a list of accepted mapped reads in BAM format , and three BED files: one for raw splice junctions (which can then be fed into a subsequent TopHat analysis), one each for detected insertions and deletions (although these will be empty if the indel search was disabled.)

The splice junctions file is formatted as a BED file, using all optional columns. This will enable the visualization of splice junctions as a pair of boxes joined by a thin line. The column headings are:

1. Chromosome name
2. Start position of junction representation
3. End position of junction representation
4. Name of the junction
5. Junction score (how many reads support the junction)
6. Strand
7. Position at which to start drawing the first box
8. Position at which to end drawing the last box
9. Color with which to paint the junction representation
10. Number of thick boxes (almost always 2)
11. Size of each of the thick boxes
12. Distance of the start position of each box from the start position of the junction representation (first number always 0)

The accepted hits file is a standard BAM file (with the standard SAM information). To verify the contents of this file, we can convert it to a human-readable SAM formatted file with the NGS: SAM Tools → BAM-to-SAM tool.

Because Bowtie allows reads to map to multiple places, and will return all found matches, there are a few tags that you will see in the SAM file that were not used in the BWA output.

NH:i	Number of hits (i.e., number of times the read mapped)
HI:i	Hit index (a way to refer to each separate mapping)
CC:Z	Reference name of the next hit
CP:i	Leftmost coordinate of the next hit

Assembly

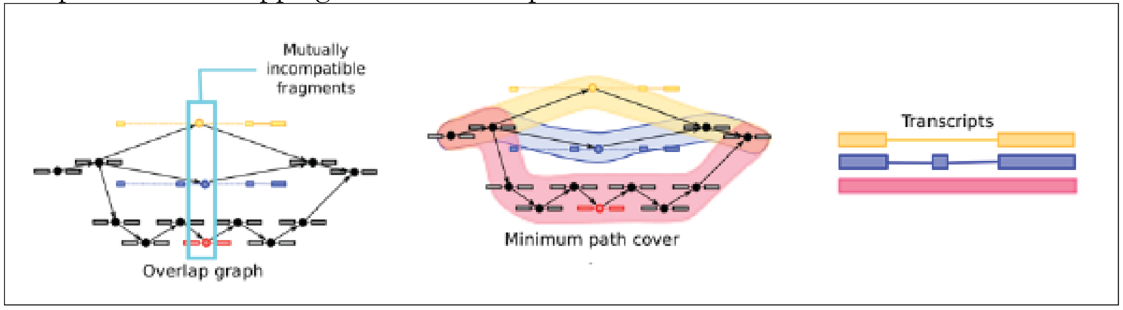
Once the reads have been mapped, we want to assemble the reads into complete transcripts which can then be analyzed for differential splicing events, or differential expression.

This is done using CuffLinks.

1. Run NGS: RNA Analysis → Cufflinks on the two accepted_hits datasets produced by our earlier Tophat analysis (one for each of our two original FastQ input datasets).
2. Change the max intron length to 100000.
3. Click Execute.

CuffLinks

CuffLinks uses a directed acyclic graph algorithm to identify the minimum set of independent transcripts that can explain the reads observed in an RNA-Seq experiment. It does this by grouping reads into clusters that all map to the same region of the genome and then identifying "incompatible" reads, which cannot possibly have come from the same transcript. Once the minimum number of possible transcripts has been identified, it then assigns each read in the cluster to one or more of those transcripts, depending on compatibility. Abundance for each transcript is estimated based on the number of compatible reads mapping to each transcript.



Trapnell et al, Nat Biotechnol (2010) 28(5): 511-515. "Transcript assembly and abundance estimation from RNA-Seq reveals thousands of new transcripts and switching among isoforms"

It is important to note that the input to CuffLinks must be a SAM/BAM file sorted by reference position. If your input is from TopHat, it is probably already in the correct format, but if this is a SAM file of some other provenance, it should be sorted. Below is a sample workflow to sort your SAM data. If you are starting with a BAM file, convert it to a SAM file first, and then back to BAM format after sorting.

1. Open the Filter and Sort → Select tool.
2. Use the pattern ^@ as the criterion for selecting lines. This will select all the SAM header lines.
3. Click Execute.
4. Open the Filter and Sort → Select tool again.
5. Use the same pattern as before, but this time change the pulldown menu to say NOT matching. This ignores all the header lines and selects all the alignment lines that now need to be sorted by reference position.
6. Open the Filter and Sort → Sort tool.
7. Sort the alignment lines file on column 3 (the reference chromosome number), alphabetically in ascending order.
8. Add a second column selection and sort on column 4 (the chromosome position), numerically, in ascending order.
9. Open the Text Manipulation → Concatenate datasets tool.
10. Ensure the first dataset selected is the SAM headers dataset from step 2.
11. Click the Add a new dataset button.
12. Add the sorted alignment lines dataset from step 8.

Options in CuffLinks

Maximum intron length. CuffLinks will not report transcripts with introns longer than this. Default: 300,000.
Minimum isoform fraction. After calculating isoform abundance for a gene, CuffLinks filters out transcripts that it believes are very low abundance, because isoforms expressed at extremely low levels often cannot reliably be assembled, and may even be artifacts of incompletely spliced precursors of processed transcripts. This parameter is also used to filter out introns that have very few spliced alignments supporting them. Default: 0.1 (i.e., 10% of the major isoform).
Pre MRNA fraction. Some RNA-Seq protocols produce a significant amount of reads that originate from incompletely spliced transcripts, and these reads can confound the assembly of fully spliced mRNAs. CuffLinks uses this parameter to filter out alignments that lie within the intronic intervals implied by the spliced alignments. The minimum depth of coverage in the intronic region covered by the alignment is divided by the number of spliced reads, and if the result is lower than this parameter value, the intronic alignments are ignored. Default: 0.15.
Perform quartile normalization. In some cases, a small number of abundant, differentially expressed genes can create the (incorrect) impression that less abundant genes are also differentially expressed. This option allows CuffLinks to exclude the contribution of the top 25 percent most highly expressed genes from the number of mapped fragments used in the FPKM denominator, improving robustness of differential expression calls for less abundant genes and transcripts.
Use reference annotation. Tells CuffLinks to use the supplied reference annotation to estimate isoform expression. It will not assemble novel transcripts, and the program will ignore alignments not structurally compatible with any reference transcript.
Perform bias correction. Requires a reference sequence file. This option forces CuffLinks to detect sequences which are overrepresented due to library preparation or sequencing bias and correct for this. Bias detection and correction can significantly improve accuracy of transcript abundance estimates.
Mean inner distance between mate pairs. [PE only] This is the expected (mean) inner distance between mate pairs. For, example, for paired end runs with fragments selected at 300bp, where each end is 50bp, you should set -r to be 200. The default is 20bp.
Standard deviation for inner distance between mate pairs. [PE only] The standard deviation for the distribution on inner distances between mate pairs. The default is 20bp.

The output from CuffLinks consists of three datasets: a GTF formatted dataset listing the assembled isoforms detected by CuffLinks, and two datasets separating out the coverage data from the GTF datasets for transcripts and for genes.

The GTF dataset contains the following information:

1. Chromosome name
2. Source (always Cufflinks)
3. Feature type (always either 'transcript' or 'exon')
4. Start position of the feature
5. End position of the feature
6. Score (the most abundant isoform for each gene is assigned a score of 1000. Minor isoforms are scored by the ratio (minor FPKM/major FPKM))
7. Strand of isoform
8. Frame (not used)
9. Attributes
    a. gene_id: Cufflinks gene id
    b. transcript_id: Cufflinks transcript id
    c. exon_number: Exon position in isoform. Only used if feature type is exon
    d. FPKM: Relative abundance of isoform
    e. frac (not used)
    f. conf_lo: Lower bound of the 95% CI for the FPKM
    g. conf_hi: Upper bound of the 95% CI for the FPKM
    h. cov: Depth of read coverage across the isoform

RPKM [Reads Per Kilobase per Million reads mapped]

RPKM is a measurement of transcript reads that has been normalized both for transcript length and for the total number of mappable reads from an experiment. This normalized number helps in the comparison of transcript levels both within and between samples. Normalizing by the total number of mapped reads allows comparison between experiments (since you may get more mapped reads in one experiment), whereas normalizing by the length of the transcript allows the direct comparison of expression level between differently sized transcripts (since longer transcripts are more likely to have more reads mapped to them than shorter ones).

$$RPKM = \frac{totalTranscriptReads}{mappedReads(millions)\ x\ transcriptLength(Kb)}$$

where mappedReads is the number of mapped reads in that experiment.

You will also see the term FPKM, where the F stands for Fragments. This is a similar measure to RPKM used for paired end experiments, where a fragment is a pair of reads.

Note that in our example, the RPKM numbers will be far higher than normally seen since the total number of mapped reads in our dataset is small (because our input dataset was a subset selected to map to a small region of chromosome 19).

Note that RPKM may not be the best method of quantifying differential expression. Other methods include DESeq and TMM from the Bioconductor package.

Comparison with reference annotations

We need to download a set of reference annotations against which to compare the transcripts assembled from the RNA-Seq experiments.

1. Open the Get Data → UCSC Main tool.
2. Select the hg19 assembly.
3. Make sure the Genes and Gene Prediction Tracks group is selected, choose the RefSeq genes track.
4. Change the region from the genome radio button to the position button, and enter chr19 (since all our reads map to chromosome 19).
5. Change the output format to GTF (gene transfer format). Make sure the Send to Galaxy button is checked and click the Get Output button.

To compare the assembled transcripts against the RefSeq data:

1. Open the NGS: RNA Analysis → Cuffcompare tool.
2. Select the CuffLinks assembled transcripts GTF-formatted dataset for the h1-hESC data.
3. Add a new Additional GTF Input File and select the GM12878 assembled transcripts.
4. Set the Use Reference Annotation to Yes and choose the GTF-formatted RefSeq Genes datasets.
5. Since we will be looking at only a small section of chromosome 19, check the box for Ignore reference transcripts that are not overlapped by any transcript in input files.
6. Set Use Sequence Data to Yes.
7. Click Execute.

CuffCompare

CuffCompare compares the assembled transcripts to a reference annotation and details the identities and differences between them.

Options in CuffCompare

<u>Use Reference Annotation.</u> An optional “reference” annotation GTF. Each sample is matched against this file, and sample isoforms are tagged as overlapping, matching, or novel where appropriate.

<u>Ignore reference transcripts that are not overlapped by any transcript in input files.</u> Causes CuffCompare to ignore reference transcripts that are not overlapped by any transcript in your assembled transcripts datasets. Useful for ignoring annotated transcripts that are not present in your RNA-Seq samples and thus adjusting the "sensitivity" calculation in the accuracy report written in the transcripts_accuracy file

<u>Use Sequence Data.</u> Use sequence data for some optional classification functions, including the addition of the p_id attribute required by CuffDiff, which is the identifier for the coding sequence contained by this transcript. Set to Yes if comparing multiple experiments.

Running CuffCompare results in a number of different datasets.

The transcript accuracy dataset calculates the accuracy of each of the transcripts as compared to the reference at various levels (nucleotide, exon, intron, transcript, gene), e.g., how often an exon that was predicted by the output from CuffLinks was actually seen in the reference. The Sn and Sp columns calculate sensitivity (the proportion of exons, for example, that have been correctly identified) and specificity (the proportion of predicted exons that are annotated as such in the reference) at each level, while the fSn and fSp columns are "fuzzy" variants of these same accuracy calculations, allowing for a very small variation in exon boundaries to still be counted as a match.

There are two tmap datasets, one for each of the input assembled transcripts dataset. The tmap dataset lists the most closely matching reference transcript for each transcript identified by CuffLinks for that dataset. Each row in the dataset contains the following information:

1. ref_gene_id: Reference gene name, derived from the gene_name attribute from the reference GTF record, if present, else the gene_id.
2. ref_id: Reference transcript id, derived from the transcript_id attribute from the reference GTF record.
3. class_code: Relationship between the CuffLinks transcript and the reference transcript.
4. cuff_gene_id: The gene_id from the CuffLinks assembled transcripts dataset.
5. cuff_id: The transcript_id from the CuffLinks assembled transcripts dataset.
6. FMI: Expression level of transcript expressed as fraction of major isoform. Ranges from 1 to 100.
7. FPKM: Expression of this transcript.
8. FPKM_conf_lo: The lower limit of the 95% FPKM CI.
9. FPKM_conf_hi: The upper limit of the 95% FPKM CI.
10. cov: The estimated average depth of read coverage across the transcript.
11. len: The length of the transcript.
12. major_iso_id: The CuffLinks transcript_id of the major isoform for this transcript's gene.
13. ref_match_len: Length of the matching reference gene.

The class codes are defined as follows:

=	Match.
c	Contained.
j	New isoform.
e	A single exon transcript overlapping a reference exon and at least 10 bp of a reference intron, indicating a possible pre-mRNA fragment.
i	A single exon transcript falling entirely with a reference intron.
r	Repeat. Currently determined by looking at the reference sequence and applied to transcripts where at least 50% of the bases are lower case.
p	Possible polymerase run-on fragment.
u	Unknown, intergenic transcript.
o	Unknown, generic overlap with reference.
x	Exonic overlap with reference on the opposite strand.

Using these codes, we could now, for example, extract all new isoforms from the tmap dataset by using the Filter and Sort → Filter and using c3 == 'j' as our match criterion.

There are two refmap datasets, one for each input assembled transcripts dataset. The refmap dataset lists, for each reference transcript, all the transcripts identified by CuffLinks for that dataset that at least partially match it. Each row in the dataset contains the following information:

1. ref_gene_id: Reference gene name, derived from the gene_name attribute from the reference GTF record, if present, else the gene_id.
2. ref_id: Reference transcript id, derived from the transcript_id attribute from the reference GTF record.
3. class_code: Relationship between the CuffLinks transcript and the reference transcript. (Can only be either = (full match) or c (partial match)).
4. cuff_id_list: A list of all CuffLinks transcript_ids matching the reference transcript.

The transcript tracking dataset is created when multiple assembled transcript datasets are given to CuffCompare as input. The tracking file matches transcripts up between samples, assigning each (unified) transcript its own internal transfrag and locus id.

The last dataset is a GTF-formatted combined transcripts dataset which contains one line for every exon identified, its position in the genome, along with a number of attributes

1. gene_id: the internal locus id (XLOC_*) as defined in the tracking file.
2. transcript_id: the internal transcript id (TCONS_*) as defined in the tracking file.
3. exon_number: the position of that exon in the transcript.
4. gene_name: if mapped to an annotated CDS, the name of the gene the transcript has been mapped to.
5. oId: the transcript id assigned to that transcript by CuffLinks.
6. nearest_ref: the name of the nearest gene in the reference annotation.
7. class_code: Relationship between the CuffLinks transcript and the reference transcript.
8. tss_id: an identifier for the inferred transcription start site for the transcript that contains that exon. Determines which primary transcript this processed transcript is believed to come from.
9. p_id: if mapped to an annotated CDS, an identifier for the sequence coded for by that gene.

Comparison between RNA-Seq experiments

Finally, we can use the combined transcripts dataset to compare the relative expression levels of each exon and each transcript in both of our original input datasets, using CuffDiff.

1. Open the NGS: RNA Analysis → Cuffdiff tool.
2. Select the combined transcripts dataset from CuffCompare.
3. For the first BAM file, choose the output from TopHat of accepted reads for the GM12878 input dataset, and for the second, the one for the h1-hESC dataset.
4. Click Execute.

CuffDiff

CuffDiff tests the significance of gene and transcript expression levels in more than one condition. It takes as input the GTF-formatted dataset of transcripts, along with (at least) two SAM/BAM datasets containing the accepted mappings for the samples under consideration and outputs changes in expression at the level of transcripts, primary transcripts, and genes as well as changes in the relative abundance of transcripts sharing a common transcription start site (tss_id), and in the relative abundances of the primary transcripts of each gene. The p_id which was generated by CuffCompare is used to denote a coding sequence (and is only used when the reference sequence includes CDS records). To calculate whether the relative expression of a transcript isoform is significantly different between two samples, CuffDiff uses a two-tailed t-test which incorporates information about the variability in the number of fragments generated by the transcript across replicates (if available), and also incorporates any uncertainty in the expression estimate itself, which is calculated based on how many other transcript expression levels a read could be contributing to.

Options in CuffDiff

False Discovery Rate. The allowed false discovery rate (used for multiple hypothesis correction). Default: 0.05.
Min Alignment Count. The minimum number of alignments needed to conduct significance testing on changes for a transcript. If a transcript does not have the minimum number of alignments, no testing is performed, and any expression changes are deemed not significant, and that transcript does not contribute to correction for multiple testing. Default: 10.
Perform quartile normalization. As for CuffLinks, this option allows the exclusion of the contribution of the top 25 percent most highly expressed genes from the number of mapped fragments used in the FPKM denominator, to improve robustness of differential expression calls for less abundant genes and transcripts.
Perform Bias Correction. Again, similar to the option in CuffLinks, designed to significantly improve accuracy of transcript abundance estimates. Requires a reference sequence file. Detects sequences which are overrepresented due to library preparation or sequencing bias and corrects for this.

CuffDiff outputs many files, including FPKM tracking datasets at different levels (i.e., individual transcripts, gene (combines all transcripts sharing a gene_id), coding sequence (combines all transcripts sharing a p_id), transcript start site (combines all transcripts sharing a tss_id)) and differential expression test datasets for each of these same levels, testing the significance of the relative expression levels of each group between samples.

In addition to the standard information about each object (id, position, annotation), each of the FPKM tracking datasets also includes information about the estimated read depth for each sample, the FPKM +/- 95% CI for each sample as well as the quantification status for that sample (options include: OK (deconvolution successful), LOWDATA (too complex or shallowly sequenced), HIDATA (too many fragments in locus), or FAIL, (when an ill-conditioned covariance matrix or other numerical exception prevents deconvolution)).

The differential expression test datasets summarize the t-test result that calculates the significance of the differential expression levels between samples. These datasets include information about the calculated p-value, the corrected p-value (called q-value) adjusted for FDR using Benjamini-Hochberg multiple hypothesis correction and whether the q-value is significant or not.

We can now inspect these datasets and retrieve transcripts that may be of interest. For example, we can search for novel isoforms.

1. Open the Filter and Sort → Filter tool.
2. Select the transcript FPKM tracking dataset.
3. Use c2 == 'j' as a filter, where c2 is the column containing the class code for the relationship between the assembled CuffLinks transcript and the reference annotation. This will retrieve all assembled transcripts that do match any annotated isoforms ('j' indicating a new isoform).
4. Click Execute.
5. Open the Filter and Sort → Filter GTF data by attribute values_list tool.
6. Select the Cuffcompare combined transcripts dataset.
7. Select the transcript_id field from the pulldown menu and make sure that the filtered dataset containing only the significantly differentially expressed transcripts is selected as the dataset to filter on. Note that if you want to select on any other field, the filtered dataset must be manipulated such that the field you want to select on is the first field in the dataset.
8. Click Execute.
9. The result will be a GTF formatted dataset containing only the novel transcript isoforms, one exon per line, that is now viewable in a genome browser.

We can also search for transcripts that are significantly differentially expressed between the two samples.

1. Open the Filter and Sort → Filter tool.
2. Choose the transcript differential expression testing dataset generated by CuffDiff.
3. Filter on c14=='yes' (where c14 is the column that denotes whether the corrected differential expression is significant).
4. Click Execute.

Visualization in UCSC Genome Browser

The best way to investigate the data is to visualize it. The output of TopHat includes a BED and a BAM file which are both formats that the UCSC Genome Browser knows how to deal with. To view the data for the h1-hESC experiment, we can simply click the display at UCSC main link in the history preview window for the h1-hESC Tophat accepted hits dataset. Using the UCSC Genome Browser is useful because it already contains all the features that we may want to compare our results against (i.e., gene annotations). A new browser tab will be opened which now contains the TopHat accepted hits as a custom track. The default appearance of that custom track will be set to 'dense', but that can be changed to 'pack' which will enable us to see each read mapped to the genome. Clicking on any read will take you to a page with detailed information derived from the SAM file about the read, including the alignment of the sequence and the mapping quality.

Going back to the Galaxy browser window, we can now click the display at UCSC main link in the history preview window for the splice junctions file for the h1-hESC experiment and this will be added to the custom tracks and can be viewed simultaneously with the accepted hits file. We can now see the reads spanning exons, and the junctions that define those exon boundaries.

Finally, we can add the CuffLinks assembled transcripts dataset for the h1-hESC dataset to the visualization which shows the complete transcripts alongside the mapped reads, junctions, and reference genes.

Scroll around the genome browser to find examples of correctly and incorrectly called junctions or missed annotations (e.g., at positions chr19:274,000-297,000, chr19:1,010,000-1,020,000 or chr19:75,353-86,513). Of note is that this particular experiment fails to join many of the longer exons as there are not enough reads spanning their whole length, and annotates them as two (or more) different transcripts (e.g., KLF16).

Visualization in Galaxy

Galaxy also has a framework for visualizing results. Unlike the UCSC Genome Browser, though, we need to import all the annotations that we would like to compare our results to.

To create a new visualization in Galaxy:

1. Select New Visualization from the Visualization tab in the main Galaxy menu at the top of the page.
2. Name your visualization.
3. Assign the hg19 build as the reference genome.
4. Add datasets to your visualization by clicking on the Add Datasets to Visualization button. Choose both the accepted_hits and splice junctions files from your current history, as well as the uploaded chromosome 19 RefSeq annotations.
5. Choose Insert.
6. To view the data, choose chr19 from the Select Chrom/Contig pulldown menu. The controls are similar to those at the UCSC Genome Browser. To zoom in to the first section of chromosome 19 where all the hits are located, just drag a box over the base positions you want to see in greater detail.
7. To save your visualization, click on the small disk icon in the upper right hand corner.

ChIP-Seq analyses with MACS

ChIP-Seq

ChIP-Seq experiments are designed to map DNA-protein interactions by identifying all the genomic elements that are bound by a protein or transcription factor, coupling chromatin immunoprecipitation with NGS sequencing techniques. Although ChIP-Seq has many advantages over other ChIP-type experiments, the NGS side of it also lends it some idiosyncratic disadvantages. The first of these is that the reads represent only the ends of the ChIP fragments and the user has to extrapolate from these partials where the fragment lies on the genome. Secondly, there are regional biases along the genome, due to sequencing and mapping biases, chromatin structure and genome copy number variations.

MACS

For this exercise, we will be using the MACS (Model-based Analysis of ChIP-Seq) program to call peaks. MACS attempts to address both of these issues. The first issue, of reads representing the ends of the ChIP fragments, is dealt with by searching for patterns of bimodally mapped reads (since the expectation is that fragments are sequenced, on average, at the same rate from either end), with plus-strand reads enriched in one peak, and minus-strands in the other. It calculates the distance between the forward and reverse strand peaks, and then shifts each fragment inward (towards the center of the peak) by half that distance, resulting in the region between the two peaks now being narrowed somewhat and "filled in" as one peak. The second issue of dealing with biases along the genome is dealt with by using a control dataset (which is not available with all datasets). A control dataset will also show many of the same biases that are present in the experimental dataset, so the implication is that any time the experimental dataset shows peaks that are not mirrored by the control dataset, that these peaks are not accounted for by biases and are therefore real.

Options in MACS tool

Note that the version of MACS in Galaxy is not the latest version!
Paired end Sequencing. If this is selected, the tool will expect two input files, and if using a control, will expect two control files. Additionally, it will include a new option Best distance between Pair-End Tags which MACS will use to decide the best mapped locations for 3' and 5' pairs of reads (optimized over distance and incurred mismatches.)
Effective genome size. This is the mappable genome size and should be changed according to the genome you are working with. In general, the effective size will be smaller than the actual size of the genome, because of repetitive sequences, etc. The effective genome sizes for some of the more common organisms are: human: 2.7e+9, mouse: 1.87e+9, C. elegans: 9e+7, fruitfly: 1.2e+8. Default: 2.7e+9.
Tag size. This is the length of your tags, which should be specified, otherwise MACS will take the length of your first 10 sequences as being representative of your dataset.
Band width. This is used in the first step of finding bimodal peaks and is expected to approximate the sonication fragment size.

MFOLD. Only regions above the selected range of high-confidence enrichment ratio of background to build model are returned. Using two numbers here, delimited by a comma, will return only those regions that have ratios within those limits.
Wiggle. Saves all the shifted tag file locations as a wiggle file. If you choose to save this information, you also need to tell MACS how far out to extend each fragment (default is the distance calculated between the forward and reverse strand peaks) and at what resolution to save the wiggle information (the default is 10 bases.) Note that wiggle tracks can be converted to BigWig format using the Convert Formats → Wig-to-bigWig tool, and visualized in your preferred genome browser.
Background lambda. The lambda is used to describe the Poisson distribution of tags along the genome. If this is checked, then the background lambda is used as the local lambda and MACS will not consider the local bias at peak candidate regions.
3 levels of regions. This determines the different levels of regions that are used to calculate the local lambda. A large region like 10000bps will capture the bias from long range effects like open chromatin domains.
Build shifting model. If this is not selected, then instead of looking for bimodal pairs of peaks, calculating the distance between them and moving each tag inwards by d/2, you can tell MACS by exactly how much you want the tags moved in a 3' direction. For example, if you knew that your protein footprint was 200bp, you could choose not to build the shifting model and instead set the Arbitrary shift size to 100 (this parameter becomes available once this option is selected).

Download the input file

1. Open the Get Data → Upload File tool.
2. Download the input file at
http://chagall.med.cornell.edu/galaxy/chipseq/G1E-estradiol.fastqsanger
3. This is a reduced dataset (chr19) looking for CTCF binding sites, derived from the G1E line, a GATA1 null-derived line used as a model for erythropoiesis. In this case, the cell line has been induced to differentiate by estradiol treatment.
4. Choose the fastqsanger option from the File Format menu.
5. Associate this input file with the mm9 genome build.
6. Click Execute.
7. Open the NGS: QC and Manipulation → FASTQ Summary Statistics tool and run a quality control check on the imported dataset.
8. Visualize the results with Graph/Display Data → Boxplot.
9. From the boxplot, it can be seen that the sequences are 36 bases long, and all quality scores are above 30, so we do not have to trim or discard any data.

Map the reads against the mouse genome

1. Open the NGS: Mapping → Map with Bowtie for Illumina tool.
2. Select the fastqsanger formatted reads dataset.
3. Select the mm9 Full genome.
4. Click Execute.

Find peaks using MACS

Once the reads have been mapped to the mouse genome, we can run the MACS program to determine where the peaks are.

1. Open the NGS: Peak Calling → MACS tool.
2. Change the experiment name so that the track will have a unique name when we visualize it.
3. Select the dataset of reads mapped by Bowtie.
4. Change the Tag Size to 36 (the length of the reads as determined in the QC step).
5. Change the genome size to "1.87e+9" (effective size for mouse genome).
6. Click Execute.

Once the MACS program finishes, there are two results files, one with the positions of the peaks and the other a more general HTML report file. The html report has links to various statistics files, as well as the log of the MACS run. This includes, at the end of the output, the number of peaks that were called. In this case, we found 750 peaks.

Find peaks, using control data

Since some of the peaks that are predicted may be due to experimental set up and sequencing biases, it is often useful to use a control dataset (i.e., one where no transcription factor is bound to the DNA) to detect such biases and eliminate them from our predictions.

1. Open the Get Data → Upload File tool.
2. Download the control input file at

http://chagall.med.cornell.edu/galaxy/chipseq/G1E-estradiolControl.fastqsanger

3. This is a control dataset, derived from the same G1E line as that experiment, that has also been induced to differentiate by estradiol treatment, but has no CTCF bound to the DNA.
4. Choose the fastqsanger option from the File Format menu.
5. Associate this input file with the mm9 genome build.
6. Click Execute.
7. Open the NGS: QC and Manipulation → FASTQ Summary Statistics tool and run a quality control check on the imported dataset.
8. Visualize the results with Graph/Display Data → Boxplot tool.
9. From the boxplot, it can be seen that the sequences are 36 bases long, and all the reads have a median quality score of at least 26, so we do not have to trim or discard any data.
10. Open the NGS: Mapping → Map with Bowtie for Illumina tool.
11. Select the fastqsanger formatted control reads dataset.
12. Select the mm9 Full genome as the reference genome.
13. Click Execute.

Now we can compare the mapped reads from both the experiment and control.

14. Open the NGS: Peak Calling → MACS tool.
15. Select the dataset of experimental mapped reads as the ChIP-Seq Tag File.
16. Select the dataset of control mapped reads as the ChIP-Seq Control File.
17. Change the Tag Size to 36 (the length of the reads as determined in the QC step).

18. Change the genome size to "1.87e+9" (effective size for mouse genome.)
19. Click Execute.

This time, we get 852 peaks. This indicates that corrections due to sequencing bias along the genome go both ways, both removing and adding peaks. Since our output is a BED file, we can visualize the peaks at the UCSC Genome Browser by choosing the Display at UCSC main link.

Comparison between conditions

We can also compare predicted peaks between different conditions. Here, we will identify sites that have differential binding across the differentiated and undifferentiated states.

First, create a workflow from the above analysis, such that it takes as input two fastqsanger datasets, one a control and the other the experiment, maps both of them with Bowtie against the mm9 mouse genome build, and then analyzes the reads with MACS to come up with a set of predicted peaks.

1. Import two more datasets using the Get Data → Upload File tool. These datasets are from the same cell line, but this time without estradiol treatment (i.e., it remains undifferentiated). The experimental dataset is:
http://chagall.med.cornell.edu/galaxy/chipseq/G1E-undifferentiated.fastqsanger
2. and the control dataset is:
http://chagall.med.cornell.edu/galaxy/chipseq/G1E-undifferentiatedControl.fastqsanger
3. Run a quality control check on both, using the NGS: QC and Manipulation → FASTQ Summary Statistics and Graph/Display Data → Boxplot tool.
4. Use these files as input into the workflow you just created.

Note. If Bowtie is taking too long to run, copy over the four datasets of Bowtie-mapped reads from the accessible history and run MACS on them.
https://main.g2.bx.psu.edu/u/luce/h/mappingresults

Once we have access to all the peaks called by MACS, we can compare the peaks from the two samples to identify CTCF sites that are a) found in both conditions (exposed and not exposed to estradiol); b) found in the differentiated state (+ER4) but not in the undifferentiated state (-ER4); c) found in the undifferentiated state but not in the differentiated state.
To obtain a dataset of all peaks found in both conditions:
1. Open the Operate on Genomic Intervals → Intersect tool.
2. Choose the dataset of CTCF peaks identified with estradiol treatment as the first dataset.
3. Choose the dataset of CTCF peaks identified without estradiol treatment as the second dataset.
4. Click Execute.

To get a dataset of all CTCF peaks identified in the cell line when treated with estradiol, but not in the undifferentiated line:

5. Open the Operate on Genomic Intervals → Subtract tool.
6. Choose the dataset of CTCF peaks identified without estradiol treatment as the first dataset.
7. Choose the dataset of CTCF peaks identified with estradiol treatment as the second
8. Click Execute.

And finally, to get a dataset of all CTCF peaks identified in the undifferentiated cell line that disappear when treated with estradiol:

9. Open the Operate on Genomic Intervals → Subtract tool.
10. Choose the dataset of CTCF peaks identified with estradiol treatment as the first dataset.
11. Choose the dataset of CTCF peaks identified without estradiol treatment as the second
12. Click Execute.

We can create a custom track with information on all three datasets above for visualization in the UCSC Genome Browser.

1. Open the Graph / Display Data → Build Custom Track tool.
2. Add a new track for each of the three datasets above.
3. Give each track a different name and color, and specify pack as the visibility option.
4. Once the custom track has been created, choose the link to display it at UCSC in the history preview window. A region where examples of peaks from all three comparisons can be seen is chr19:3,156,415-3,556,414.

Finally, we might like to identify those sites predicted to bind CTCF in the estradiol-treated cell line that lie near promoters.

1. Open the Get Data → UCSC Main tool.
2. Select mouse as the organism and mm9 as the genome build.
3. Select the RefSeq genes track.
4. Make sure the BED output format is selected and the Send to Galaxy checkbox is checked.
5. Click Get Output.
6. Select the "promoters" by selecting 1000 bases upstream (or you can get the gene data now and then transform it with the Operate on Genomic Intervals → Get flanks tool.)
7. Click the Send query to Galaxy button.
8. Open the Operate on Genomic Intervals → Join tool.
9. Select the peaks identified for the cell line with estradiol treatment as the first dataset and the promoter dataset as the second.
10. Click Execute.

Running Galaxy in the Cloud

The Galaxy wiki page http://wiki.g2.bx.psu.edu/Admin/Cloud for setting up an instance of Galaxy in the Amazon cloud is very comprehensive and gives detailed step-by-step instructions.

Before starting to set up a Galaxy instance, you will have to set up and verify an Amazon Web Services account. This account gives you access to the Amazon cloud service. The payment scheme is such that you only pay for the resources you use.

To start a Galaxy CloudMan cluster, we need to start a master instance which will be used to control all of the needed services as well as worker instances which run the analysis jobs.

1. Log in to your Amazon Web Services account at http://aws.amazon.com/
2. From the My Account / Console tab, choose AWS Management Console and sign in.
3. Go to the Security Credentials options on the same tab and make a note of your Access Key ID and your Secret Access Key.
4. Go to the EC2 tab.
5. Make sure your AWS Region is set to US East (Virginia).
6. Click Launch Instance.
7. Choose the Classic Wizard in the pop-up window.
8. Click on the Community AMIs tab and select ami-46d4792f as your AMI.
9. Set Number of Instances to 1. This is the head node of the cluster.
10. For the instance type, select at least a **large** node.
11. Choose any availability zone. It does not matter which zone you choose the first time, but once selected, you must select this same zone every time you instantiate the given cluster.
12. Click Continue.
13. Enter your user data in the format below, which specifies the desired name of the cloud cluster and provides Galaxy CloudMan with user account information. Note that there must be a space between the colon and the value of the field

    ```
    cluster_name: <DESIRED CLUSTER NAME>
    password: <DESIRED Galaxy CloudMan WEB UI PASSWORD>
    access_key: <YOUR AWS ACCESS KEY>
    secret_key: <YOUR AWS SECRET KEY>
    ```

14. The next popup allows you to Set Metadata Tags for this instance. Set the Name tag for this instance, as that will appear in the instance list of the AWS EC2 Management Console.
15. Choose the key pair you created during the initial setup.
16. Select the security group in the initial setup, and the default group and continue.
17. Check your entries one more time, and then Launch the instance and wait (about 5 minutes on average) for the instance and CloudMan to boot.

18. Go to the AWS management console, and click Instances, then select the instance you just launched. You need to wait until the instance state is Running, and Status checks says "2/2 checks passed".
19. Copy the URL that appears at the top of the instance details panels into a web browser and hit enter. You should see a "Welcome to Galaxy on the cloud" page.
20. Click on the "please use the cloud console" link.
21. Login to the instance by entering the password you specified in User Data when starting the master instance.
22. The first time you login to this instance's Galaxy CloudMan interface, an "Initial Cluster Configuration" popup will appear, asking you how much disk space you want to allocate for your data. This can be increased later.
23. Click on Start Cluster.
24. It will take a few minutes for the master node to come up. The Access Galaxy button will go from grayed out to active. Disk status will show a disk with a green plus on it. Service status for both Applications and Data will be green (instead of yellow).
25. Once the Access Galaxy button is no longer grayed out, you can add nodes to the cluster by either clicking the Add Nodes button.
26. Once the worker nodes are up, click the Access Galaxy button. This opens up a new window with Galaxy on the Cloud. You are now running an elastic and fully loaded and populated version of Galaxy on the cloud.
27. Register yourself as a new user on the cloud instance and continue as you normally would.